CEA-R-6599



ISSN 0429 - 3460

COMMISSARIAT À L'ÉNERGIE ATOMIQUE
ET AUX ÉNERGIES ALTERNATIVES

# *TRIPOLI-4® VERSION 12 USER GUIDE*

***par***

***Equipe TRIPOLI-4®***

CEA CENTRE DE PARIS-SACLAY/SACLAY
DIRECTION DES ÉNERGIES
INSTITUT DES SCIENCES APPLIQUÉES ET
DE LA SIMULATION POUR LES
ÉNERGIES BAS CARBONE
DÉPARTEMENT DE MODÉLISATION DES SYSTÈMES
ET STRUCTURES
SERVICE D'ÉTUDES DES RÉACTEURS ET
DE MATHÉMATIQUES APPLIQUÉES

DIRECTION DES ÉNERGIES




DIRECTION DES SYSTÈMES
D'INFORMATION
CEA / SACLAY 91191 GIF-SUR-YVETTE CEDEX FRANCE

**2025**

RAPPORT
CEA-R-6599

- Rapport CEA-R-6599 -

CEA Centre de Paris-Saclay/Saclay
Direction des Énergies
Institut des Sciences Appliquées et de la Simulation pour les énergies bas carbone
Département de Modélisation des Systèmes et Structures
Service d'Études des Réacteurs et de Mathématiques Appliquées

# TRIPOLI-4® VERSION 12 USER GUIDE

par

Equipe TRIPOLI-4®

- Avril 2025 -

**RAPPORT CEA-R-6599 – Equipe TRIPOLI-4®**

**"Guide de l'utilisateur TRIPOLI-4® version 12"**

***Résumé :*** TRIPOLI-4® est un code de transport généraliste. Il simule le transport des neutrons, photons, électrons et positrons par la méthode de Monte-Carlo dans des géométries tridimensionnelles. Les domaines d'application principaux sont la radioprotection, la sûreté-criticité, la physique des réacteurs (y compris l'évolution à partir de la version 10) et l'instrumentation. TRIPOLI-4® est compatible avec les bibliothèques de sections efficaces ponctuelles au format ENDF/B en provenance des évaluations JEF2, ENDF/B-VI, JEFF3, ENDF/B-VII, JENDL3, ENDF/B-VIII, JENDL4, etc. Pour les neutrons thermiques le modèle gaz libre comme les S(alpha, béta) sont disponibles. Des méthodes performantes et automatisées de réduction de variance permettent de traiter des problèmes présentant de très fortes atténuations de flux. Grâce à un mode parallèle à la fois robuste et performant, des simulations sur plusieurs processeurs sont aisément conduites sur des machines massivement parallèles. Des outils d'aide au diagnostic, graphiques et algorithmiques, permettent de corriger rapidement les jeux de données erronés. TRIPOLI-4® dispose d'une base de V&V étendue, tant en comparaison calcul-calcul que calcul-mesure. C'est un outil qualifié pour la R&D, l'enseignement et l'industrie.

**RAPPORT CEA-R-6599 – Equipe TRIPOLI-4®**

**"TRIPOLI-4® version 12 user guide"**

***Abstract :*** TRIPOLI-4® is a general-purpose particle-transport code. It simulates the transport of neutrons, photons, electrons, and positrons using the Monte Carlo method in three-dimensional geometries. Its main application areas are radiation shielding, criticality safety, reactor physics (including depletion calculations, since version 10), and nuclear instrumentation. TRIPOLI-4® is compatible with continuous-energy cross-section libraries in ENDF/B format from several evaluations, encompassing JEF2, ENDF/B-VI, JEFF3, ENDF/B-VII, JENDL3, ENDF/B-VIII, or JENDL4. For thermal neutrons, both the free gas and S(alpha, beta) models are available. Powerful and automated variance reduction methods allow handling configurations with very high flux attenuations. Thanks to a robust and efficient parallel mode, simulations with multiple processors can be easily run on massively parallel machines. Graphical and algorithmic diagnostic tools allow for the fast correction of mistakes in user data files. TRIPOLI-4® has an extensive V&V database, both for code-to-code and code-to-measurement comparisons. It is a qualified tool for R&D, education, and industry.

***2025 – Commissariat à l'Énergie Atomique et aux Énergies Alternatives – France***

# Contents

**1 GENERALITIES** **20**
1.1 General description of the code . . . 20
1.2 Purpose of this document . . . 20
1.3 Execution . . . 20
1.4 Displaying physical units . . . 22
1.5 Version number . . . 22
1.6 Verification and Validation tool VALJEAN . . . 23

**2 CROSS-SECTIONS** **24**
2.1 Nuclear data file names . . . 24
2.2 Optimizing cross-section use . . . 25

**3 DEFINING PATH AND DICTIONARY FILES** **26**
3.1 The path file . . . 26
3.1.1 Legacy format . . . 26
3.1.2 New format . . . 27
3.2 The dictionary file . . . 28

**4 GENERAL CONSIDERATIONS : DATA, CONVENTIONS, UNITS, FRAMES** **31**
4.1 The default units . . . 31
4.2 Unit changes . . . 32
4.3 Conventions used in this manual . . . 32
4.4 Character style conventions . . . 34
4.5 Defining frames . . . 35
4.6 List layout . . . 38
4.7 List of calculation commands . . . 38

**5 EXAMPLE** **42**

**6 USING COMMENTS AND INSERTING FILES** **44**
6.1 Comments . . . 44
6.1.1 Comments using COMMENT . . . 44
6.1.2 C type comments . . . 44
6.1.3 C++ type comments . . . 45
6.1.4 Restrictions . . . 45
6.2 Inserting files . . . 45

**7 GEOMETRY DEFINITIONS** **46**
7.1 Geometrical accuracy . . . 48
7.2 Global structure . . . 49
7.3 Different ways to define volumes . . . 50
7.3.1 Combinatorial geometry . . . 50
7.3.2 Surface-based geometry . . . 50
7.3.3 Definition using a rotation . . . 50
7.3.4 Defining a lattice . . . 50
7.3.5 Redefining a cell . . . 50
7.3.6 Operators, virtual volumes, and syntax for a volume . . . 51
7.4 Defining a type . . . 52
7.4.1 Spheres . . . 52
7.4.2 Boxes . . . 53
7.4.3 Cylinders parallel to an axis . . . 54
7.4.4 Generic cylinders . . . 54
7.4.5 Cones . . . 55
7.4.6 Generic parallelepipeds . . . 56
7.4.7 Hexagonal prisms . . . 56
7.4.8 Tori . . . 57
7.5 Surface data definitions . . . 58
7.5.1 Plane parallel to (OY,OZ) . . . 59
7.5.2 Plane parallel to (OX,OZ) . . . 59
7.5.3 Plane parallel to (OX,OY) . . . 59
7.5.4 Generic planes . . . 60
7.5.5 Spheres . . . 60
7.5.6 A cylinder that is parallel to axis OX . . . 60
7.5.7 A cylinder that is parallel to axis OY . . . 61
7.5.8 A cylinder that is parallel to axis OZ . . . 61
7.5.9 A cylinder with a generic axis . . . 61
7.5.10 A cone whose axis is parallel to axis OX . . . 61
7.5.11 A cone whose axis is parallel to axis OY . . . 62
7.5.12 A cone whose axis is parallel to axis OZ . . . 62
7.5.13 A cone with a generic axis . . . 62
7.5.14 An elliptic torus . . . 62
7.5.15 Generic quadratic surface . . . 64
7.5.16 Surface transformation . . . 64
7.6 Defining simple volumes . . . 65
7.6.1 Defining simple volumes using predetermined types . . . 65
7.6.1.1 Sphere, cylinder with axis parallel to Ox, cylinder with axis parallel to Oy, cylinder with axis parallel to Oz, box . . . 66
7.6.1.2 Cone . . . 67
7.6.2 Defining simple volumes using surfaces . . . 67
7.7 Defining rotation type volumes . . . 69
7.8 Defining operators related to volumes . . . 70
7.8.1 The SMASH operator . . . 71
7.8.1.1 Properties of SMASH . . . 72
7.8.1.2 The SMASH operator and virtual volumes . . . 72
7.8.1.3 Rotation and operator SMASH . . . 73
7.8.2 The VMINUS operator . . . 73

7.8.2.1 Restrictions on the SMASH and VMINUS operators . . . 75
7.8.2.2 Transitivity, link with the SMASH and FICTIVE operators . . . 76
7.8.3 The INTE Operator . . . 78
7.8.4 The UNION Operator . . . 78
7.9 Operator priority . . . 80
7.9.1 Prioritization of VMINUS, INTE, and UNION . . . 80
7.9.2 Prioritization of (INTE, UNION, VMINUS), and SMASH . . . 80
7.10 Defining a lattice . . . 81
7.10.1 The concept of lattices . . . 81
7.10.2 Syntax . . . 82
7.10.3 Lattice restrictions . . . 83
7.10.4 Lattices and the SMASH operator . . . 84
7.10.5 Cell numbering . . . 84
7.10.6 Lattice of lattices . . . 86
7.10.7 Volume definitions based on cells . . . 87
7.10.7.1 Replacing a volume of a cell . . . 87
7.10.7.2 Replacing a cell of a lattice . . . 87
7.10.7.3 Replacing a sub-cell of a lattice of lattices . . . 89
7.10.7.4 Replacing a cell using EXCEPT and VOL_EXCEPT . . . 91
7.10.8 The virtual cell trick . . . 99
7.11 Copies of volumes . . . 100
7.12 2D-view display . . . 100
7.12.1 Choice of colors . . . 100
7.12.2 2D-cross sectional graph request . . . 101
7.13 Geometry Checks . . . 103
7.13.1 Combinatorial geometry checks . . . 103
7.13.2 Requesting Checks . . . 103
7.13.3 1D scan of geometry . . . 105
7.13.4 1D scan of geometry-composition association . . . 105
7.13.5 Requesting Checks Using Tracking Routines . . . 106
7.14 Influence of data input on calculation time . . . 107
7.14.1 Connectivity maps . . . 107
7.14.1.1 Usage . . . 107
7.14.1.2 Map Format . . . 108
7.14.2 Surface-based geometries . . . 109
7.14.3 Combinatorial geometries . . . 109

**8 THE VOLSURF COMMAND 118**

**9 BOUNDARY CONDITIONS 120**
9.1 Defining boundary conditions . . . 120
9.1.1 Leakage . . . 120
9.1.2 Reflection . . . 121
9.1.3 Albedo . . . 121
9.1.4 Translation . . . 121
9.1.5 Rotation . . . 121
9.1.6 Cosine . . . 122
9.2 Sequence of instructions . . . 123
9.3 Additional boundary surfaces . . . 125

**10 DEFINING COMPOSITIONS** **126**
10.1 Pointwise cross-sections in ENDF format . . . 128
10.1.1 Defining compositions by supplying the number of atoms per 1e-24 cm3 . . . 128
10.1.2 Defining compositions by entering the density (in g/cm3) . . . 129
10.2 Multigroup homogenized cross-sections . . . 131
10.2.1 Self-shielded homogenized multigroup cross-sections from APOLLO2 and TRIPOLI-4® . . . 131
10.3 Multigroup Cross-Sections in GENDF Format . . . 132
10.4 TRIPOLI-4®-NJOY Coupling . . . 132

**11 DEFINING VOLUME-COMPOSITION ASSOCIATIONS** **136**

**12 DEFINING SOURCES** **138**
12.1 Defining a source . . . 138
12.2 Spatial distribution . . . 140
12.2.1 Point sources . . . 140
12.2.2 Distributed sources . . . 140
12.2.2.1 Definition context . . . 141
12.2.2.2 Syntax for a factorized source . . . 141
12.2.2.3 Surface, analytical and tabulated distributions . . . 144
12.3 Angular distribution . . . 153
12.3.1 Dirac . . . 153
12.3.2 Uniform distribution . . . 154
12.3.3 Ray distribution . . . 154
12.3.3.1 Discrete mu . . . 154
12.3.3.2 Discrete phi . . . 154
12.3.4 Tabulated distributions . . . 155
12.3.4.1 Mu tabulated distribution . . . 155
12.3.4.2 Syntax . . . 155
12.3.4.3 Phi tabulated distribution . . . 157
12.3.4.4 Syntax . . . 157
12.3.5 2D Tabulated distribution . . . 158
12.3.5.1 Syntax . . . 158
12.3.6 Analytical distribution . . . 159
12.4 Energy distribution . . . 161
12.4.1 Dirac . . . 161
12.4.2 Line spectrum . . . 161
12.4.3 Uniform distribution . . . 161
12.4.4 Gauss spectrum . . . 162
12.4.5 Watt spectrum . . . 162
12.4.6 Maxwell spectrum . . . 163
12.4.7 Tabulated spectrum . . . 163
12.4.8 Analytical distribution . . . 165
12.4.9 Fission spectrum from the evaluation . . . 166
12.5 Time distribution . . . 166
12.5.1 Dirac . . . 167
12.5.2 Uniform distribution . . . 167
12.5.3 Gauss distribution . . . 167
12.5.4 Tabulated distribution . . . 168
12.5.5 Analytical distribution . . . 168
12.6 Sources taken from source files . . . 169

12.6.1 Volume particle sources . . . 169
12.6.2 Surface particle sources . . . 170
12.6.2.1 Syntax . . . 170
12.6.2.2 Zooming approach principle . . . 172
12.6.2.3 Parallel mode . . . 172
12.7 Complete source examples . . . 173
12.7.1 Isotropic point source with no time dependence . . . 173
12.7.1.1 Line spectrum . . . 173
12.7.1.2 Interpolated spectrum . . . 175
12.7.2 Distributed isotropic source without time dependence . . . 175
12.8 External sources . . . 179
12.8.1 Description . . . 179
12.8.2 Example . . . 181
12.9 Multi-particle sources . . . 182
12.9.1 Uncorrelated source particles . . . 183
12.9.2 Correlated source particles . . . 183
12.10 Sources for activation calculations . . . 187
12.11 GEANT4 particle sources . . . 187
12.12 Spontaneous fission sources with the FIFRELIN fission model . . . 187

**13 DEFINING RESPONSE TYPES 189**
13.1 The available responses . . . 189
13.2 Additional data for reaction rates . . . 194
13.3 Sum of reaction rates . . . 198
13.4 DATA additional data . . . 198
13.5 Additional data for a weighted current . . . 201
13.6 Macroscopic reaction rates . . . 201
13.7 Deposited energy . . . 202
13.7.1 Reaction Q-values for energy deposition . . . 202
13.7.2 Fission energy . . . 203
13.8 Kinematic limits . . . 204
13.9 DPA . . . 205
13.10 Gas production . . . 207
13.11 Fission spectrum . . . 207
13.12 Fixed source emission . . . 208
13.13 Deposited charge . . . 208
13.14 Special option to except a part of the contribution for secondary particles . . . 209

**14 DEFINING GRIDS 211**

**15 DEFINING SCORES 214**
15.1 Notes regarding score definitions . . . 214
15.2 Standard Monte Carlo estimators . . . 214
15.2.1 Volume-based estimators . . . 214
15.2.2 Surface-based estimators . . . 215
15.3 Next-event estimators . . . 215
15.3.1 Point-based estimators . . . 215
15.3.2 Surface-based estimators . . . 215
15.3.3 Remarks . . . 216
15.4 Variance evaluation . . . 216

15.5 Geometrical elements . . . 216
15.6 Filtering . . . 216
15.7 Defining the score . . . 217
15.8 Extended meshes . . . 224
15.8.1 Uniform window syntax . . . 226
15.8.2 Nonuniform window syntax . . . 226
15.8.3 Variance reduction and TRACK estimators . . . 227
15.8.4 Note on COLLISION estimator . . . 227
15.8.5 Note on mesh edition results . . . 227
15.9 Examples of score requests . . . 229
15.9.1 Example on a simple geometry . . . 229
15.9.2 Using the keyword ENTROPY . . . 233

**16 DEFINING SCORES DECOMPOSED OVER REAL SPHERICAL HARMONICS 235**

**17 VARIANCE REDUCTION TECHNIQUES (BIASING) 238**
17.1 The concept of a weight . . . 240
17.2 Spatial biasing . . . 241
17.2.1 Biasing energy groups . . . 241
17.2.2 Detectors . . . 241
17.2.3 Calculation of spatial importance . . . 243
17.2.3.1 For biasing with discrete detectors . . . 243
17.2.3.2 For an analytical biasing (plane, cylindrical, or spherical) . . . 244
17.2.3.3 For biasing with a splitting sphere . . . 244
17.2.4 Mesh and importance maps in TRIPOLI-4® . . . 245
17.2.4.1 Cartesian mesh . . . 245
17.2.4.2 Cylindrical mesh . . . 245
17.2.4.3 Refining the mesh . . . 246
17.2.4.4 Uniform mesh syntax . . . 248
17.2.4.5 Nonuniform mesh syntax . . . 248
17.2.5 Bias parameter k . . . 249
17.2.6 Smoothing option . . . 250
17.3 Directions of interest . . . 250
17.4 Energy biasing . . . 252
17.5 Time biasing . . . 252
17.6 Special biasing . . . 253
17.6.1 Splitting when entering specific volumes VOLUME_SPLITTING . . . 253
17.6.2 Optical biasing . . . 254
17.6.3 Russian roulette control . . . 254
17.6.4 Special source biasing . . . 255
17.7 Variance reduction techniques for coupled neutron photon simulations . . . 255
17.8 Graphical display . . . 257
17.9 Hints concerning biasing . . . 258
17.9.1 Hints concerning Inipond module . . . 258
17.9.1.1 The automatic case . . . 258
17.9.1.2 Mesh . . . 259

**18 THE ADAPTIVE MULTILEVEL SPLITTING VARIANCE REDUCTION TECHNIQUE 261**
18.1 SYNTAX . . . 262
18.2 TARGET DEFINITON . . . 262
18.2.1 SINGLE TARGET . . . 262
18.2.2 MULTIPLE TARGETS . . . 262
18.2.3 SCORING OUTSIDE TARGETS . . . 263
18.3 SPLITTING PROPORTION . . . 263
18.4 AMS OPTIONS . . . 263
18.4.1 CROSSING POINTS SPLITTING . . . 263
18.4.2 COLLISION RESAMPLING . . . 264
18.5 IMPORTANCE FUNCTION . . . 264
18.5.1 Importance maps . . . 264
18.5.2 SPATIAL IMPORTANCES . . . 265
18.5.2.1 `POINT` . . . 265
18.5.2.2 LINE . . . 265
18.5.2.3 PLANE . . . 266
18.5.2.4 SPHERE . . . 266
18.5.2.5 CYLINDER . . . 266
18.5.2.6 RING . . . 267
18.5.2.7 PATH . . . 267
18.5.2.8 VOLUME IMPORTANCE WEIGHTING . . . 268
18.6 RELATED OPTIONS IN OTHER COMMAND BLOCKS . . . 268
18.6.1 ANALOG . . . 269
18.6.2 MONITORING . . . 269
18.6.3 SMOOTHING . . . 269
18.7 RESTRICTIONS AND FORBIDDEN OPTIONS . . . 269

**19 THE WEIGHT WINDOWS VARIANCE REDUCTION TECHNIQUE 270**
19.1 SYNTAX . . . 271
19.2 WEIGHT WINDOWS FILE SYNTAX . . . 271
19.3 SPECIFIC SYNTAX WHEN DATA ARE PROVIDED BY ADVANTG CODE . . . 273
19.4 OTHER RELATED SIMULATION OPTIONS . . . 273

**20 THE CADIS VARIANCE REDUCTION TECHNIQUE 274**
20.1 Principle of the CADIS method . . . 274
20.2 Modelling and approximations . . . 275
20.3 Syntax . . . 275
20.3.1 IDT parameters . . . 276
20.3.1.1 `TARGET` keyword . . . 277
20.3.1.2 `SPECTRUM` keyword . . . 277
20.3.1.3 `CONVERGENCE` keyword . . . 278
20.3.1.4 `MICROSCOPIC_CROSS_SECTIONS` block . . . 278
20.3.2 Other keywords . . . 279
20.4 Caveats . . . 280
20.5 Example . . . 282

**21 THE HOMOGENIZATION COMMAND 284**

**22 DEFINING NUMERICAL SIMULATION PARAMETERS 286**
22.1 Simulation parameters . . . 286
22.2 Syntax . . . 291
22.3 Option KEEP_RESULT . . . 295
22.4 Option XML_EXPORT . . . 296
22.5 Option RANDOM . . . 296
22.6 Computing the kinetics parameters . . . 296
22.7 Neutron importance (critical adjoint flux) . . . 300
22.8 Critical parameter search . . . 303
22.8.1 Albedo search . . . 303
22.8.2 Boron search . . . 304
22.9 Slowing down current . . . 306
22.10 Six factor formula . . . 306
22.11 Diven dispersion factor . . . 307
22.12 DYNAMIC (time-dependent simulation) options . . . 309
22.12.1 Time mesh for forced precursor decay . . . 310
22.12.2 Population control . . . 311
22.12.2.1 Russian roulette & splitting . . . 311
22.12.2.2 Combing . . . 311
22.12.3 Time mesh for scores . . . 311
22.12.4 Relative particle importance . . . 312
22.12.5 Critical source . . . 313
22.12.6 Adjusting the reactivity . . . 314
22.12.7 Changing geometries . . . 314
22.12.8 Branchless collisions . . . 315
22.13 EPITHERMAL BROADENING options . . . 315
22.13.1 SVT model with a different threshold energy E_threshold . . . 316
22.13.2 Doppler Broadening Rejection Correction (DBRC) . . . 317
22.13.3 Weight Correction Method (WCM) . . . 318
22.14 Electromagnetic shower options . . . 319
22.15 Option PHOTON_NEUTRON_RATIO . . . 321
22.16 Option ALL_COLLIDED_FLUX_PT_POINTS . . . 321
22.17 Option SUPER_HISTORY . . . 321
22.18 Option WIELANDT_METHOD . . . 322
22.19 The MULTIPLE simulation mode (neutronics-shielding coupling) . . . 323
22.19.1 Objective . . . 323
22.19.2 Principle . . . 323
22.19.3 History of the multiple simulation mode . . . 323
22.19.4 Syntax . . . 324
22.20 Storage of particle events . . . 325
22.21 Counting holes within the simulation options . . . 325
22.21.1 Options HOLE_COUNT, HOLE_MAP, HOLE_VOLU_MAP . . . 325
22.21.2 Procedure for testing geometry . . . 326

**23 CALCULATION VERIFICATION DEFINITIONS 329**
23.1 Volume calculations . . . 329
23.2 Density calculations . . . 331
23.3 MASS calculations . . . 332

**24 KIJ MATRIX CALCULATION** **335**
24.1 Syntax . . . 335
24.2 Output . . . 336

**25 DISPLAY OF COLLISIONS SITES** **337**
25.1 Syntax . . . 337
25.1.1 Types of particles . . . 338
25.1.2 Variation of the parameter `height` . . . 339
25.1.3 Display for "full" 2D-views or contour-2D-views of the geometry . . . 339
25.1.4 Option `DISCARD` . . . 340
25.1.5 Options `SCORE` and `SCORE_SEGMENTS` . . . 340
25.1.6 Option `SOURCE` . . . 340
25.1.7 Option `NB_COLLISION` . . . 341
25.1.8 Optional definitions for an energy grid and related colors . . . 341
25.1.8.1 Definition of an energy grid, keyword `GRID` : . . . 341
25.1.8.2 Color definition, keyword `COLOR` : . . . 341
25.2 Order of apparition for the display windows . . . 342
25.3 Number of particles in the simulation . . . 342
25.4 Number of projected points . . . 342

**26 THE OUTPUT FILE** **343**
26.1 Printing out the output file . . . 343
26.2 When a variance reduction technique is used . . . 343
26.3 Outputs relating to sources . . . 345
26.4 Outputs for each batch . . . 346
26.5 Complete outputs . . . 346
26.6 Criticality calculations . . . 346
26.7 Sub-criticality calculations . . . 347

**27 STORING PARTICLES** **348**

**28 DEFINING PERTURBATIONS** **351**
28.1 General . . . 351
28.2 Data on a perturbation . . . 351
28.3 Density and concentration perturbations . . . 352
28.4 Partial cross-section perturbation . . . 354
28.4.1 Syntax . . . 354
28.4.2 The carry-over file . . . 356
28.5 Covariance perturbations . . . 359

**29 STANDARD PERTURBATION THEORY** **361**
29.1 Reactivity perturbations . . . 361
29.1.1 Iterated Fission Probability Syntax . . . 362
29.1.2 Wielandt Syntax . . . 366
29.1.3 Super-History Syntax . . . 367
29.2 k-eigenvalue sensitivity coefficients . . . 369
29.2.1 Iterated Fission Probability Syntax . . . 370
29.2.2 Wielandt Syntax . . . 374
29.2.3 Super-History Syntax . . . 376
29.2.4 Comments . . . 377

**30 GENERALIZED PERTURBATION THEORY** **378**
30.1 Reaction rate ratio sensitivity coefficients . . . 378
30.1.1 Super-History Syntax . . . 379
30.2 Limitations . . . 381

**31 SPECIAL OUTPUTS** **383**
31.1 XVGR plots . . . 383
31.2 Cross-section data and plots . . . 385
31.3 Information on simulated nuclei . . . 386
31.4 T4G dedicated outputs . . . 386
31.4.1 Mesh outputs . . . 386
31.4.2 Point event outputs . . . 387

**32 CONNECTION WITH THE DARWIN/PEPIN2 CODE** **389**

**33 PARALLELISM** **391**
33.1 Introduction . . . 391
33.2 Launching tools . . . 393
33.2.1 Mpirun or Srun tools . . . 393
33.2.2 Lpar* tools . . . 394
33.3 The parallelism file . . . 394
33.3.1 Heterogeneous clusters . . . 395
33.3.2 Calculation on a machine with a resource manager . . . 395
33.3.2.1 Results . . . 396
33.3.2.2 Advices regarding execution . . . 396
33.4 Differences between DISCARD processing in mono-processor and parallel calculations . . . 397
33.5 Incompatibilities . . . 398

**34 COLLISION FILES** **399**
34.1 Creation of a collision file . . . 400
34.2 Use of a collision file . . . 401

**35 GREEN'S FUNCTION FILES** **402**
35.1 Objective . . . 402
35.2 Principle . . . 402
35.3 Limitations and hypotheses . . . 403
35.4 Command structure for user data . . . 404
35.4.1 Creation phase . . . 404
35.4.2 Exploitation phase . . . 405
35.5 Use of parallelism . . . 407
35.6 Aspect of the output file . . . 407
35.6.1 Creation phase . . . 407
35.6.2 Exploitation phase . . . 407
35.7 Incompatibilities . . . 408

**36 TRIPOLI-4/ROOT COUPLING** **409**
36.1 Installation procedure . . . 409
36.2 Using a ROOT geometry . . . 409
36.2.1 Defining the geometry . . . 409
36.2.1.1 Defining the geometry via a file . . . 410

36.2.1.2 Defining the geometry directly in C++ or Python code . . . 411
36.2.2 Defining boundary conditions . . . 412
36.2.3 Defining a score . . . 412
36.2.4 How to check a ROOT geometry . . . 413
36.2.5 Caveat: use of overlapping nodes . . . 413
36.2.6 Example . . . 413
36.3 ROOT post-processing: recording particle tracks . . . 417

**37 ENERGY CUT PER VOLUME 418**

**38 VISUALIZER T4G 420**
38.1 Introduction to T4G . . . 420
38.2 How to launch T4G . . . 421
38.3 2D visualization . . . 423
38.3.1 Geometry retrieval . . . 423
38.3.1.1 "Pursuit" mode . . . 423
38.3.1.2 "Position" mode . . . 423
38.3.1.3 "Contours" mode . . . 423
38.3.2 Display of the color policy . . . 423
38.3.2.1 "Volumes values" file . . . 424
38.3.2.2 "Colors" file . . . 425
38.3.3 Tripoli-4 scores, meshes and points plot . . . 425
38.3.3.1 Scores plot . . . 426
38.3.3.2 Meshes plot . . . 427
38.3.3.3 Points plot . . . 428
38.3.4 Window . . . 429
38.3.4.1 Definition . . . 429
38.3.4.2 Default window . . . 430
38.3.4.3 Command line option . . . 430
38.3.4.4 Window file . . . 430
38.3.5 Available keys in 2D mode . . . 430
38.4 3D visualization . . . 434
38.4.1 Launch 3D visualization . . . 434
38.4.2 Basic features . . . 434
38.4.2.1 Exclusion of volumes . . . 434
38.4.2.2 Manipulation . . . 434
38.4.3 Hiding a part of a Root geometry with a simple Tripoli-4 geometry . . . 434
38.4.4 View and screen . . . 436
38.4.4.1 View . . . 436
38.4.4.2 Screen . . . 436
38.4.5 Available keys in 3D mode . . . 437
38.5 Configuration files . . . 439

**39 T4TOMED TOOL 440**
39.1 Starting with T4toMED . . . 440
39.1.1 How to launch T4toMED . . . 440
39.1.2 T4toMED options . . . 440
39.1.3 Particular data . . . 441

**40 DEPLETION CALCULATION** **443**
40.1 Depletion calculation input file . . . 443
40.1.1 TRIPOLI-4 input file . . . 443
40.1.2 Energy grids file . . . 454
40.1.3 Initial compositions file . . . 454
40.1.4 Irradiation history file . . . 454
40.2 Launching independent simulations . . . 455
40.3 Restart calculation . . . 456
40.4 Using the source from the previous depletion step . . . 457
40.5 Results and post-processing . . . 457
40.5.1 Content of the output directory . . . 457
40.5.2 Content of the ROOT output file . . . 458
40.5.3 Post-processing . . . 460
40.5.3.1 Post-processing a unique ROOT output file using *BurnupResults* class . . . 460
40.5.3.2 Post-processing several ROOT output files using the *MeanBurnupResults* class . . . 463
40.5.3.3 Producing TRIPOLI-4 composition files from ROOT outputs . . . 466

**41 ACTIVATION CALCULATION** **468**
41.1 Input for an activation calculation . . . 468
41.1.1 TRIPOLI-4 input file . . . 468
41.1.2 Homogenized cells . . . 471
41.2 Calculation steps . . . 472
41.2.1 Step 1: activation . . . 472
41.2.2 Step 2: photon transport . . . 472
41.2.3 Parallelization of the activation calculation . . . 474
41.2.4 Content of the ROOT output file . . . 474
41.3 Uncertainties . . . 476

**42 TRIPOLI-4/GEANT4 COUPLING** **478**
42.1 Requirements on the GEANT4 compilation phase . . . 478
42.2 Using GEANT4 geometries in TRIPOLI-4 . . . 478
42.2.1 Wrapping a GEANT4 geometry with `g4so` . . . 479
42.2.2 Building the geometry wrapper by hand . . . 480
42.2.3 Declaring the GEANT4 geometry in the TRIPOLI-4 input file . . . 481
42.3 Visualizing the geometry with T4G . . . 482
42.4 Physics coupling . . . 482
42.4.1 Syntax . . . 483
42.5 Using GEANT4 particle sources in TRIPOLI-4 . . . 483
42.5.1 Wrapping a GEANT4 source with `g4so` . . . 484
42.5.2 Building the source wrapper by hand . . . 484
42.5.3 Declaring the GEANT4 source in the TRIPOLI-4 input file . . . 486
42.6 Wrapping sources and geometries in the same shared library . . . 486
42.7 Scores . . . 486
42.8 Support for GEANT4 UI commands . . . 486
42.9 Troubleshooting . . . 487

**43 TRIPOLI-4/MORET COUPLING** **489**
43.1 Using a MORET geometry . . . 489
43.2 Using MORET boundary conditions . . . 490

**44 SCHEDULER** **491**
44.1 KIJ acceleration of critical calculations . . . 492
44.2 Specialised tasks . . . 493
44.2.1 Cleaning of results . . . 493
44.2.2 Initialising sources from KIJ matrix fundamental . . . 493

**45 GENERATOR OF STOCHASTIC GEOMETRIES (CASTOR)** **494**
45.1 Introduction . . . 494
45.2 Construction of random geometries . . . 495
45.2.1 Type of geometries (mandatory) . . . 495
45.2.2 Number of realizations . . . 496
45.2.3 Container . . . 496
45.2.4 Coloring probabilities (volume ratios) . . . 498
45.2.5 Geometrical specifications (tessellation density or radius distribution) . . . 499
45.2.5.1 Poisson and Poisson-Box geometries . . . 499
45.2.5.2 Spherical inclusions . . . 500
45.2.6 Construction algorithm . . . 500
45.3 Export input files for TRIPOLI-4 and T4G . . . 501
45.3.1 Activation and target repository . . . 501
45.3.2 Start rank and end rank . . . 502
45.3.3 Name of mixture compositions . . . 502
45.3.4 Translation . . . 502
45.3.5 Boundary conditions . . . 503
45.3.6 Language . . . 504
45.3.7 Export for the visualizer T4G . . . 504
45.3.8 Connectivity map . . . 505
45.3.9 Mesh (spherical inclusions only) . . . 505
45.4 Export input files for VtK . . . 507
45.4.1 Activation and target repository . . . 507
45.4.2 Start rank and end rank . . . 507
45.4.3 Visualization option . . . 507
45.4.4 Resolution of the spheres (spherical inclusions only) . . . 508
45.5 Statistical properties . . . 509
45.5.1 Activation and target repository . . . 509
45.5.2 Number of test lines . . . 510
45.5.3 Options for pre-defined grids . . . 510

**A EXAMPLE OF GEOMETRY** **511**
A.1 Introduction . . . 511
A.2 A hexagonal and parallelepipedic lattice, a lattice of lattices . . . 511
A.3 Homogenizer . . . 516
A.3.1 Defining the combinatorial geometry of a homogenizer . . . 516
A.3.2 Defining the surface-based geometry of a homogenizer . . . 518
A.4 Transport cask . . . 519
A.4.1 Defining the combinatorial geometry of a transport cask . . . 519
A.4.2 Defining the surface-based geometry of a transport cask . . . 521

**B DATA ENTRY SEQUENCE** **524**

**C TRANSLATION KEYWORDS** **551**

**D BIBLIOGRAPHY** **562**

**E INDEX** **567**

# TRIPOLI-4® Project Team

## (in alphabetical order)

CEA : François-Xavier HUGOT, Alexis JINAPHANH, Cédric JOUANNE, Coline LARMIER,
Yi-Kang LEE, Davide MANCUSI, Odile PETIT, Thierry VISONNEAU, Andrea ZOIA

# MAJOR CHANGES MADE TO THE CODE WITH RESPECT TO VERSION 11.1.

**Important warning**:

- The keywords used in the present manual belong to the **English dictionary**. However, the user must execute the code using the specific *english* option (as explained in Chapter 1 of this document) in order for the code to recognize these keywords. Otherwise, the code still uses the keywords given in the default dictionary (named *old*).

**Generalities** (see Chapter 1)

- The `-a` command-line option is now the default (all score values, even zeros) are printed.
- New command-line options `-z` and `-x`.

**Geometry** (see Chapters 7, 43 and 45)

- Matrices and origin change for surface transformations (see Subsection 7.5.16).
- MORET geometries can be read with minimal changes to data file (see Chapter 43).
- Generator of stochastic geometries (see Chapter 45).

**Sources** (see Chapter 12)

- New helpers for factorized source descriptions `UPTO_LIN` and `REPEAT` (see Subsubsection 12.2.2.3).

**Responses and scores** (see Chapters 13, 15 and 16)

- ZA distribution of fission products (see Section 13.1).
- Scores decomposed over real spherical harmonics, for needs of deterministic codes (see Chapter 16).
- Change of method for evaluating variances (see Section 15.4).

**Variance Reduction Techniques** (see Chapters 17, 19 and 20)

- Weight Windows (see Chapter 19).
- CADIS method (see Chapter 20).

**Simulation options** (see Chapter 22)

- Kinetic simulation mode (`DYNAMIC` option) (see Section 22.12).
- Wielandt method for criticality calculations (see Sections 22.1 and 22.18).
- Super-history method for criticality calculations (see Sections 22.1 and 22.17).
- Temperature stochastic interpolation available also with probability tables (see Section 22.1).
- Multiple mode (for subcritical or critical configurations) with AMS variance reduction technique (see Section 22.19).
- Inelastic threshold correction (see Section 22.1).

- New version 2.0.5 of the external LLNL fission library as fission model, to be used when requesting the new ZA distribution feature (see Section 22.1).
- K-th collision histories for partial contribution to the scores up to the k-th collision source (see Section 22.1).
- `SUPPRESS_MESH_EDITION_TEXT` option for mesh tallies edition (see Section 22.1).

**Depletion** (see Chapter 40)

- Allow normalization of power on a list of compositions or volumes and not only to the total.

**Perturbations and sensitivities** (see Chapters 29 and 30)

- Standard Perturbation Theory using history-based (Super-History or Wielandt) methods to reduce the memory occupation (see Chapter 29).
- Creation of default keywords to compute sensitivities for all nuclides for a list of reactions (see Section 29.2.1).
- Generalized Perturbation Theory for a reaction rate ratio using the Super-History simulation mode (see Chapter 30).

**Parallelism** (see Chapter 33)

- Cluster mode is now by default (mandatory use of mpirun or compatible launchers, see Section 33.2)
- Launcher to simplify launch in parallel (lparm tool, see Section 33.2)
- New command-line options `-S` (load balancing in parallel), `-E` (shorten simulator output files) and `-M` (tune the size of TCP packets in communications, for all options see Section 33.1)

**Graphical tools** (see Chapter 25 and 38)

- New T4G options mainly on meshes (see Chapter 38)
- New default in the collision sites feature: the maximum number of simulated particles for this graphical option is now 10000 (instead of 1000), unless the option for memory management `MAXP_FOR_COLLISION_SITES` is activated (see Section 25.3).

**Verification and validation** (see Chapter 1)

- Tool for verification and validation (see Section 1.6)

# Chapter 1

# GENERALITIES

## 1.1 GENERAL DESCRIPTION OF THE CODE

The TRIPOLI-4® code is a three-dimensional, continuous energy computer code for particle transport based on the Monte Carlo method. The code currently simulates the transport of neutrons, photons, electrons, and positrons [18, 44].

TRIPOLI-4® is designed for two major classes of problems, those relating to radiation shielding and those relating to reactor physics. Radiation shielding problems deal with particle propagation over long distances with many orders of magnitude of flux attenuation. These arise with sources in non-multiplying media, in steady-state and time-dependent conditions. Problems related to reactor physics refer to the behavior of particles in multiplying media. The system can be either critical or sub-critical, with or without fixed sources, under steady-state conditions.

The code builds on previous versions of TRIPOLI: TRIPOLI-1, TRIPOLI-2, and TRIPOLI-3 [88], [89], [1], [100], the development of which began in the late sixties and early seventies. It differs from earlier versions in that it has been completely rewritten in more recent programming languages (C and C++), with fuller descriptions of geometrical elements (surface-based and/or combinatorial geometries) and more precise representations of basic nuclear data (pointwise representation of cross-sections, although the code also allows multigroup homogenized cross-sections) [18].

For additional information, see also the website:
https://www.cea.fr/energies/tripoli-4

## 1.2 PURPOSE OF THIS DOCUMENT

This manual provides a full description of the code and describes the syntax of the TRIPOLI-4® code data.

## 1.3 EXECUTION

The code execution call (assuming that the executable is named `tripoli4`) is as follows (the options are to be entered on the same line as the `tripoli4`) command:

**tripoli4**
**-d** <**data_file**>
**-s** <**format**>
**-c** <**path_file**>
*-o* <*results_file*>
*-l* <*language*>
*-T notest*
*-x*
*-a*
*-u*
*-j* <*Njoy_code_path*>
*-k*
*-P* <*command*>
*-H*

The options in bold are required; those in italics are optional. Descriptions of these options are as follows:

- The option **-d** is used to specify the data file <**data_file**>. The syntax of the data file will be explicitly defined in the following chapters.
- The option **-s** is used to specify the format for the cross-sections. <**format**> equals:
  - **NJOY**, for a calculation with pointwise cross-sections from NJOY 99;
  - **TABPROB**, for a calculation with pointwise cross-sections from NJOY 99 for the resolved range, and probability tables for the unresolved resonance range from **CALENDF**.
  - **ENDL**, for a calculation with pointwise cross-sections from the EPDL libraries for high energy photons: up to 1 GeV (this is the option that must be chosen in order to simulate an electromagnetic shower without loading neutron data, for example);
  - **GENDF**, for a calculation with multigroup cross-sections generated by the GROUPR module of the NJOY code.
  - **TABPROB_GENDF**, for a calculation with multigroup cross-sections in **GENDF** format, and probability tables in **CALENDF** format.

  It should be noted that for a calculation with homogenized multigroup cross-sections (in APOTRIM format from the APOLLO2 code) [95] the choice of this option has no bearing on the calculation.
  Restrictions when using the **GENDF** and **TABPROB_GENDF** formats: neutron-only problem, calculation in a non-multiplying medium, natural simulation (without biasing), no neutron thermalization, no point-based estimators, no energy deposition calculations, no perturbation.
- The **-c** option is used to specify the file <**path_file**> containing the access paths for the nuclear data required by the code.
- The *-o* option is used to specify the file <**results_file**> where the code output is to be stored.
- The *-l* option is used to indicate the language used when entering the keywords. The possible values for <*language*> are "`english`," "`french`," and "`old`". The last option is a mixture of English and French words. It referred on the column *Default (old)* of Annex C. It used to be the only choice up to version 3 of TRIPOLI-4®. The line *-l* <*language*> is optional, and the default language is "`old`."
  This option can be replaced or overridden by the keyword `LANG` < *language* > in the very beginning of the data file, < *language* > being `FRENCH` or `ENGLISH`.

  Important Note : the keywords for this manual correspond to the option `english`. For other choices, see Annex C. Warning: some keywords in the `english` option have been modified with respect to version 4.3 of the code for reasons of clarity.

- The *-T notest* option disables the geometrical checks. This option can be useful when the checks are lengthy and the geometry has already been tested during an earlier calculation.
- The *-x* option can be used to disable the sorting of the scores by volume number during edition. By default, the scores edition is sorted by volume number.
- The *-a* option should be used if all of the results are required. In this case, even zero results will be displayed in the output. This option is useful for the post-processing of output files generated by TRIPOLI-4®. Starting from version 12 of TRIPOLI-4®, this option is set by default. The inverse option is *-z*, which prints only non-zero values.
- The *-u* option should be used to display results with physical units (see section 1.4)
- The *-j* option is used to couple the TRIPOLI-4® and NJOY 99 codes (generation of cross-section files at given temperatures, see the section 10.4). <Njoy_code_path> is the path for the NJOY executable. It could, for example, be 'which njoy_executable_name'. This option should exclusively be run in mono-processor mode. If any run in parallel should accidentally occur, the created cross-section files must be cleaned and the run should be executed again in mono-processor mode.
- The *-k* option enables compatibility with nuclear data libraries (namely .aniso, .xdr and .dictio files) processed by versions of the TRIPOLI-4® code earlier than 7 : the reading of cross-sections for new interactions implemented in the version 7 (see Notes on Table 13.1 of chapter 13) is disabled. **Warning**: starting with version 11, this option is deprecated and may be removed in a future release.
- The *-R* option disables the checks of which ROOT version is used.
- The *-P <command>* option enables the precompilation of the data file. *<command>* can simply be the name of the precompiler if this one is to be called without arguments or the complete command line with quotes.
  **WARNING !** If the *-P* option is to be used with cpp, it is reminded that, by default, the cpp precompilation will add '#' characters within the data file that will cause TRIPOLI-4® to crash. To avoid this behaviour, the *-P* option of cpp should be added within *<command>*.
  Examples of use of the *-P <command>* option of TRIPOLI-4® are given below:
  **-P m4**
  **-P'm4 -Dbatch=10'**
  **-P'cpp -P -Dbatch=10'**
- The help option *-H* is used to obtain a list of the execution options for the TRIPOLI-4® code.

## 1.4 DISPLAYING PHYSICAL UNITS

Option `-u` allows the user to display physical units. The displayed units are relevant for steady state calculations and are described in chapter 13. These units may not be correct for transient calculations.
This option must not be used if the user has requested a change of units in the input deck (see section 4.2).
In this case, the displaying of units will be disabled with a warning.
Displaying units will not turn on any special check on the sums. In cases where the user has requested a sum, the units of the first term of the sum will be displayed without verifying that the other terms are consistent.

## 1.5 VERSION NUMBER

The code now outputs the number of the version into the results file.
This information can be obtained by entering `tripoli4 -V` (or `tripoli4 --version`), where `tripoli4` is the name of the executable (with the path if necessary).

This manual briefly describes the types of cross-sections read by TRIPOLI-4®, explains how and why the dictionary file is used, and includes a description of the syntax for the calculation data definition file.

Once these three concepts are clear, particle transport calculations are relatively simple to implement. Further information concerning TRIPOLI-4® can be found in the following references: [10], [11], [14], [54], [68], [69], [70], [78], and [92].

## 1.6 VERIFICATION AND VALIDATION TOOL VALJEAN

Most of the results of TRIPOLI-4® calculations are provided in ASCII format. VALJEAN is an open-source Python library that can be used to parse the TRIPOLI-4® output into data structures suitable for post-processing and manipulation. VALJEAN provides tools to perform statistical tests on datasets and produce extensive and customizable reports.

The VALJEAN documentation and installation instructions can be found at https://valjean.readthedocs.io/.

The minimum VALJEAN version required for parsing outputs of TRIPOLI-4® version 12 is v0.10.

# Chapter 2

# CROSS-SECTIONS

## 2.1 NUCLEAR DATA FILE NAMES

The cross-sections used in the TRIPOLI-4® code simulations are pointwise cross-sections in ENDF format, or multigroup cross-sections in the GENDF format, or multigroup cross-sections calculated with the APOLLO2 code [95]. The name of an ENDF file containing simulation data is created as follows:
`lib.particle.isotope.format[.section_type(.therm).temperature]`
where

- `lib` can be:
  - **endfb6**;
  - **jef2**;
  - **jeff3**;
  - **fendl**.
- `particle` can be:
  - **neutron**;
  - **gamma**;
  - **electron**;
  - **positron**.

  Note: For multigroup cross-section in the GENDF format, only particles of type **neutron** are allowed.
- `isotope` can be any library isotope value;
- `format` can be:
  - **bcd** if it is an ASCII file;
  - **bin** if it is a binary file;
  - **xdr** if it is a binary file of the `xdr` type.
- `section_type` can be:
  - nothing for evaluation files;

  - pointwise cross-sections:
    - **pendf** for files containing pointwise cross-sections;
    - **aniso** for files containing anisotropy data.
  - multigroup cross-sections:
    - **gendf** for files containing multigroup cross-sections;
    - **mganiso** and **mgextra** for files containing anisotropy data.

- `temperature` can have any desired temperature value, provided that Doppler broadening by the NJOY system has been performed;
- `therm` is a suffix indicating whether the tape contains thermal data or not.

All of the terms between [ ] and ( ) are suffixes indicating the types of cross-sections contained in the file.
The other terms are mandatory.

## 2.2 OPTIMIZING CROSS-SECTION USE

To accelerate code execution, several options are available to access macroscopic cross-sections. Only neutron simulations without probability tables are concerned.

1. By default, for all materials including up to 25 isotopes, the total macroscopic cross-section is calculated during the initialization step and stored using an isotope-unionized energy grid. This speeds-up the running time during the simulation, at the price of a larger memory use. For materials with more than 25 isotopes, the macroscopic cross-section is calculated on the fly during the simulation.
2. If memory occupation is an issue, it is possible to disable this option. This is carried out using the following additional command with keyword `MEMORY`.
3. Conversely, if memory capacities are large enough, the storage of macroscopic cross-sections can be requested for all materials, whatever the number of isotopes. This is carried out using the following additional command with keyword `SPEED_UP`.

The syntax for the two last options is as follows (see section 4.3 for syntax conventions):

```
      OPTIMIZE
#                          MEMORY  1
#                          SPEED_UP  1
      END_OPTIMIZE
```

# Chapter 3

# DEFINING PATH AND DICTIONARY FILES

The elementary data used for particle transport are stored in data files. TRIPOLI-4® needs the location of these files to read the data.

## 3.1 THE PATH FILE

The path file gives the access paths for the following files:

- the dictionary file (see Section 3.2 below) describing the location of the photon and neutron cross-sections. This file should only be modified by an expert user
- a file giving the atomic weights of the nuclei
- a file giving the energies released by fissile isotopes during fission for the different particles emitted
- a directory giving all data necessary to electron and positron transport:
  - the tabulation of Mott cross-sections relative to the Rutherford cross-sections
  - electron interactions data (EEDL)
  - atom relaxation data (EADL)
- a file containing the isotopic abundances of natural elements
- a directory (with write-enable rights) where the files of cross-sections at given temperatures are to be written if they do not exist in the available libraries.

### 3.1.1 LEGACY FORMAT

An example path file follows:

```
/home/TRIPOLI-4/GALILEE-V0-2.1/CEAV5.1/ceav5.1.dictionary
/home/TRIPOLI-4/GALILEE-V0-2.1/Common_Data/mass_rmd.mas95
/home/TRIPOLI-4/GALILEE-V0-2.1/CEAV5/Qfission
/home/TRIPOLI-4/GALILEE-V0-2.1/ENDL97/ELECTRON_DATA
/home/TRIPOLI-4/GALILEE-V0-2.1/Common_Data/abundance
```

### 3.1.2 NEW FORMAT

Starting from version 11, TRIPOLI-4® accepts a new path file format, which is structured as follows:

```
t4path 0
njoy_dictionary     /home/TRIPOLI-4/GALILEE-V0-2.1/CEAV5.1/ceav5.1.dictionary
mass_data           /home/TRIPOLI-4/GALILEE-V0-2.1/Common_Data/mass_rmd.mas95
Q_fission           /home/TRIPOLI-4/GALILEE-V0-2.1/CEAV5/Qfission
electron_xs         /home/TRIPOLI-4/GALILEE-V0-2.1/ENDL97/ELECTRON_DATA
abundance_data      /home/TRIPOLI-4/GALILEE-V0-2.1/Common_Data/abundance
```

The integer following the `t4path` string is a version identifier, reserved for future compatibility. The rest of the file consists of key-value pairs, which may be given in any order.
Some keys are optional:

- the `own_eval` key can be used to specify the location of the directory where TRIPOLI-4 will write the nuclear data libraries generated on the fly by Njoy (see Section 10.4);
- the `electron_bremsstrahlung_xs` key can be used to specify the location of data tables for Seltzer and Berger's electron bremsstrahlung model [96] (see Section 22.14).

Starting from version 12, the following file format is also accepted:

```
t4path 1
njoy_dictionary             /(...)GALILEE/GALILEE-V0-3.0/CEAV512/ceav512.dictionary
mass_data                   /(...)GALILEE/GALILEE-V0-3.0/Standard_data/mass_rmd.mas95
Q_fission                   /(...)GALILEE/GALILEE-V0-3.0/CEAV512/Qfission
electron_xs                 /(...)GALILEE/GALILEE-V0-3.0/PEID/Electron_Photon
abundance_data              /(...)GALILEE/GALILEE-V0-3.0/Standard_data/abundance
electron_bremsstrahlung_xs  /(...)tripoli4.<X>.<Y>/AdditionalData/Bremsstrahlung
idt_photon_xs_file          /(...)tripoli4.<X>.<Y>/AdditionalData/apolib/Gamma/ap3(...)3.1.8_Photo.h5
idt_neutron_xs_file         /(...)tripoli4.<X>.<Y>/AdditionalData/apolib/Neutron/ap3(...)_N-a3ver1.h5
```

The `idt_photon_xs_file` and `idt_neutron_xs_file` keywords specify the default location of the HDF5 cross section files for the IDT solver (see Chapter 20).
The AdditionnalData directory comes with the code and not with a nuclear data library. When installing the code, if the reference nuclear data is already installed, as required by the automatic installation tests, the code installer will automatically migrate the t4path file available in the nuclear data library (e.g.: /(...)/GALILEE/GALILEE-V0-3.0/CEAV512/t4path.ceav512) into a new t4path file using the new path file format in the Env directory (e.g. /(...)tripoli4.<X>.<Y>/Env/t4path.ceav512). In particular it will feed the last three lines about the `electron_bremsstrahlung_xs`, `idt_photon_xs_file`, `idt_neutron_xs_file` keywords. In case the user wants to use another nuclear data library, the following tool can be used to help with this task (/(...)tripoli4.<X>.<Y>/CODE/<OS>/tools/Pathfile/migrate-t4path).

## 3.2 THE DICTIONARY FILE

The dictionary, or catalogue, file is required for a proper execution of a calculation performed using the TRIPOLI-4® code.
A dictionary contains:

- the path to a directory containing all the cross-section files referred to in the previous chapter;
- the names of the isotopes used to define the material compositions used in the calculation and all the NJOY codes associated with the isotopes (see specification of material compositions).
- Starting from TRIPOLI-4® version 8, specifications on the calculations of Q-values (see later on).

The dictionary is installed with the tapes and code. Its use should be transparent to the user. However, it can be helpful to understand the principles involved when connecting libraries or creating new ones.
The first item in the dictionary file is the path to the directory containing the nuclear data files necessary for the simulation. The code automatically seeks the evaluation files, the pointwise or multigroup cross-section files and the anisotropy files in the directory (all of which must be present).
For dictionary files to be used with TRIPOLI-4® versions up to 7 (included), the second item in the file is the number of isotopes contained in the directory. Starting from TRIPOLI-4® version 8, the second item in the file is the number of isotopes contained in the directory increased by 10000. For instance, if 25 isotopes are present, this item will be 10025.
The following data are present for each isotope:

- the name of the isotope as given in the TRIPOLI-4® code data file (any name is possible and must only be identical to that of the isotope as used in the composition specifications);
- the material number for neutron cross-sections (which is assigned to each isotope by NJOY): **NJOY(neutron)**;
- the material number for gamma cross-sections (which is assigned to each isotope by NJOY): **NJOY(gamma)**;
- the material number for thermal cross-sections. The material number for a thermal cross-section is a combination of NJOY material numbers. As for the neutron cross-sections, each isotope with thermalization data has a thermal material number assigned by NJOY: **NJOY(therm)**. The number to be given in the dictionary is, therefore, coded as follows: **T4(therm)** = **NJOY(neutron)** $\times$ 10000 + **NJOY(therm)**;
- (only for dictionary files intended to be used with TRIPOLI-4® versions starting from 8) a flag by which the user specifies whether reaction Q-values are to be computed from mass defect (based on the atomic weights file: flag= 0; this is the value suggested for most isotopes in the dictionary, and coincides with older versions of TRIPOLI-4®), or read from nuclear evaluation files. This flag affects the calculations of energy deposition, gas production, DPA and kinematic limits. See section 13.7 and in particular section 13.7.1 for the values of this flag. The flag values chosen by default should only be modified by an expert user. When the item defining the number of isotopes in the directory is not augmented by 10000, this column of the dictionary file is ignored (to preserve compatibility with older versions of TRIPOLI-4®). This means that in this case the Q-values are by default computed from mass defect.
- the name of the file containing the evaluation in the directory.

For the pointwise case, in the absence of thermal or gamma data, or if these types of data are not used in the calculation (in which case they do not have to be read), these material numbers must be set to -1. In the multigroup case, these numbers must always be set to -1.

Note: for certain materials, the isotopes required are special isotopes (as a result of different chemical bonds). A frequent case is that of water, for which a specific hydrogen must be used: H_H2O (or H1_H2O, depending on the name used in the dictionary).

It is, therefore, necessary to use:

- hydrogen H_H2O (or H1_H2O) for water;
- hydrogen H_CH2 (or H1_CH2) in certain polymers (polyethylene for instance);
- deuterium D_D2O for heavy water;
- carbon C_G (or C_GRAPHITE) for graphite.

The dictionary file syntax is, therefore, as follows (TRIPOLI-4® versions up to 7 included):

```
path
number of isotopes
NAME(isotope_1) NJOY(neutron) NJOY(gamma) T4(therm) name(file_1)
...
NAME(isotope_n) NJOY(neutron) NJOY(gamma) T4(therm) name(file_n)
```

where `name(file_i)` is the name of the evaluation file in ASCII form.
Starting from TRIPOLI-4® version 8, the dictionary file syntax is instead as follows:

```
path
number of isotopes + 10000
NAME(isotope_1) NJOY(neutron) NJOY(gamma) T4(therm) FLAG(Q-value) name(file_1)
...
NAME(isotope_n) NJOY(neutron) NJOY(gamma) T4(therm) FLAG(Q-value) name(file_n)
```

**Dictionary of the criticality calculation example of chapter 5, with TRIPOLI-4® version 8**

```
/home/TRIPOLI-4/GALILEE-V0-2.1/JEFF311/T4XS/
10025
FE54 2625 -1 -1 0 jeff311.neutron.Fe54.bcd
FE56 2631 -1 -1 0 jeff311.neutron.Fe56.bcd
FE57 2634 -1 -1 0 jeff311.neutron.Fe57.bcd
FE58 2637 -1 -1 0 jeff311.neutron.Fe58.bcd
CR50 2425 -1 -1 0 jeff311.neutron.Cr50.bcd
CR52 2431 -1 -1 0 jeff311.neutron.Cr53.bcd
CR53 2434 -1 -1 0 jeff311.neutron.Cr53.bcd
CR54 2437 -1 -1 0 jeff311.neutron.Cr54.bcd
NI58 2825 -1 -1 0 jeff311.neutron.Ni58.bcd
NI60 2831 -1 -1 0 jeff311.neutron.Ni60.bcd
NI61 2834 -1 -1 0 jeff311.neutron.Ni61.bcd
NI62 2837 -1 -1 0 jeff311.neutron.Ni62.bcd
NI64 2843 -1 -1 0 jeff311.neutron.Ni64.bcd
N14  725  -1 -1 0 jeff311.neutron.N14.bcd
O16  825  -1 -1 0 jeff311.neutron.O16.bcd
U234 9225 -1 -1 0 jeff311.neutron.U234.bcd
U235 9228 -1 -1 0 jeff311.neutron.U235.bcd
```

```
U236 9231 -1 -1 0 jeff311.neutron.U236.bcd
U238 9237 -1 -1 0 jeff311.neutron.U238.bcd
H1   125  -1 -1 0 jeff311.neutron.H1.bcd
AL27 1325 -1 -1 0 jeff311.neutron.Al27.bcd
SI   1400 -1 -1 0 jeff311.neutron.Si.bcd
CA   2000 -1 -1 0 jeff311.neutron.Ca.bcd
BE9  425  -1 -1 2 jeff311.neutron.Be9.bcd
NA23 1125 -1 -1 0 jeff311.neutron.Na23.bcd
```

Here, `/home/TRIPOLI-4/GALILEE-V0-2.1/JEFF311/T4XS` is the directory where the cross-sections are located, and 25 is the number of isotopes (plus 10000), each line giving the name, neutron number, photon number (here set to -1 as there are no photon data), thermal number (here set to -1 as there are no thermal data), Q-value flag, and the name of the evaluation file (neutron) for each of these isotopes. A Q-value flag 0 means that the reaction Q-values are computed based on the mass defect, a Q-value flag 1 means that the reaction Q-values are read in the evaluation file, and a Q-value flag 2 means that the reaction Q-values are read in the evaluation file and that special kinematics are applied for the isotope (see sections 13.8, 13.9 and 13.10).

With the above dictionary, the links are only created for neutron cross-sections since the photon and thermalization cross-sections are not loaded (indices set to -1).

### Mixed evaluations

If cross-sections found in two directories are to be used (for example, in two evaluations), the root may be changed and the path included in the file name:

```
/home/TRIPOLI-4/GALILEE-V0-2.1/
10006
FE54 2625 -1 -1 0 JEFF311/T4XS/jeff311.neutron.Fe54.bcd
FE56 2631 -1 -1 0 JEFF311/T4XS/jeff311.neutron.Fe56.bcd
FE57 2634 -1 -1 0 JEFF311/T4XS/jeff311.neutron.Fe57.bcd
FE58 2637 -1 -1 0 JEF2/T4XS/jef2.neutron.Fe58.bcd
CR50 2425 -1 -1 0 JEF2/T4XS/jef2.neutron.Cr50.bcd
BE9  425  -1 -1 2 JEFF311/T4XS/jeff311.neutron.Be9.bcd
```

### Using dosimetry or activation evaluations

Dosimetry and activation evaluations provide only neutron cross-sections data but neither angular nor energy distributions data. These evaluations can thus be used only to request reaction rates on virtual nuclei which are not part of the transport simulation (see section 13.2). Starting from TRIPOLI-4® version 8, the user can request reaction rates from dosimetry evaluations (such as International Reactor Dosimetry Files 2002) or activation evaluations (such as European Activation File 2003), without any change in the dictionary file (as was the case with previous versions). The corresponding names of the isotopes (which can be found in the dictionary) are ended by `_D` (e.g `FE54_D`) for the `IRDF2002` case and by `_A` (e.g `FE54_A`) for the `EAF2003` case. Moreover, when cross-section data are only present as production of radioactive elements data in these evaluations, TRIPOLI-4®, starting from version 8, reads and handles them as standard cross-sections.

# Chapter 4

# GENERAL CONSIDERATIONS : DATA, CONVENTIONS, UNITS, FRAMES

In this chapter, we give the conventions for writing the data file and describe its general structure. This file consists of a number of modular commands, each indicating an instruction.

First of all, it must be pointed out that the input data is in free format: the number of white spaces after the first is not meaningful. It is therefore possible to indent the data file in order to make it more legible. In addition, empty lines can be inserted anywhere.

The keywords must be entered in UPPERCASE.

## 4.1 THE DEFAULT UNITS

The units in the data file are as follows:

- Length: **centimeters**;
- Time: **seconds**;
- Temperature: **Kelvin** (note: the temperature read from an APOLLO2 file for multigroup cross-sections is in degrees Celsius);
- Energy: **MeV** (important: data on ENDF tapes are expressed in eV);
- Angles: **degrees** for the definition of the cone surface and shape, and of rotation volumes, **radians** for the definition of sources and frames;
- The cross-sections entered by the user (DATA option) are in **barns** ($10^{-24}$ cm$^2$);
- The number of atoms in the compositions are given in $10^{-24}$ **cm**$^3$;
- The calculated densities are in **g/cm**$^3$;
- The unit for particle sources is the number of particles (this value is integrated in volume, directions, energy, and time).

## 4.2 UNIT CHANGES

It is possible to change some units (length and energy). Care is required when changing the unit of length, as the cross-sections remain in barns, the number of atoms for $10^{-24}$ cm$^3$, and the densities in g/cm$^3$. **Beware that it also affects the display of results.** For example, the fluxes are by default in length units so a change of length unit will have a direct change on fluxes.

Changes are made as follows:

**UNITS**
*LENGTH*
*x* <*UNIT*>
*ENERGY*
*y* <*UNIT*>
**END_UNITS**

where

- `x` is a decimal value;
- `<UNIT>`: is the name of the unit. For length, the choices are {`mm, cm, dm, m, dam, hm, km`}, and for energy the choices are {eV, keV, MeV, GeV}. Names of units in uppercase only can be used.

Thus, `2.54 cm` means that the unit of length is 1 inch.

## 4.3 CONVENTIONS USED IN THIS MANUAL

The basic syntax of a command (or sets of commands) is given in a bold framebox.
The basic syntax of a sub-command is shown in a double frame.
The basic syntax of a lower level command is shown is in a single frame.
Examples are given on the next page.

```
GEOMETRY
        TITLE  title
*               TYPE  num(type)
                    Definition for type num(type)
                ...
*               SURF  num(surf)
                    Definition for surface num(surf)
                ...
                VOLU  num(volu)
                    Definition for volume num(volu)
                ENDV
                ...
                POSITION  x y z
                COLOR
                    Colormap
                GRAPH
                    Data on 2D-cut
ENDGEOM
```

```
TYPE
        num(type)
        <SHAPE>
        Descriptive data
```

```
BOX  hx hy hz
```

## 4.4 CHARACTER STYLE CONVENTIONS

This convention applies to frameboxes used in describing a syntax. In ordinary text, entries of code are given in `typewriter-style characters` (font: `Courier`).
Letters are:

- In **bold** and
  - in UPPERCASE for keywords. e.g.,
    **GEOMETRY**
    Exception: apart from the keywords **X**, **Y**, **Z**, **E**, and **T**, used for specifying sources or the `GRAPH` sub-command of `VARIANCE_REDUCTION`, a single letter in uppercase corresponds to the coordinate of a point (to be replaced by its value).
  - in UPPERCASE and between < >: to be replaced by a keyword.
    E.g. <**SHAPE**> is to be replaced by **BOX** , **SPHERE** , **CYLX** ...
  - in lowercase: an item of data, e.g.
    **numXXX** to be replaced by number of XXX
    **nbXXX** to be replaced by number of XXX
- Non-bold for definitions related to data whose structure is given elsewhere in the manual, e.g.
  <**SHAPE**> Definition
  After the keyword has been selected from the available range, the characteristics of the shape must be entered. The syntax is given, below (for instance, in the frame defining `BOX`).
- *In italics* for non-mandatory lines.

If an asterisk is placed to the left of the line, it means that the line itself is not mandatory but a choice must be made between it and the other asterisked lines included in the data set.
If several sets of choice are placed on consecutive lines, the asterisks are subscripted by an integer in order to indicate the extent of each set.
E.g.

* ***TYPE numtype***
* ***SURF numsurf***

it is possible to define a geometry without calling `TYPE` or without calling SURF, but one of the two must be called.

If there is a hash mark at the beginning of the line, it means that the line is not mandatory and is also incompatible with the other lines beginning with a «#». e.g.

\# **CRITICALITY**
\# **FIXED_SOURCES_CRITICALITY**

it is not possible to carry out a calculation designated as both a criticality calculation and a sub-criticality calculation with sources.

## 4.5 DEFINING FRAMES

The frame definition is introduced using the keyword FRAME followed by :

- a frame type.
- an origin $O'$;
- three vectors defining an orthonormal frame: **i**, **j**, **k**; The **j** axis is only specified in the case of Cartesian coordinates

All these data are given in the cartesian reference frame (X,Y,Z), see figures 4.1, 4.2 and 4.3.
The syntax is as follows :

**FRAME**
<**FRAME_TYPE**>
**O'**$_x$ **O'**$_y$ **O'**$_z$
**i**$_x$ **i**$_y$ **i**$_z$
(**j**$_x$ **j**$_y$ **j**$_z$) if cartesian
**k**$_x$ **k**$_y$ **k**$_z$

The frame type can be **Cartesian**: a vector **O'M** is defined by its coordinates $(x,y,z)$, projections of **O'M** along axes **i**, **j**, **k** (see Figure 4.1) :

$$\mathbf{O'M} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}.$$

It can also be **cylindrical**, in which case the vector **O'M** is defined by its coordinates $(r,\theta,z)$ where $r$ is the normal of the projection **O'M** on the plane (**i**, **j**), $\theta$ is the angle between the projection (between 0 and $2\pi$) and the vector **i**, and $z$ is the projection of **O'M** along axis **k** (see Figure 4.2) :

$$\mathbf{O'M} = r\cos\theta\mathbf{i} + r\sin\theta\mathbf{j} + z\mathbf{k}.$$

Finally, the frame type can be **spherical**, in which case the vector **O'M** is defined by its coordinates $(r,\phi,\theta)$ where $r$ is the normal of **O'M**, $\phi$ is the angle (between 0 and $2\pi$) between the projection of **O'M** on the plane (**i**, **j**) and the vector **i**, and $\theta$ is the angle between OM and the axis (**i**, **k**) (between 0 and $\pi$) (see Figure 4.3) :

$$\mathbf{O'M} = r\cos\phi\sin\theta\mathbf{i} + r\sin\phi\sin\theta\mathbf{j} + r\cos\theta\mathbf{k}.$$

Note: when a non-unit vector is entered to specify the reference frame, TRIPOLI-4® automatically renormalizes it to 1.

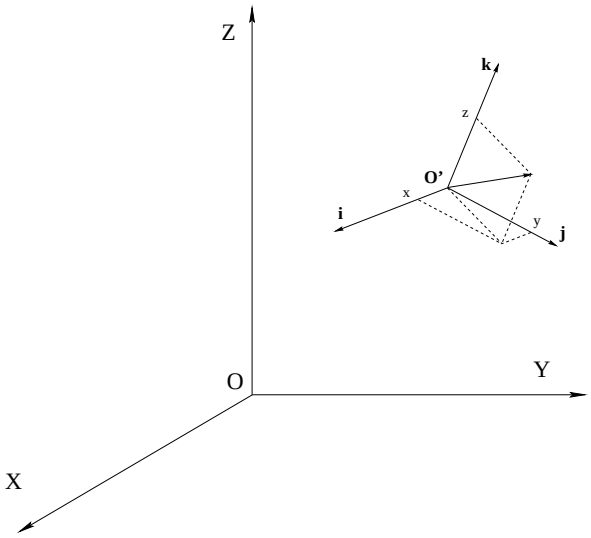



Figure 4.1: Cartesian coordinates

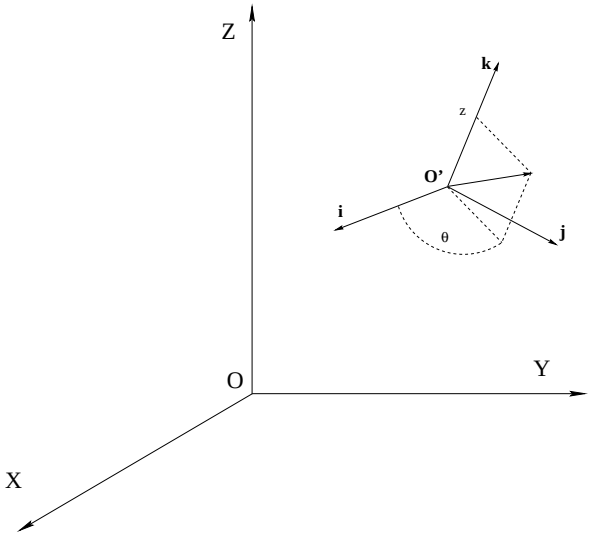



Figure 4.2: Cylindrical coordinates

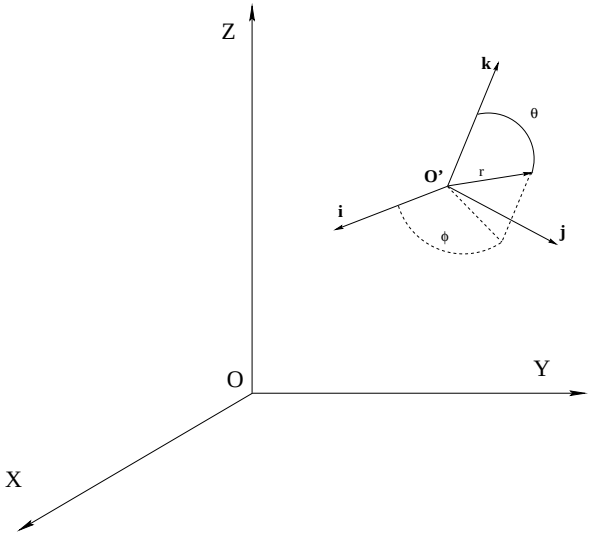



Figure 4.3: Spherical coordinates

## 4.6 LIST LAYOUT

A list of items or a list of sets of items is written as follows:

**item(1) ... item(n)**

or

**set_of_items(1)**
...
**set_of_items(n)**

or

**set_of_items**
...

## 4.7 LIST OF CALCULATION COMMANDS

A definition of the TRIPOLI-4 data file is now given.
TRIPOLI4 calculation input data consist of the following commands:

- geometry definition.
  **GEOMETRY ... ENDGEOM**
  This section defines the geometrical items used in the calculation.
- values of areas and volumes
  **VOLSURF ... END_VOLSURF**
  This section provides volume and area values.
- boundary condition definition.
  **BOUNDARY_CONDITION ... END_BOUNDARY_CONDITION**
  This section defines calculation boundary conditions.
- medium and composition definition
  **COMPOSITION ... END_COMPOSITION**
  This section sets calculation media isotopic compositions.
- association of volumes and compositions
  **GEOMCOMP ... END_GEOMCOMP**
  This section is used to associate each geometry volume with the isotopes composing it. The geometry and compositions must be defined beforehand.
- source definition
  **SOURCES_LIST ... END_SOURCES_LIST**
  This phase defines the emission densities of the source particles used in the calculation.
- response definition
  **RESPONSES ... END_RESPONSES**
  The responses are physical quantities calculated by the code (flux, current, reaction rates etc.).
- grid definition
  **GRID_LIST ... END_GRID_LIST**
  Grids correspond to time, angle, or energy intervals in the simulation domain for which the responses are calculated. They include homogenization grid.

- score definition
  **SCORE ... END_SCORE**
  A score corresponds to a combination including an estimator, grids, and geometrical units determining the calculation of a response function. The geometry, composition, division, and response definitions must be completed beforehand.
- perturbation definition
  **PERTURBATION_LIST ... END_PERTURBATION_LIST**
- biasing definition
  **VARIANCE_REDUCTION ... END_VARIANCE_REDUCTION**
  Biasing relates to the use of variance reduction techniques in the simulation. The geometry and composition definitions must have been completed beforehand.
- simulation definition
  **SIMULATION ... END_SIMULATION**
  This section deals with the simulation itself. It sets the simulation parameters.
- announcing the use of collision tapes
  **COLLISION_TAPE ... END_COLLISION_TAPE**
  This command enables dumping to a listing file events undergone by particles or, conversely, re-using such a file to obtain new scores.
- announcing the use of Green's function files
  **GREEN_FUNCTION_TAPE ... END_GREEN_FUNCTION_TAPE**
  This command enables storing source information and tallying information for particles contributing to the score, and then allows the re-use of this information. Possible uses and applications include parametric studies with source modification, or calculations of scores from different source regions.
- announcing homogenization
  **HOMOGENIZE ... END_HOMOGENIZE**
  This section sets the compositions to be homogenized and the corresponding volumes.
- verification calculation definition
  **CALCULATION ... END_CALCULATION**
  This section sets the volume and density of the compositions to be calculated.
- special output definition
  **OUTPUT ... END_OUTPUT**
  This section sets special outputs (XVGR graphs).
- particles to be stored
  **STORAGE ... END_STORAGE**
  This section specifies the contingent storage of particle data in files, for subsequent processing (SN code etc.).
- assigning data for subsequent depletion calculations
  **PEPIN ... END_PEPIN**
  This section can be used to create files for DARWIN-PEPIN2.
- ROOT coupling
  **ROOT_GEOMETRY ... END_ROOT_GEOM**
  **ROOT_LIMIT ... END_ROOT_LIMIT**
  **TRACKS ... END_TRACKS**
  These sections can be used to define a ROOT geometry and ROOT boundary conditions, and to record particle tracks for ROOT post-processing.

The following definitions are mandatory:

- geometry;
- compositions;
- associations of volumes and compositions;
- sources;
- simulation criteria.

For a radiation shielding calculation, scores must be demanded by specifying the following items:

- grids (energy, space, angle, and time discretizations);
- responses;
- scores.

The following settings are optional:

- boundary conditions (leakage is the default);
- biasing;
- homogenization request (requiring an energy grid);
- perturbations;
- collision bands;
- verification calculations;
- DARWIN-PEPIN2 and ROOT couplings.

The dependences between definitions are symbolized by arrows in the following (simplified) diagram (an arrow from A to B means that A must be set before B).

The following pages show how a file is composed (options in dotted lines).

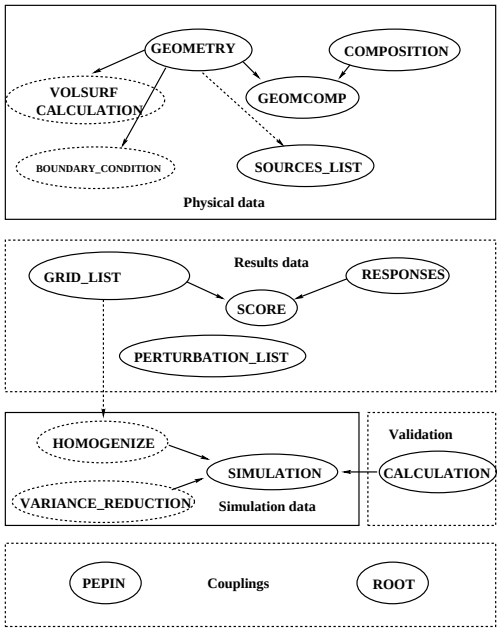



Figure 4.4: Commands

# Chapter 5

# EXAMPLE

After each of the different datablocks have been defined (geometrical data, composition data, etc.), the corresponding data for a test example are given (the dictionary has already been established for this example).

The geometrical configuration of the example (an experiment using cylinders containing a solution of uranyl nitrate), consists of:

- a concrete slab 300 x 300 x 60 $cm^3$;
- two cylindrical containers with conical bottoms (for simplicity, the conical bottoms are not modeled) made of stainless steel and placed vertically 125 cm above the concrete slab and containing:
  - uranyl nitrate solution at the bottom;
  - air at the top.

The geometry is shown on the following page.
The data are given for a critical height $H_c$=28.54 cm and a distance of $D$=0.5 cm between the containers.

The syntax is given for the following two sources:

- a point source at the center of the uranyl nitrate solution in the left-hand container (as an example of a point source, whereas convergence would be faster with a diffuse source corresponding to the fissile solution);
- a source corresponding to the uranyl nitrate solution in the right-hand container.

We will show the user how to request:

- the flux, computed using the track length estimator, in the air volumes above the uranyl nitrate solution (volumes 4 and 7);
- surface flux between the air volume (1) and the concrete (2);
- a macroscopic reaction rate for isotope U238, computed using the track length estimator, in the uranyl nitrate volumes (volumes 5 and 8);
- a macroscopic reaction rate for isotope O16, computed using the track length estimator, in the air volumes (volumes 1, 4 and 7);
- a tabulated reaction rate, computed using the track length estimator, in all the volumes.

The calculation is a criticality calculation.

Other geometry examples are given in the appendix. These examples seek to cover the different possible data definitions in TRIPOLI-4®.

Complete examples of benchmark cases will also be given in the appendix in later versions of this report.

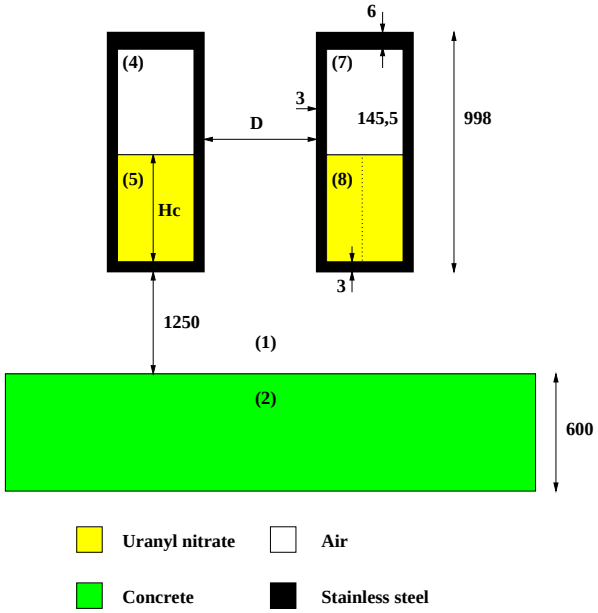



Figure 5.1: Geometrical configuration for the interaction between two uranyl nitrate containers in air

# Chapter 6

# USING COMMENTS AND INSERTING FILES

## 6.1 COMMENTS

In the TRIPOLI-4® code, there are now three syntaxes for comments.
Comments can be placed inside any command, subject to the following limitations.

### 6.1.1 COMMENTS USING COMMENT

Comments can be inserted using the old syntax:

**COMMENT**
Comment
**COMMENT**

It is not possible to have nested comments using this type of comment alone.

### 6.1.2 C TYPE COMMENTS

Comments can also be added by placing them between /* and */. This new syntax makes it possible to nest comments within a comment.
Example:

```
/* First comment
 * -------------------
      /*
       * Nested comment
       */
   End of first comment
 */
```

This can be used to comment a part of a set of data containing comments (this part can include all possible types of comments).

### 6.1.3 C++ TYPE COMMENTS

Finally, it is possible to insert a line of characters preceded by //. This type of comment can be added at the end of a line, the rest of which is then ignored in the execution of the code.
Example:

```
DATA    // Comment
```

### 6.1.4 RESTRICTIONS

Whatever syntax is used, the comment must **precede** a keyword (or name of a composition, grid etc.). This is because it is only at the time when the keywords are decoded that the comments are eliminated. In a geometry command, the comments must be given after the title.
If the keyword KEYWORD is to be followed by a number (for instance the volume), the syntax

```
// Read expected keyword
   KEYWORD 1
```

is correct, whereas

```
   KEYWORD // Read number expected
   1
```

is not.

## 6.2 INSERTING FILES

It is possible to insert one (or more) commands using:

**FILE**
**name(file)**

where name(file) is the name of the file (and possibly its path, e.g. /home/user/T4/Data) that contains the part of the data-set to be inserted.
It is not possible to insert part of a command.
It is thus possible to load a geometry derived from another code (in a suitable format), a list of compositions derived from PEPIN, etc.
It is worth noting that, when using the keyword FILE, only the line of insertion will be copied in the output file, which can be a solution of memory sparing in case of numerous lines (e.g. for the geometry definition).

# Chapter 7

# GEOMETRY DEFINITIONS

A geometry command is introduced by the keyword GEOMETRY and ends with the keyword ENDGEOM. Volume definitions corresponding to the given geometry are placed between these two keywords.

**IMPORTANT:** The whole geometry must be convex. If the geometry is not convex, some particles may prematurely be lost and some scores may be underestimated. *Note that, by default, there is no check on convexity or connectedness of the geometry*. This may be performed using the CHECK_GEOM option (see 7.13.2).

Boundary conditions are defined after the geometry definition, in the BOUNDARY_CONDITION command.

For complex cases (lattices etc.), it may be necessary to review this section twice in order to clearly understand the cross-references.

**Recommendations:**

The particularities of the geometrical operators need to be taken into account:

- a lattice cannot be rotated : it must be aligned with the axis of the reference frame.

The particularities of the other commands are important (detailed below):

- BOUNDARY_CONDITION: the boundary conditions are given on a side or a surface of a volume. Important: Only the sides associated with the TYPE (combinatorial geometry) or surfaces SURF (surface-based geometry) explicitly included in the definition can identify a surface associated with the volume. In particular, when a first volume is modified by an operator (SMASH, VMINUS, INTE and UNION) and a second volume (as argument), the first volume will not know the sides or surfaces of the second after the operation;
- SCORE: a surface-based score is defined at the interface between two volumes and not at the side or surface of a volume;
- SOURCES_LIST: for a point source, the angular distribution is defined with respect to a spherical reference frame whose $Z$ axis is the $Z$ axis of the global geometry reference frame;

The choice between a combinatorial or a surface-based geometry is not without consequences in the **running time** of the simulation. Some advice concerning the optimal choices that can be made are given at the end of this section.

A number of error diagnostics are implemented in combinatorial geometries to avoid errors in the definitions, but it is **indispensable**, in complex cases, to verify by **graphical display** that the geometry given to the code corresponds to the user intentions. It should be noted that geometry errors can also be detected if automatic biasing takes place (see, below). It is therefore sometimes advisable to launch a dry run with biasing, just to verify the geometry, with no connection to the actual calculation.
In this section, we consider the following points in detail:

- Different ways of defining volumes
  Volumes can be defined in five different ways: combinatorial, surface-based, through repetition of simple volumes, through replacement of cells of a lattice, or through rotation of simple volumes;
- Defining data for combinatorial geometries
  The shape data, introduced by the keyword `TYPE`, is the basic component of a geometrical definition in combinatorial form;
- Surface-Based data definition
  Surface-based data, introduced by the keyword `SURF`, is the basic component of a geometrical definition in surface-based form;
- Simple volume definition.
  Volume definitions, introduced by the keyword `VOLU`, on the basis of types (combinatorial) or surfaces (surface-based);
- Rotation-type volume definition;
- Definition of different operators acting on volumes;
- Definition of lattice-type volumes;
- Definition of cell or sub-cell type volumes;
- Syntax of a display request (2D-cross section view).

## 7.1 GEOMETRICAL ACCURACY

Note: Lengths are expressed in centimeters and angles are expressed in degrees in the geometry.

The accuracy used in calculating distances is $\varepsilon_{geo} = 10^{-4}$ cm. In order to find out which volume will be entered next, the code temporarily advances the particle position by a value of $\varepsilon_{geo}$ cm along the particle line of flight.

It is, therefore, pointless to attempt to create a volume with a length less than this threshold, particularly in an attempt to impose a boundary condition.
Precision is, however, required for entering coefficients that define the volume.

## 7.2 GLOBAL STRUCTURE

The whole geometry is defined in a Cartesian reference frame of origin *O* and axes *OX*, *OY* and *OZ*.

The structure command is as follows:

```
GEOMETRY
        TITLE  title
*               TYPE  num(type)
                    Definition for type num(type)
                ...
*               SURF  num(surf)
                    Definition for surface num(surf)
                ...
                VOLU  num(volu)
                    Definition for volume num(volu)
                ENDV
                ...
                POSITION  x y z
                COLOR
                    Colormap
                GRAPH
                    Data on 2D-cut
ENDGEOM
```

where:

- `title` is any chain of characters (a maximum of 256);
- as many `TYPE`, `SURF`, and `VOLU` as desired;
- `VOLU` defined for a `TYPE` or a `SURF` that already exists;
- any numbering for `num(type)`, `num(surf)`, and `num(volu)`, but `num(volu)` must not begin with a "0";
- assignment of colors to volumes (these assignments are used for 2D-cross sectional views declared after the association is made). It is possible to change the colors of the different views. In such cases, it is necessary to make as many `COLOR` commands as required (see 7.12.1);
- one (or more) 2D-views, i.e., geometrical 2D-views.

It is possible to request the position of a point $(x,y,z)$ using `POSITION x y z`: in this case, the code returns the volume where the point is located at the time the request is made (which is not necessarily at the end of construction. The current position may also be requested). The code states the cell in the case of a lattice, and the number of the side if the point is on a numbered surface (for boundary conditions, see the corresponding section).

## 7.3 DIFFERENT WAYS TO DEFINE VOLUMES

A TRIPOLI-4® geometry corresponds to an ensemble of volumes.
In this version, there are five possible kinds of volumes, the first two correspond to "**simple**" volumes, the third is used to create volumes by rotation and the last two relate to the possibilities available with lattices.
It is possible to mix combinatorial and surface-based geometries. A volume created from surfaces may also be introduced into the cell of a lattice (but the cell used to create the lattice must be `COMBI`, `RESC`, or `RESH`).

### 7.3.1 COMBINATORIAL GEOMETRY

A first option consists of creating volumes using predetermined geometrical shapes. These are introduced by the keyword `COMBI`. The `TYPE` for these volumes must have been previously declared in the data. The type corresponds to a predetermined geometrical shape (sphere, cylinder, box, etc.) plus descriptive data (for instance, for a sphere, a type is obtained by setting its radius).

### 7.3.2 SURFACE-BASED GEOMETRY

A second possibility is to specify the volumes using equations. They are introduced by the keyword `EQUA`. These volumes are defined in terms of equations of surfaces (`SURF`), which must have been previously declared, setting the boundary of the volume and the position of the volume with respect to the surface (the process used for TRIPOLI-2 and TRIPOLI-3.)

Geometry error diagnostics are not available in this type of definition (**no overlap tests done by default**).

### 7.3.3 DEFINITION USING A ROTATION

There is a third possibility, consisting of creating volumes using a rotation operator with a predetermined volume. These volumes are introduced by the keyword `ROTATION`.

### 7.3.4 DEFINING A LATTICE

The final method consists of repeating a pre-existing volume in order to create a lattice. This volume thus constitutes the reference cell. Only orthorhombic lattices (i.e. parallelepipedic) and hexagonal lattices are implemented. The reference volume has:

- either a parallelepipedic shape (`BOX` or `PARALLELEPIPEDE`) repeated using the `RESC` operator along the three axes of the parallelepiped;
- or one of the hexagonal shapes `HEXAX`, `HEXAY`, or `HEXAZ`: in which case it is repeated using the operator `RESH`.

### 7.3.5 REDEFINING A CELL

Once the lattice is created, it is possible to redefine, for a specific cell, all or part of the volumes comprising it. This is accomplished using the keyword `CELL` and specifying the volume to be replaced. In this case, the code itself creates a volume with the characteristics (shape and center) of the volume to be replaced. It is possible to replace a volume of the `COMBI` type only in this manner (see example in Appendix 2).

In the case of a lattice of lattices, it is also possible to exclude a specific cell from a lattice that is itself contained in a specific cell of a sub-lattice. This is effected using the keyword `LOWER_CELL` and specifying the volumes, cells, and lattices involved. Again, it is possible to replace a volume of the `COMBI` type only in this manner.

### 7.3.6 OPERATORS, VIRTUAL VOLUMES, AND SYNTAX FOR A VOLUME

A volume can be specified with respect to other volumes in the event of intersection (or union) using the operators SMASH, VMINUS, INTE, and UNION.
Operators can be introduced for volumes that are defined by surfaces.

In addition, a volume can have the attribute FICTIVE. In this case, the volume is virtual and is only used as an argument for operators, thus making it possible to modify the volumes during definition, or to override part of a volume (see below). In no case can a point in a geometry be specified as belonging to a virtual volume. A simple example is given to illustrate the UNION case.

Note: Operators are covered in this document before lattices, as it is important to clearly understand the concept of overriding (operator SMASH).

The general form of a volume definition is as follows:

**VOLU**
**num(volu)**
**<VOL_DEF>**
Volume description data
***<OPERATOR_NAME(1)>***
*Operator 1 data*
...
***<OPERATOR_NAME(n)>***
*Operator n data*
***FICTIVE***
***VOL_EXCEPT***
**ENDV**

where:

- num(volu) is the volume number. It will be used as the volume **identifier** in the rest of the input file. This number must not begin with a "0";
- <VOL_DEF> is one of the following keywords: COMBI, EQUA, RESC, RESH, CELL, LOWER_CELL, or ROTATION, depending on the mode of representation selected;
- <OPERATOR_NAME(i)> corresponds to one of the following keywords: SMASH, VMINUS, INTE, or UNION;
- FICTIVE indicates that the volume is virtual and is only used for purposes of modifying other volumes;
- VOL_EXCEPT indicates, if the volume belongs to the lattice, that it would be excluded for a set of cells (refer to 7.10.7.4, page 91 for information on when to use this option).

We will now consider how to specify the types and surfaces necessary for creating a simple volume.

## 7.4 DEFINING A TYPE

A **type** is a geometrical element used in a combinatorial definition (COMBI) of a volume. It consists of a **shape** and some additional descriptive data. Each type is associated with a specific number (integer). Any number can be used: the order is not important and skips in the numbering are allowed.

Note: A type can be defined at any time in the GEOMETRY module but must always precede the definition of a volume using it.

The declaration is made using the keyword TYPE, as follows:

**TYPE**
**num(type)**
**<SHAPE>**
Descriptive data

The type number num(type) is arbitrary and can be chosen by the user. It will be used as the type **identifier** in the rest of the input file.
The value of <SHAPE> is a geometry keyword, which may be one of the following:

- SPHERE;
- CYLX, CYLY or CYLZ (cylinder whose axis is parallel to an axis of the reference frame);
- CYL (any cylinder);
- BOX (parallelepiped along $X$, $Y$ and $Z$);
- CONE ;
- HEXAX, HEXAY or HEXAZ (hexagonal prism whose axes are parallel to an axis of the reference frame);
- PARALLELEPIPEDE (any);
- TORUS.

For all of the shapes allowed, we list the data needed to define a type and indicate their meanings.

### 7.4.1 SPHERES

**SPHERE r**

r is the radius of the sphere.

### 7.4.2 BOXES

**BOX hx hy hz**

`hx`, `hy`, `hz` are the lengths of the sides of the box (`hx` parallel to *OX* etc.)

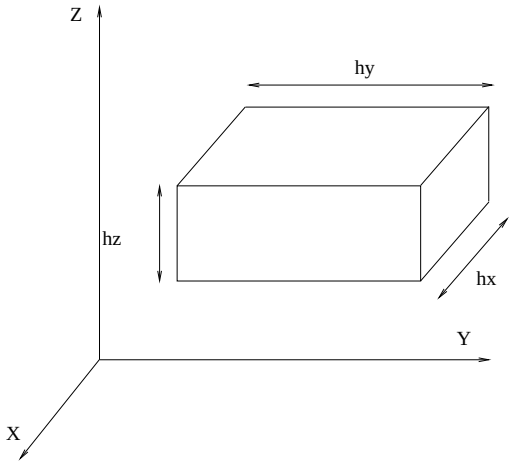



Figure 7.1: The BOX type

### 7.4.3 CYLINDERS PARALLEL TO AN AXIS

**CYLX r** $\frac{h}{2}$ or
**CYLY r** $\frac{h}{2}$ or
**CYLZ r** $\frac{h}{2}$

r is the radius of the cylinder
h is the height of the cylinder (h/2 half-height).
The cylinder is parallel to *OX*, *OY* or *OZ*.

### 7.4.4 GENERIC CYLINDERS

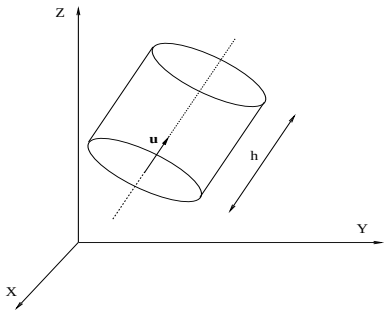



Figure 7.2: The CYL type

**CYL r $\frac{h}{2}$ $\mathbf{u}_x$ $\mathbf{u}_y$ $\mathbf{u}_z$**

r is the radius of the cylinder
h is the height of the cylinder (h/2 half-height).
$u_x$, $u_y$, $u_z$ are the components in the geometry reference frame of a vector (not necessarily a unit vector) giving the direction of the axis of the cylinder.

### 7.4.5 CONES

**CONE θ $\mathbf{u}_x$ $\mathbf{u}_y$ $\mathbf{u}_z$**

θ is the angle (in degrees) of the apex of the cone.
$u_x$, $u_y$, $u_z$ are the coordinates in the geometry reference frame of a vector (not necessary a unit vector) giving the direction of the axis of the cone, oriented from the apex to the base.

Note: The cone is the only TYPE that is not of a bounded volume, as the base of the cone is only determined when the VOLU command is input. When using the VOLU command, only the height of the cone needs to be given (the base being orthogonal to the axis).

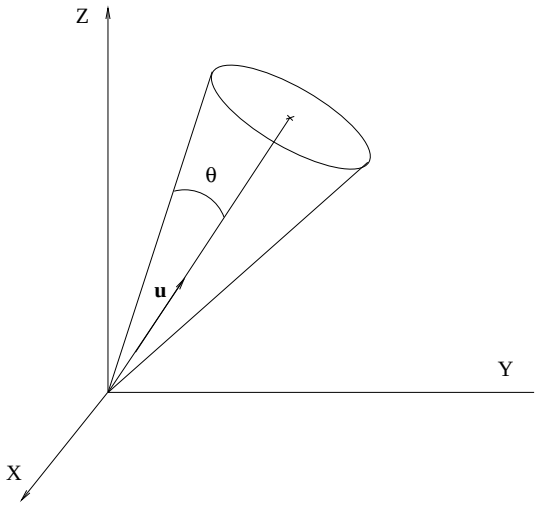



Figure 7.3: The CONE type

### 7.4.6 GENERIC PARALLELEPIPEDS

**PARALLELEPIPEDE**
**Lu Lv Lw**
$\mathbf{u}_x$ $\mathbf{u}_y$ $\mathbf{u}_z$
$\mathbf{v}_x$ $\mathbf{v}_y$ $\mathbf{v}_z$
$\mathbf{w}_x$ $\mathbf{w}_y$ $\mathbf{w}_z$

A parallelepiped is defined using three unit vectors **u**, **v**, and **w** (not necessary orthogonal or normalized). The values `Lu` , `Lv`, and `Lw` give the lengths of the sides along the vectors that define the parallelepiped.

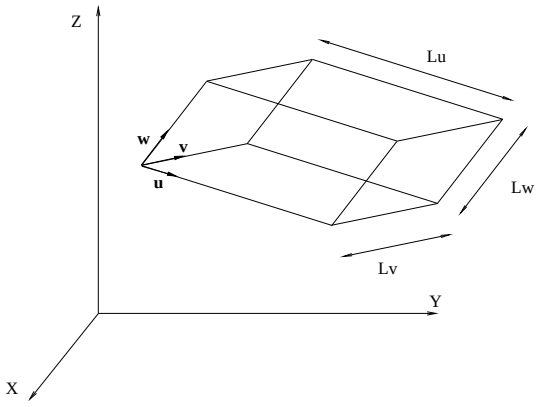



Figure 7.4: The `PARALLELEPIPEDE` type

### 7.4.7 HEXAGONAL PRISMS

**HEXAX** or **HEXAY** or **HEXAZ**
**diameter**
**height**
θ

A `HEXA[XYZ]` is a hexagonal prism with an $X$, $Y$, or $Z$ axis, depending on the keyword chosen. The diameter is that of the circle inscribed in the hexagon used to construct the prism. It is also the distance between two sides parallel to the axis ($D$ in the drawing below), the height ($h$ in the drawing) is the dimension of the prism along the axis and θ is an angle indicating the orientation of the hexagon. Currently, it can only be 0 or 90.

- In the case of a `HEXAX`, angle θ is "0" if the prism has two planes parallel to plane $OZX$ and "90" if it contains two planes parallel to plane $OYX$;

- In the case of a HEXAY, angle θ is “0” if the prism has two planes parallel to plane *OXY* and “90” if it contains two planes parallel to plane *OZY*;
- In the case of a HEXAZ, angle θ is “0” if the prism has two planes parallel to plane *OYZ* and “90” if it contains two planes parallel to plane *OXZ*.

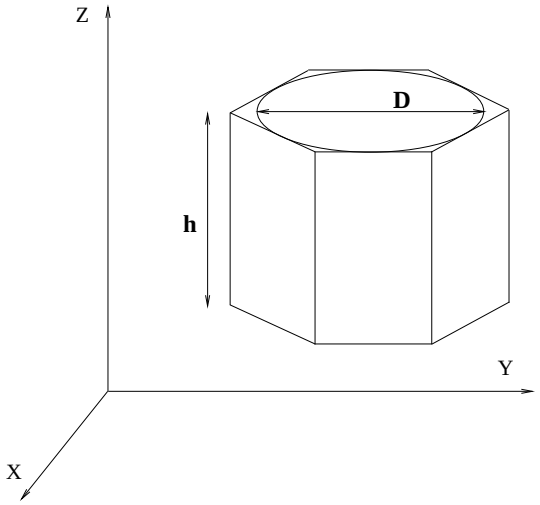



Figure 7.5: The HEXAZ type with θ = 0

### 7.4.8 TORI

**TORUS**
**$u_x$ $u_y$ $u_z$**
**R**
**r**

where:

- the values $u_x$, $u_y$, and $u_z$ are the components in the geometry reference frame of a vector (not necessarily a unit vector) giving the direction of the axis of the torus;
- *R* is the major radius, as shown in the following sketch,
- *r* is the minor radius, as shown.

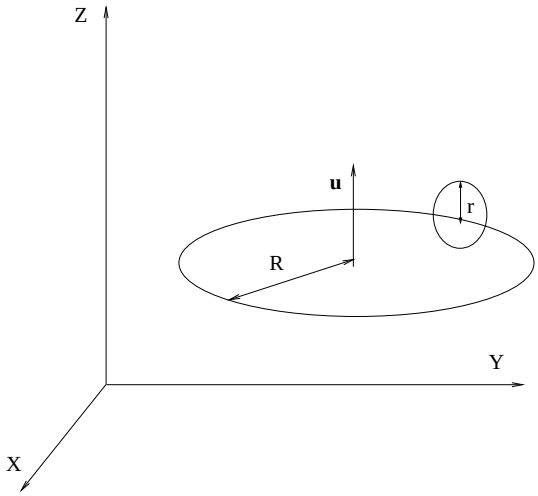



Figure 7.6: The data of the TORUS type

## 7.5 SURFACE DATA DEFINITIONS

A surface is declared using the keyword SURF in the following syntax:

**SURF**
**num(surf)**
<**SURFNAME**>
***TRANSFORM*** *transformation_identifier*
Descriptive data

The number of the surface num(surf) is arbitrarily chosen by the user. It will be used as the surface **identifier** in the rest of the input file. A surface can be declared at any time, but its definition must precede its use for a volume.

An optional keyword TRANSFORM with a transformation identifier can be specified. The transformation must have been declared anywhere before in the geometry (see 7.5.16).

The surfaces recognized by the geometry are:

- PLANEX, PLANEY, or PLANEZ (planes parallel to planes of the geometry reference frame);
- PLANE(general);
- SPHERE;
- CYLX, CYLY, or CYLZ (cylinders with axes parallel to the axes of the geometry reference frame);

- CONEX, CONEY or CONEZ (cones with axes parallel to the axes of the geometry reference frame);
- TORUSX, TORUSY or TORUSZ (torus with axes parallel to the axes of the geometry reference frame);
- QUAD (generic quadratics).

Each of the preceding surfaces divides the space into two regions: a positive region and a negative region. This property is used to define volumes of the EQUA type (surface-based).

In the following, we will give, for each type of surface, the appropriate keyword, the data required, as well as the relevant explanations. The sign convention for determining the position of a point relative to the surface is also indicated.

### 7.5.1 PLANE PARALLEL TO $(OY, OZ)$

**PLANEX** $\mathbf{x_0}$

$\mathbf{x_0}$ is the location of the plane.
A point $P = (x, y, z)$ is in the positive region if $x > x_0$.

### 7.5.2 PLANE PARALLEL TO $(OX, OZ)$

**PLANEY** $\mathbf{y_0}$

$\mathbf{y_0}$ is the location of the plane.
A point $P = (x, y, z)$ is in the positive region if $y > y_0$.

### 7.5.3 PLANE PARALLEL TO $(OX, OY)$

**PLANEZ** $\mathbf{z_0}$

$\mathbf{z_0}$ is the location of the plane.
A point $P = (x, y, z)$ is in the positive region if $z > z0$.

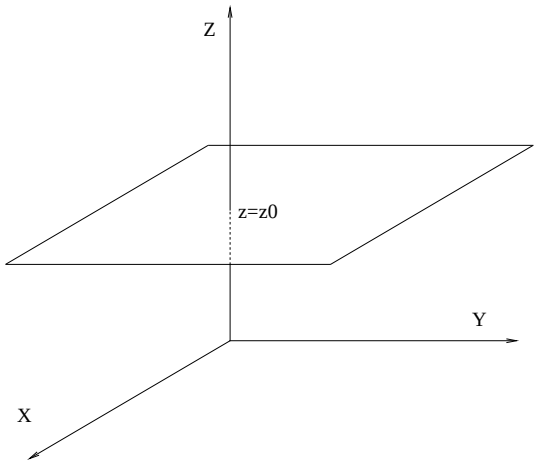



Figure 7.7: Example of a plane PLANEZ parallel to plane $(Ox, Oy)$.

### 7.5.4 GENERIC PLANES

**PLANE a b c d**

**a**, **b**, **c**, and **d** are such that the equation of the defined plane is:

$$ax + by + cz + d = 0.$$

A point $P = (x, y, z)$ is in the positive region if $ax + by + cz + d > 0$.

### 7.5.5 SPHERES

**SPHERE $x_0$ $y_0$ $z_0$ R**

($x_0$,$y_0$,$z_0$) are the coordinates of the center of a sphere in the geometry reference frame, and R is the radius of the sphere.
A point $P = (x, y, z)$ is in the positive region if it is outside the sphere.

### 7.5.6 A CYLINDER THAT IS PARALLEL TO AXIS *OX*

**CYLX $y_0$ $z_0$ R**

($\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of a point on the axis of the cylinder in the $(OY, OZ)$ reference frame, and R is the radius of the cylinder.
A point $P = (x, y, z)$ is in the positive region if it is outside the cylinder.

### 7.5.7 A CYLINDER THAT IS PARALLEL TO AXIS *OY*

**CYLY $\mathbf{x_0}$ $\mathbf{z_0}$ R**

($\mathbf{x_0}$,$\mathbf{z_0}$) are the coordinates of a point on the axis of the cylinder in the $(OX, OZ)$ reference frame, and R is the radius of the cylinder.
A point $P = (x, y, z)$ is in the positive region if it is outside the cylinder.

### 7.5.8 A CYLINDER THAT IS PARALLEL TO AXIS *OZ*

**CYLZ $\mathbf{x_0}$ $\mathbf{y_0}$ R**

($\mathbf{x_0}$,$\mathbf{y_0}$) are the coordinates of a point on the axis of the cylinder in the $(OX, OY)$ reference frame, and R is the radius of the cylinder.
A point $P = (x, y, z)$ is in the positive region if it is outside the cylinder.

### 7.5.9 A CYLINDER WITH A GENERIC AXIS

**CYL $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ R $\mathbf{u_x}$ $\mathbf{u_y}$ $\mathbf{u_z}$**

($\mathbf{x_0}$,$\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of a point on the axis of the cylinder and R is the radius of the cylinder. The axis of the cylinder is given by $u_x$, $u_y$, $u_z$.
A point $P = (x, y, z)$ is in the positive region if it is outside the cylinder.

### 7.5.10 A CONE WHOSE AXIS IS PARALLEL TO AXIS *OX*

**CONEX $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ θ**

($\mathbf{x_0}$,$\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of the apex of the cone; θ is half the apex angle.
A point $P = (x, y, z)$ is in the positive region if it is outside the cone.

### 7.5.11 A CONE WHOSE AXIS IS PARALLEL TO AXIS *OY*

**CONEY** $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ $\theta$

($\mathbf{x_0}$,$\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of the apex of the cone; $\theta$ is half the apex angle. A point $P = (x, y, z)$ is in the positive region if it is outside the cone.

### 7.5.12 A CONE WHOSE AXIS IS PARALLEL TO AXIS *OZ*

**CONEZ** $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ $\theta$

($\mathbf{x_0}$,$\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of the apex of the cone; $\theta$ is half the apex angle. A point $P = (x, y, z)$ is in the positive region if it is outside the cone.

### 7.5.13 A CONE WITH A GENERIC AXIS

**CONE** $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ $\theta$ $\mathbf{u_x}$ $\mathbf{u_y}$ $\mathbf{u_z}$

($\mathbf{x_0}$,$\mathbf{y_0}$,$\mathbf{z_0}$) are the coordinates of the apex of the cone; $\theta$ is half the apex angle. The axis of the cone is given by $u_x$, $u_y$, $u_z$.

### 7.5.14 AN ELLIPTIC TORUS

- whose axis is parallel to axis *OX*

**TORUSX** $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ **R a b**

- whose axis is parallel to axis *OY*

**TORUSY** $\mathbf{x_0}$ $\mathbf{y_0}$ $\mathbf{z_0}$ **R a b**

- whose axis is parallel to axis *OZ*

**TORUSZ $x_0$ $y_0$ $z_0$ R a b**

with :

**($x_0$, $y_0$, $z_0$)** : center coordinates of the torus;
**R** : main radius (also called ring radius) (circular);
**a** : semiaxis of the circular or elliptical section, perpendicular to $\vec{\mathbf{R}}$;
**b** : semiaxis of the circular or elliptical section, parallel to $\vec{\mathbf{R}}$;

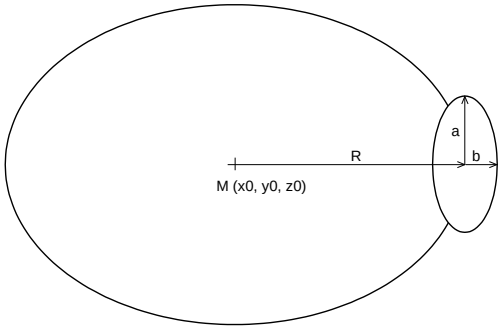



Figure 7.8: Schematic of the torus.

Thus, if **a = b**, the section is circular, otherwise the section is elliptical. It is authorized to choose **b** > **R**, the resulting volume can be seen as an apple. A point P=(x,y,z) is in the positive region if it is outside the torus.

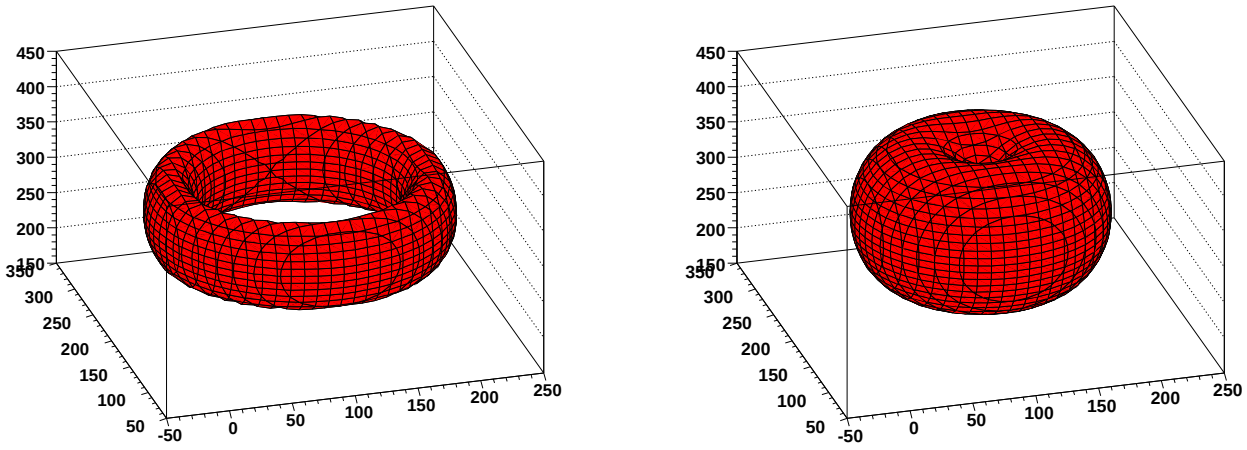



Figure 7.9: Examples of elliptical torus, with **a** > **b** (left figure), and **b** > **R** (right figure).

### 7.5.15 GENERIC QUADRATIC SURFACE

**QUAD a b c d e f g h i j**

**a, b, c, d, e, f, g, h, i, j** are the coefficients of the quadric equations:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fxz + gx + hy + iz + j = 0.$$

A point P=(x,y,z) is in the positive region if $ax^2 + by^2 + cz^2 + dxy + eyz + fxz + gx + hy + iz + j > 0$.

### 7.5.16 SURFACE TRANSFORMATION

A surface transformation is available applicable on all types of surface. It must be given in the form of a **unitary matrix** transformation and a change of origin.

**TRANSFORM**
**transformation_identifier**
**MATRIX**
$x_0$ $y_0$ $z_0$
$m_{11}$ $m_{12}$ $m_{13}$
$m_{21}$ $m_{22}$ $m_{23}$
$m_{31}$ $m_{32}$ $m_{33}$

The transformation is done to the point in the space before checking if the implicit equation is verified. If you want the transformation to be done on the object, you have to give the matrix of the inverse transformation. For example, the following declarations are equivalent

```
TRANSFORM 2
  MATRIX
     -2  0  0
        0  1  0
       -1  0  0
        0  0  1
SURF 1 TRANSFORM 2 SPHERE 4 0 0 16
```

and

```
SURF 1 SPHERE -2 -6 0 16
```

## 7.6 DEFINING SIMPLE VOLUMES

It should be noted that all volume definitions should take the following form (exluding operators with other volumes and the virtual attribute):

```
VOLU                    num(volu)
     ...
ENDV
```

where `num(volu)` is the number of the volume being defined.

A simple volume can be defined in two ways:

- combinatorial, mainly used for repetitive geometries (a single type is defined to create as many volumes as required).

  In certain cases, it is necessary to define virtual volumes (e.g., in the case of sectors) to cut the real volumes. On the other hand, it is then not possible to set boundary conditions on a virtual surface.

  Numerous consistency tests are available in this case (overlap testing, etc.).

- surface-based, adapted to geometries in which certain dimensions are often modified (depth of critical solution, for example). In the case of a combinatorial geometry, it is necessary to recenter the volumes for each calculation. The choice of a surface-based geometry usually allows faster execution than an equivalent combinatorial case involving a large number of operators. Unfortunately, there is no way to assure that the geometry is properly defined. It is entirely up to the user to define a consistent geometry (note that it is possible to request automatic biasing to provide some degree of testing).

### 7.6.1 DEFINING SIMPLE VOLUMES USING PREDETERMINED TYPES

A simple volume is defined on the basis of a type as follows:

**VOLU**
**num(volu)**
**COMBI**
**num(type)**
Additional data on volume
**ENDV**

where:

- `num(volu)` is the number of the volume being defined;
- `num(type)` is the number of the type (from a previously set) of the volume being defined.

The additional data concerning volume are identical for all types, exception cones. Cones are a special case and require some additional data.

#### 7.6.1.1 Sphere, cylinder with axis parallel to Ox, cylinder with axis parallel to Oy, cylinder with axis parallel to Oz, box

Such volumes can be defined using types as follows:

```
VOLU
        num(volu)
                COMBI
                        num(type)
                        cx cy cz
#               V_ORIGIN num(vol-orig)
#               M_ORIGIN
                        ni nj nk
                         LATTICE numres
ENDV
```

where:

- $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ are the coordinates of the center of the volume;
- `V_ORIGIN num(vol-orig)` means that the coordinates $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ are given with respect to the center of the volume `num(vol-orig)` (or the apex in the case of a cone), and not with respect to the origin $O$ of the reference frame;
- `M_ORIGIN (ni,nj,nk) LATTICE numres` means that the coordinates $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ are given with respect to the center of the cell `(ni,nj,nk)` of the lattice volume `numres` (see below).

Example:

```
TYPE 3 BOX 4 4 4
VOLU 7 COMBI 3 -2.  0.  0.  ENDV
VOLU 9 COMBI 3 4.  0.  0.  V_ORIGIN 7 ENDV
```

Two volumes (7 and 9) are built from a cube with 4 cm sides (`TYPE` 3). Volume 7 is centered at $(-2,0,0)$ and volume 9, positioned with respect to the center of 7, is centered at $(2,0,0) = (4,0,0) + (-2,0,0)$. Thus, by changing the coordinates of 7, volume 9 is also moved.

#### 7.6.1.2 Cone

```
VOLU
        num(volu)
                COMBI
                        num(type)
                        sx sy sz
                        height
#               V_ORIGIN  num(vol-orig)
#               M_ORIGIN
                        ni nj nk
                         LATTICE  numres
ENDV
```

where:

- $(\mathbf{s}_x, \mathbf{s}_y, \mathbf{s}_z)$ are the coordinates of the apex $S$ of the cone;
- `height` is the distance between the apex and the base of the cone, the base being orthogonal to the axis;
- `V_ORIGIN` and `M_ORIGIN`, along with their arguments, are used to give the coordinates of the apex with respect to the center of a volume or a cell, as with other combinatorial volumes.

### 7.6.2 DEFINING SIMPLE VOLUMES USING SURFACES

Simple volumes are defined in terms of surface as follows:

```
VOLU
        num(volu)
         EQUA
                PLUS  posnum
                        surf(1) ... surf(posnum)
                MINUS  negnum
                        surf(1) ... surf(negnum)
ENDV
```

where:

- `posnum` (or `negnum`) is the total number of surfaces with respect to which any point in the volume has a positive (or negative) sign (the sign determines which side of the surface contains the point);

- `surf(1) surf(2) ... surf(posnum)` (or `surf(1) surf(2) ... surf(negnum)`) is the list of surfaces (identified by their numbers) with respect to which the volume belongs to the positive (or negative) region. For the sign conventions, refer to the section on surfaces;
- the line `PLUS posnum` is not included when no surfaces are concerned (`posnum = 0`). The same applies to `MINUS negnum`.

Example:

```
      SURF 11 PLANEX -4    SURF 12 PLANEX 0
      SURF 13 PLANEY -2    SURF 14 PLANEY 2
      SURF 15 PLANEZ -2    SURF 16 PLANEZ 2
      VOLU 7 EQUA PLUS 3   11 13 15 MINUS 3   12 14 16 ENDV
```

The volume obtained is a cube with 4 cm sides centered on $(-2,0,0)$.

## 7.7 DEFINING ROTATION TYPE VOLUMES

This type of volume is called ROTATION. The necessary data include:

- the number for the volume used to create the current volume by rotation;
- the axis of rotation;
- the angle of rotation in degrees;
- the center of rotation.

The syntax is, therefore, as follows:

```
VOLU
        num(volu)
         ROTATION
                VOLU  num(volu-rot)
                      ux uy uz
                      θ
                      Ωx Ωy Ωz
ENDV
```

where:

- num(volu) is the number of the volume being defined;
- num(volu-rot) is the number of the simple volume to be rotated (volume COMBI or EQUA only, but not necessarily virtual);
- $(u_x, u_y, u_z)$ are the coordinates of a vector carrying the axis of rotation in the geometry reference frame;
- $\theta$ is the angle of rotation (in degrees);
- $(\Omega_x, \Omega_y, \Omega_z)$ are the coordinates of the center of rotation in the geometry reference frame.

**Restrictions on rotation**: A volume obtained by ROTATION must conform to certain restrictions

- it must be obtained from an element defined with a combinatorial or surface-based geometry (it is not possible to effect rotation on a lattice);
- it cannot override the volume from which it is derived (but the latter can be virtual);
- it cannot be augmented by the UNION operator (see below).

There is currently no checking of non-overlap for a ROTATION volume.

Examples:

```
TYPE 3 BOX 4 4 4
VOLU 7 COMBI 3 -2.  0.  0.  ENDV
VOLU 9 ROTATION VOLU 7
        0 0 1
        45
        10 0 0
ENDV
```

Two volumes are created: volume 7, a cube with 4 cm sides centered on $(-2,0,0)$, and volume 9, obtained by rotation of volume 7. The rotation is about axis $Z = (0,0,1)$ with a center at $(10,0,0)$ and an angle of 45 degrees.

```
SURF 11 PLANEX -4    SURF 12 PLANEX 0
SURF 13 PLANEY -4    SURF 14 PLANEY 0
SURF 15 PLANEZ -4    SURF 16 PLANEZ 0
VOLU 7 EQUA PLUS 3   11 13 15 MINUS 3   12 14 16 FICTIVE ENDV
VOLU 9 ROTATION VOLU 7
        0 0 1
        45
        10 0 0
ENDV
```

A single volume 9 is actually created. Volume 7 (a cube centered on $(-2,-2,-2)$ with 4 cm sides), is a virtual volume used to construct other volumes (9 in this case). Volume 9 is obtained by rotating volume 7. The rotation is about axis $Z = (0,0,1)$ with a center at $(10,0,0)$ and an angle of 45 degrees.

## 7.8 DEFINING OPERATORS RELATED TO VOLUMES

Operators can appear before the keyword ENDV, as they participate to the definition of a volume.
Several operators can appear in the definition of a single volume.
The following operators are currently known by the geometry:

- SMASH enabling one volume to override another;
- VMINUS, which has the opposite effect of SMASH;
- INTE for volume intersection;
- UNION for volume unions.

### 7.8.1 THE SMASH OPERATOR

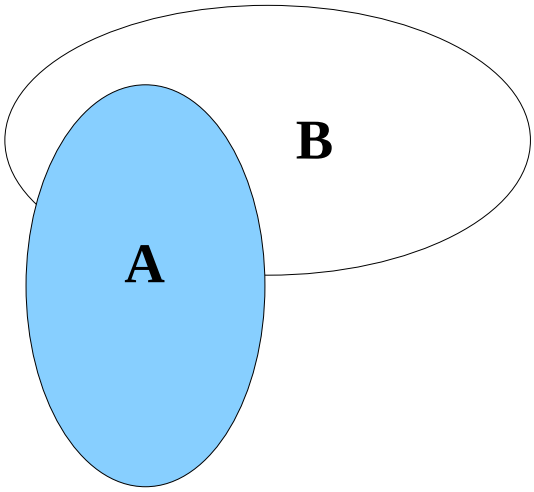
The SMASH operator is used to indicate, in the event of intersection of two volumes, to which volume the intersection belongs.

**SMASH**
**nb(volu)**
**volu(1) ... volu(n)**

where:

- nb(volu) is the number of volumes overridden by the volume being defined;
- volu(1) ... volu(n) is a list of the nb(volu) numbers (separated by spaces) of previously defined volumes overridden by the current volume.

The following figure illustrates the use of the SMASH operator (volume A overrides volume B if the intersection of A and B belongs to A. The grey part represents the result of A overriding B):




Example:

```
TYPE 3 BOX 4 4 4
TYPE 5 BOX 1 1 1
VOLU 7 COMBI 3 -2.  0.  0.  ENDV
VOLU 8 COMBI 5 -2.  0.  0.  SMASH 1 7 ENDV
```

Two cubes are created, centered on $(-2,0,0)$. The first, 7, has 4 cm sides (TYPE 3), the second, 8, has 1 cm sides (TYPE 5). Volume 8 is included in 7 and overrides it. The central part no longer belongs to 7.

#### 7.8.1.1 Properties of SMASH

Transitivity:
If A overrides B and B overrides C, then the part of A included in B overrides C. For example, if A is totally included in B, it is enough to state that B overrides C and it will make the code run **faster**.
On the other hand, if there is a part of A not included in B but overriding C, then it is necessary to add that A overrides C.

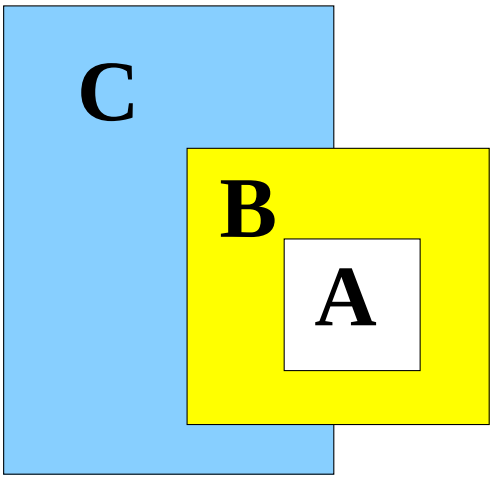



Figure 7.10: Here `A overrides B` and `B overrides C` implying that `A overrides C`

Important Restriction: if A and B are tangent, then the above property remains valid if and only if A and B are simple volumes, ie not constructed themselves with operators. If A and/or B are constructed with operators, then again it is necessary to add that A overrides C.

#### 7.8.1.2 The SMASH operator and virtual volumes

The `SMASH` operator is mandatory when two non-virtual volumes overlap. In the case where one of the two volumes is virtual and the other is not, the real volume overrides the virtual volume by default (as the latter does not physically exist), and it is not necessary to use the `SMASH` operator.

However, it is possible to make a virtual volume `B` override a real volume `A` to reduce it: in which case the common part *P* no longer belongs to the real volume. If volume `A` were to override volume `C`, part *P* would once again belong to `C`, otherwise care would be required to fill the space to obtain convex geometry.
Let us consider the following case:

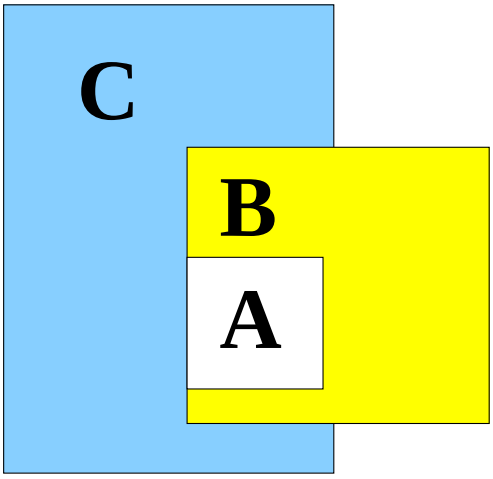



Figure 7.11: `A overrides B` and `B overrides C` AND A and B are tangent, CAUTION when using the transitivity property

```
TYPE 1 BOX 4 4 4
VOLU 1 COMBI 1 0 0 0 ENDV

TYPE 2 BOX 2 2 2
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

TYPE 3 BOX 1 1 1
VOLU 3 COMBI 3 0 0 0 SMASH 1 2 FICTIVE ENDV
```

Despite the transitivity, 3 does not override 1. As 3 is virtual, a “gap” remains at the center, which becomes part of 1 again.

#### 7.8.1.3 Rotation and operator SMASH

The volume of the rotation type cannot override the volume it is derived from.

### 7.8.2 THE `VMINUS` OPERATOR

This operator corresponds to the opposite of the `SMASH` operator. It can be used to truncate a volume by removing those areas where it intersects with other volumes. This operator is used as follows:

**VMINUS**
**nb(volu)**
**volu(1) ... volu(n)**

where:

- `nb(volu)` is the number of volumes truncating the volume defined.
- `volu(1) ... volu(n)` is the list of `nb(volu)` numbers of previously defined volumes truncating the current volume (at their intersection).

Figure 7.12 shows the operation performed by `VMINUS`.

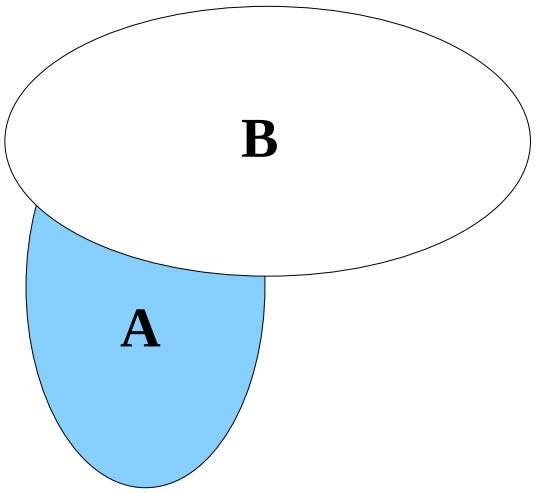



Figure 7.12: The grey part represents the results of `A VMINUS B`. $A - (A \cap B)$ is retained.

Example:

```
      TYPE 3 BOX 4 4 4
      TYPE 5 BOX 1 1 1
      VOLU 8 COMBI 5 -2.  0.  0.  ENDV
      VOLU 7 COMBI 3 -2.  0.  0.  VMINUS 1 8 ENDV
```

Two cubes are created, centered on $(-2,0,0)$. The first cube, named 8, has 1 cm sides (`TYPE` 5), the second cube, named 7, has 4 cm sides (`TYPE` 3). The central part does not belong to 7, as it is removed by `VMINUS`.

#### 7.8.2.1 Restrictions on the SMASH and VMINUS operators

It is not possible to have two overriding definitions that are contradictory: if A overrides B and B overrides C, then C cannot override A (using the `VMINUS` operator).

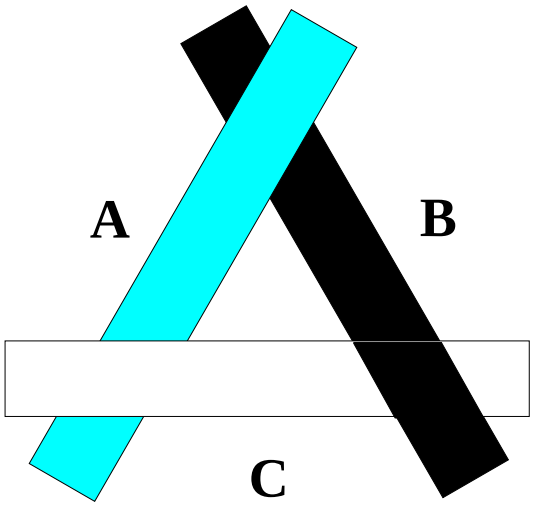



Figure 7.13: A overrides B, B overrides C, and C overrides A

The following approach at entering a geometry corresponding to Figure 7.13 is incorrect:

```
VOLU 1 (definition of A)
ENDV

VOLU 2 (definition of B)
     VMINUS 1 1
ENDV

VOLU 3 (definition of C)
     VMINUS 1 2
     SMASH 1 1
```

```
ENDV
```

To define Figure 7.13 it is necessary, for instance, to back up volume B with a virtual volume D (same definition as B but with a different number and with the FICTIVE attribute).

```
VOLU 1 (definition of A)
ENDV

VOLU 2 (definition of B)
     VMINUS 1 1
ENDV

VOLU 20 (definition of D identical to that of B)
     FICTIVE
ENDV

VOLU 3 (definition of C)
     VMINUS 1 20
     SMASH 1 1
ENDV
```

#### 7.8.2.2 Transitivity, link with the SMASH and FICTIVE operators

**Real volumes**

The VMINUS operator is almost symmetrical to SMASH. One peculiarity of the resulting transitivity is that if volume B cuts off part of volume A with VMINUS, then all of the volumes (C, ...) contained in B (as a result of SMASH or VMINUS) also cut off part of A, as shown in the following examples. (In certain cases, it would make more sense if only the part of C remaining in volume B made the cutting.)

Examples

Given that the following types are available:

```
TYPE 1 BOX 1 1 1
TYPE 2 BOX 2 2 2
TYPE 3 BOX 3 3 3
```

There are four possible ways to create drawing 7.14:

**Possibility 1: two SMASH operations**

```
VOLU 3 COMBI 3 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 3 ENDV
VOLU 1 COMBI 1 0 0 0 SMASH 1 2 ENDV
```

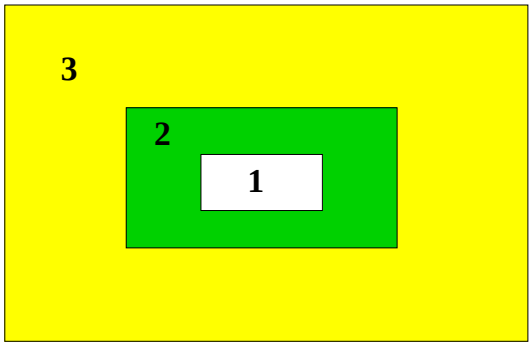



Figure 7.14: SMASH and VMINUS transitivity: the real volume case

### Possibility 2: two VMINUS operations

```
VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 VMINUS 1 1 ENDV
VOLU 3 COMBI 3 0 0 0 VMINUS 1 2 ENDV
```

### Possibility 3: one SMASH operation and one VMINUS operation

```
VOLU 2 COMBI 2 0 0 0 ENDV
VOLU 1 COMBI 1 0 0 0 SMASH 1 2 ENDV
VOLU 3 COMBI 3 0 0 0 VMINUS 1 2 ENDV
```

### Possibility 4: one VMINUS operation and two SMASH operations

```
VOLU 3 COMBI 3 0 0 0 ENDV
VOLU 1 COMBI 1 0 0 0 SMASH 1 3 ENDV
VOLU 2 COMBI 2 0 0 0 VMINUS 1 1 SMASH 1 3 ENDV
```

Note that an additional operator is required in this case, as 1 is defined before 2.

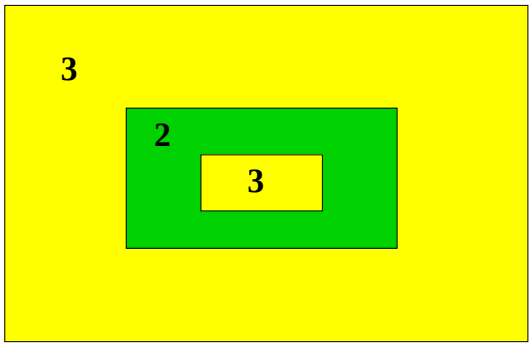



Figure 7.15: SMASH and VMINUS transitivity: the virtual volume case

### Introducing virtual volumes

In the case where the central part is removed by virtual volume 1, the symmetry between SMASH and VMINUS remains, and there are still four possible ways to recreate the situation shown in 7.15: adding FICTIVE to the volume 1 definition (in the fourth case the additional operator is not required).
In all cases, the central part becomes part of volume 3.

**Restrictions on the VMINUS and lattice operators**
VMINUS must not be part of the definition of the reference cell of a lattice (see below).

### 7.8.3 THE INTE OPERATOR

This operator is used to define a volume as the intersection of the volume being defined with one or more previously defined volumes. The volumes with which the current volume intersects must be virtual. The part of the current volume not included in the intersections disappears from the geometry. This operator is used as follows:

**INTE**
**nb(volu)**
**volu(1) ... volu(n)**

where:

- nb(volu) is the number of volumes to intersect the volume being defined;
- volu(1) ... volu(n) is a list of the nb(volu) numbers (separated by spaces) of previously defined virtual volumes.

Example:

```
TYPE 3 BOX 4 4 4
TYPE 5 SPHERE 2
VOLU 8 COMBI 5 -2.  0.  0.  FICTIVE ENDV
VOLU 7 COMBI 3 0.  0.  0.  INTE 1 8 ENDV
```

Volume 8, which is virtual, is only used in the construction of volume 7.
Volume 7 corresponds to the intersection between a cube with 4 cm sides centered on $(0,0,0)$ and a sphere with a radius of 2 cm centered on $(-2,0,0)$.

### 7.8.4 THE UNION OPERATOR

This operator is used to define a volume as the union of the volume being defined with one or more previously defined volumes. The volumes that are joined in the current volume must be virtual. This operator is used as follows:
where:

**UNION**
**nb(volu)**
**volu(1) ... volu(n)**

- `nb(volu)` is the number of volumes to be joined with the volume being defined;
- `volu(1) ... volu(n)` is the list of `nb(volu)` numbers (separated by spaces) of previously defined virtual volumes.

Verification of non-overlap for a volume with `UNION` is not currently implemented.
Note: `UNION` cannot be used with a `ROTATION` volume.

Example:

```
TYPE 3 BOX 4 4 4
TYPE 5 BOX 1 1 1
VOLU 8 COMBI 5 -2.  0.  0.  FICTIVE ENDV
VOLU 7 COMBI 3 0.5.  0.  0.  UNION 1 8 ENDV
```

A single volume, 7, is created. It consists of a cube with 1 cm sides, centered on $(0.5,0,0)$, which constitutes the simple volume. The `UNION` operator is used to add a cube with 4 cm sides centered on $(-2,0,0)$. Volume 8 is virtual and its only purpose is the construction of 7.

Important note: If the volume being defined is to override other volumes, then the `SMASH` operator (and its arguments) must not be included in the definitions of added virtual volumes. This is because they would, in this case, override the volumes as virtual ones.

Example:
Volume A results from the union of 1 (virtual) and 2, which overrides volume 10. The following syntax is correct

```
TYPE 100 BOX 5 5 5
VOLU 100 COMBI 100 0 0 0 ENDV

TYPE 10 BOX 4 4 4
VOLU 10 COMBI 10 0 0 0 SMASH 1 100 ENDV

TYPE 1 BOX 1 1 1
VOLU 1 COMBI 1 -1 0 0 FICTIVE ENDV
VOLU 2 COMBI 1 1 0 0 UNION 1 1 SMASH 1 10 ENDV
```

However,

```
TYPE 100 BOX 5 5 5
VOLU 100 COMBI 100 0 0 0 ENDV
```

```
TYPE 10 BOX 4 4 4
VOLU 10 COMBI 10 0 0 0 SMASH 1 100 ENDV

TYPE 1 BOX 1 1 1
VOLU 1 COMBI 1 -1 0 0 SMASH 1 10 FICTIVE ENDV
VOLU 2 COMBI 1 1 0 0 UNION 1 1 SMASH 1 10 ENDV
```
is not correct and will return an error.

## 7.9 OPERATOR PRIORITY

The code first searches for the presence of VMINUS, INTE, and/or UNION.

### 7.9.1 PRIORITIZATION OF VMINUS, INTE, AND UNION

If a number of UNION, INTE, and VMINUS operators are present, the operations are parsed from left to right.

Thus, to obtain

$$(1 \cap 2) \cup 3$$

it is necessary to write `VOLU 1 COMBI num x y z INTE 1 2 UNION 1 3 ENDV`.

However, to obtain

$$(1 \cup 2) \cap 3$$

it is necessary to write `VOLU 1 COMBI num x y z UNION 1 2 INTE 1 3 ENDV`.

### 7.9.2 PRIORITIZATION OF (INTE, UNION, VMINUS), AND SMASH

The code analyzes the SMASH operator differently from the others, as if it came last. This means that the volume that overrides is the simple volume modified by VMINUS, UNION, and INTE, even if these operations are given afterwards:
For
```
VOLU 1 COMBI num x y z SMASH 1 4 INTE 1 2 UNION 1 3 ENDV
```
**it is the entire volume [(1 INTE 2) UNION 3] that overrides volume 4**.

## 7.10 DEFINING A LATTICE

### 7.10.1 THE CONCEPT OF LATTICES

The GEOMETRY command is used to define orthorhombic lattices (parallelepipeds with individual patterns) and hexagonal lattices.
This type of volume is referred to as **RESC** (parallelepipeds) or **RESH** (hexagons). The data required to define these lattices are:

- the number of the volume establishing the reference cell;
- the number of repetitions of the volume along the three axes of the lattice.

The reference **cell** associated with the volume $V_{base}$ can only be a BOX, a PARALLELEPIPEDE, or a hexagonal prism (HEXAX, HEXAY, HEXAZ), defined using combinatorial geometry (no rotation, for instance). The sides (defined in the TYPE declaration) and the center of the volume determine the reference cell.

The **pattern** consists of all the elements located in the cell (and not only in the reference volume, as the box may have been overridden by other volumes) at the time of creation of the lattice. If another volume is located in the cell, it is also repeated.

For a RESC, the lattice consists of $n_x \times n_y \times n_z$ cells, numbered along the axes associated with the reference cell:

- In the case of a BOX, the indices of the cells along the axes increase as the axes increase;
- In the case of a PARALLELEPIPEDE, the indices increase in the direction of the descriptive vectors (see 7.4.6).

For a RESH, extension is more complex (see below).

At this point, a number of important remarks need to be made concerning the structure of this type of lattice.

The lattice volume created is an extension of the reference cell and does not actually bear the volume number num(volu) that was declared when defining the RESC (or RESH). The lattice is similar to a virtual volume. Each volume belonging to the reference cell is duplicated with its virtual or real attribute intact (adding FICTIVE to the lattice definition does not change this).
When overriding a lattice, the number to be given to the SMASH operator is the number of the lattice and not the number of the reference cell (or the volumes included in the cell). However, when associating compositions with volumes, materials are assigned to reference cell volumes, the composition of the lattice being unspecified. The volume of an element in the reference cell is multiplied by its occurrence in the lattice.

### 7.10.2 SYNTAX

The syntax used in defining a lattice differs slightly depending on whether the reference cell is a parallelepiped or a hexagonal prism. In the second case, it is necessary to create a lattice included in a large regular hexagon, which leads to restrictions concerning the number of repetitions. In both cases, it is possible to exclude cells, or only keep certain ones.

```
      VOLU
          num(volu)
                <RESCH>
                      VOLU  volume-cell
                          ni (nj) nk
                      BASIS
                          nbase(i) nbase(j) nbase(k)
#                     EXCEPT  cellcount
                          ni(1) nj(1) nk(1)
                          ...
                          ni(n) nj(n) nk(cellcount)
#                     KEEP  cellcount
                          ni(1) nj(1) nk(1)
                          ...
                          ni(n) nj(n) nk(cellcount)
                      MATCH
      ENDV
```

where:

- `num(volu)` is the number of the volume being defined;
- `<RESCH>` is equal to `RESC` for a `BOX` or `PARALLELEPIPEDE` lattice or `RESH` for a `HEXA[XYZ]` lattice;
- `volume-cell` is the number of the simple volume to be repeated in the lattice;
- $n_i, n_j, n_k$ are the numbers of repetitions of the reference cell in the three directions of the geometrical reference frame for a `BOX`, or in directions **(u,v,w)** for a `PARALLELEPIPEDE`. In the case of hexagonal lattices, $n_k$ refers to the axis and does not give $n_j$ (which is necessarily equal to $n_i$). These integers are always positive (see drawings below);
- the optional command `BASIS` indicates the indices of the cell to form the reference cell of the lattice. By default, the indices are $(1,1,1)$. For a $5 \times 5 \times 5$ lattice, `BASIS 3 3 3` enables the initial volume to be obtained (reference cell) at the center of the lattice;
- the optional command `EXCEPT` specifies the `cellcount` cells to be given special treatment:

  - either the volume of the reference cell is subsequently to be replaced by invoking the keyword CELL (see below) and the volume as well as all those that override it (directly or as a result of transitivity) are excluded from the lattice for the cells concerned (see 7.10.7.4, "Case 1" example on page 92).
  - or a volume belonging to the reference cell has been marked with VOL_EXCEPT (see 7.10.7.4 for meanings below, and the "Case 3" example on page 95); it will be excluded from the lattice (but not those that override it) for the cells concerned. If several volumes are marked, then they are all excluded;

- the optional command KEEP specifies the cellcount cells (integers, with all volumes) to be retained. By default, all cells are retained. This command is incompatible with the command EXCEPT. It is mandatory to retain the reference cell;
- the optional command MATCH, in the case of a hexagonal lattice, can be used to obtain lattice bounds that are always the bounds of a cell (see drawings 7.17 and 7.18).

Example:

```
TYPE 3 BOX 4 4 4
TYPE 5 BOX 1 1 1
VOLU 7 COMBI 3 -2.  0.  0.  ENDV
VOLU 8 COMBI 5 -2.  0.  0.  SMASH 1 7 ENDV
VOLU 10 RESC VOLU 7 3 2 2 ENDV
```

The reference cell consists of volume 7, a cube with 4 cm sides, centered on $(-2,0,0)$. As this volume contains volume 8 (as a result of SMASH), the latter is also repeated as part of the cell.
The lattice created is a $3 \times 2 \times 2$ lattice which extends over the parallelepiped $-4 < x < 8, -4 < y < 4, -4 < z < 4$.

### 7.10.3 LATTICE RESTRICTIONS

- The reference volume used to create the lattice obtained using RESC must not contain the VMINUS operator.
- The volumes to be repeated, as they are contained in the reference volume, must not extend beyond the cell (they must be completely contained within it).
- If the cells are not defined (deleted using EXCEPT or KEEP), it is necessary to redefine the missing region to ensure there are no gaps.
- Boundary conditions cannot be applied to hexagonal lattices defined with the MATCH keyword.

### 7.10.4 LATTICES AND THE SMASH OPERATOR

Suppose that volume 1, overridden by volume 2, is repeated as a result of the definition of a lattice 10. It is to be noted that the lattice has no physical existence: the particles belong to a volume of the reference cell (in this case, 1 or volume 2 included in 1).
However, as soon as the lattice has been defined, if volume 3 overrides part of the lattice, it must also override lattice 10 and not a volume (1 or 2) of the cell.

Syntax for the above example

```
TYPE 1 BOX 1 1 1
TYPE 2 BOX 2 2 2
TYPE 3 BOX 3 1 5

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1    18 18 18 ENDV

VOLU 3 COMBI 3 5 5 2 SMASH 1 10 ENDV
```

If volume 1 overrides volume 2, it is not necessary to state that lattice 10, built from 1, overrides volume 2.
Important: If the MATCH option has been used, this is, in fact, impossible.
(There are however a few particularities for lattices of lattices, as is indicated below.)

### 7.10.5 CELL NUMBERING

The numbering system is shown in the following figures. It is obvious for parallelepipeds and boxes, but more complex in the case of hexagons.

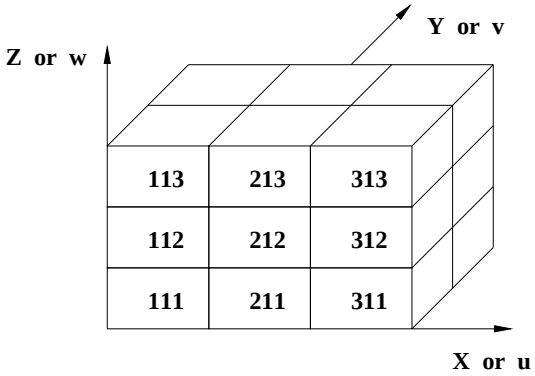



Figure 7.16: **RESC lattice composed of boxes or parallelepipeds**

An example is given of a hexagon of axis *Z*. The conventions for the other hexagons are obtained by circular

permutation (when in doubt, the user can use the POSITION option described at the beginning of this section).

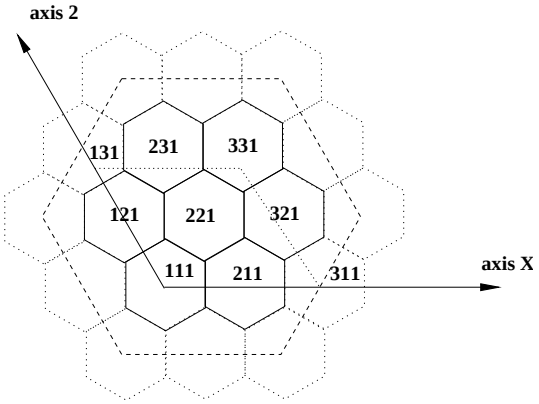



Figure 7.17: **RESH lattice composed of HEXAZ hexagons (angle θ null), section along $z$: ni = nj = 2.**

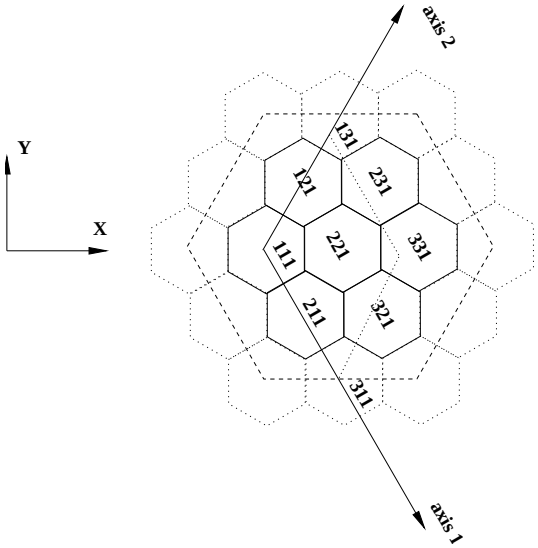



Figure 7.18: **RESH lattice composed of HEXAZ hexagons (angle $\theta = 90$), section along $z$: ni = nj = 2.**

The cells that do not belong to the lattice if the MATCH option has been indicated are shown in dotted lines. The boundaries of the hexagonal box surrounding the lattice are shown in lines of dashes: these are the boundaries if the MATCH option has not been selected. Indices of cells are determined by taking the coordinates of the center of the cell in the reference frame (non-orthogonal) defined by the axes $(X,Y)$: the dotted lines for the cell $(3,3,1)$.
In the hexagonal case, if the MATCH option has not been indicated, then the lattice will be a hexagon rotated by 90 degrees with respect to the cell orientations.

### 7.10.6 LATTICE OF LATTICES

It is now possible to create lattices of lattices, lattices of lattices of lattices etc. whenever required.
Let volume 1 be repeated in lattice 10 (all within volume 11):

```
TYPE 1 BOX 1 1 1
TYPE 2 BOX 0.5 0.5 0.5
TYPE 11 BOX 100 100 100

VOLU 11 COMBI 11 0 0 0 ENDV
VOLU 1 COMBI 1 0 0 0 SMASH 1 11 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1 2 2 1 (SMASH 11) ENDV
```

Two possibilities are available:

- repeating lattice 10 in lattice 100 (inside box 11 or slightly spilling over 11) by
  `VOLU 100 RESC VOLU 10 3 2 1 ENDV`.
  In this case, (`SMASH` 11) is not required;
- repeating volume 11, which contains lattice 10.
  `VOLU 100 RESC VOLU 11 3 2 1 ENDV`.
  In this case (`SMASH` 11) is required.

Restriction: In order to be repeated, a volume must override, either directly or by transitivity, the reference cell *M*. However, this is not necessary for lattice volume *R* (which makes it possible to use option `MATCH`, as in the example in the appendix). If, however, volume *V* overrides lattice volume *R*, difficulties can arise:

- Lattice *R* is the reference cell of the higher level lattice or overrides this reference cell. In this case there is no problem;
- Lattice *R* is in the reference cell of the higher level lattice, but does not override it (use of `MATCH`, for instance). Volume *V* must then explicitly override it (even if *V* is entirely within *R*) in order to be repeatable.

These restrictions also apply when the cells are defined using the options `CELL` and `LOWER_CELL`, which are covered at the end of this section. It is not necessary to state that these cells override a volume when the latter is overridden by the reference cell of a lattice *R*. This becomes necessary when *R* is inscribed, in a volume to be used as the reference cell of a lattice, without overriding it.

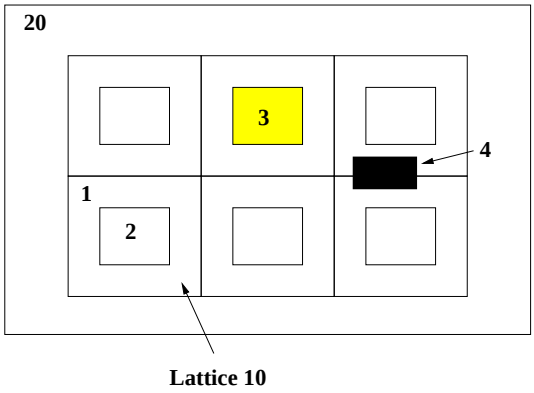



Figure 7.19: Here, volumes 1 and 2 are repeated in lattice 10, which is included in volume 20. Volume 3 re-defines part of a cell (via CELL) and volume 4 overrides 10. If volume 20 is used as the reference cell of a lattice and 10 does not override it, it is then **necessary** to specify that 3 and 4 override 20.

## 7.10.7 VOLUME DEFINITIONS BASED ON CELLS

### 7.10.7.1 Replacing a volume of a cell

In order to replace a particular volume in one or more specific cells, the volume can be redefined. To do so, it is necessary to indicate the lattice and cell involved, as well as the type of the volume to replace.
Using the cell indices supplied, the code then creates a volume of the combinatorial type. It is then no longer necessary for the user to worry about the coordinates of the center nor the dimensions of this combinatorial volume, which are calculated using the characteristics of the reference cell and the cell indices. The volume created can replace the entire cell or only part of it.

Given the case shown in Figure 7.20, corresponding to the following syntax:

```
TYPE 1 BOX 1 1 1
TYPE 2 BOX 0.5 0.5 0.5

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1 3 3 1 ENDV
VOLU 11 COMBI 2 1 1 0 SMASH 1 10 ENDV
```

### 7.10.7.2 Replacing a cell of a lattice

It is possible to replace the cell containing volume 11 using the CELL command, by specifying that the cell $(2,2,1)$ of lattice 10 must be re-defined with another volume.

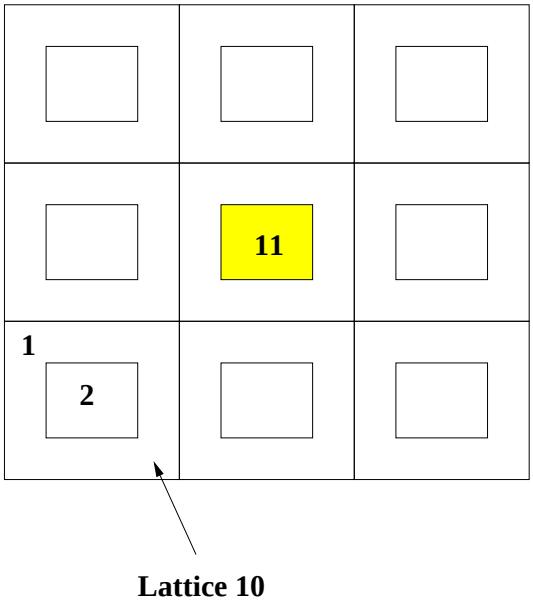



Figure 7.20: Redefining a cell

The syntax is:

```
VOLU
      num(volu)
       CELL  ni nj nk
             LATTICE  num(lattice)
             REPLACE  num(volu-replaced)
ENDV
```

where:

- `num(volu)` is the number of the created volume;
- `num(lattice)` is the number of the lattice in which the replacement is made;
- `(ni,nj,nk)` are the indices of the cell for which the replacement is made;
- `num(volu-replaced)` is the number of the volume defining the cell in the lattice which will be replaced by the volume `num(volu)` in the cell `(ni,nj,nk)` specified.

This volume replaces volume $V$ numbered `num(volu-replaced)` as well as all the volumes that override the latter. It is, therefore, necessary that all the volumes that override (either directly or through transitivity) the volume $V$ to be removed, be totally included within $V$.

The introduction example corresponds to the definition:

```
VOLU 11 CELL 2 2 1
        LATTICE 10
        REPLACE 1
ENDV
```

Note:

- If the lattice is used as the reference cell of a lattice of lattices, the replacement volume must be defined before reproducing the lattice. The replacement volume is also duplicated by the lattice operator. However, in this case it is necessary for either the lattice or the cell to explicitly override one of the volumes of the cell (see example in appendix).
- If the replaced volume overrode other volumes, the definition of the new replacing volume must take this into account.

**Important restriction**:
As stated, it is possible to exclude only a part of the cell rather than excluding the entire cell. However, when a volume (and all those that override it) is excluded, it will be excluded in **all** of the excluded cells of the lattice.
In particular, it is not possible, for instance, to exclude the totality of cell $(2,2,2)$ while only excluding part of cell $(3,3,3)$.
The exception property is a property associated with that volume, and not the cells that replace it.

#### 7.10.7.3 Replacing a sub-cell of a lattice of lattices

The situation is equivalent to that of replacement of cells, but, in this case, it applies to lattices of lattices.

```
VOLU
        num(volu)
         LOWER_CELL   DEPTH  depth
                 LATTICE  num(lattice,depth)
                      VOLU  num(depth)
                           CELL  ni nj nk
                 ...
                 LATTICE  num(lattice,1)
                      VOLU  num(depth=1)
                           CELL  ni nj nk
ENDV
```

where:

- `num(volu)` is the number of the volume being defined;

- `depth` is the depth of the sublattice for which the replacement takes place. `depth = 1` corresponds to the largest lattice, `depth = 2` to the next one etc. (see example below);
- `num(lattice,depth)` corresponds to the number of the lattice directly containing the volume to be replaced: it is necessary to give the number `num(depth)` of the reference cell volume to be replaced, as well as the indices of the cells;
- `num(lattice,i)` corresponds to the number of an intermediate lattice: it is necessary to give the number `num(i)` of the volume to be replaced, which is `num(lattice,i-1)`, as well as the indices of the cells;
- `num(lattice,1)` corresponds to the number of the largest lattice.

Unlike `CELL`, here it is the entire cell that must be replaced (not just a part).

Example:

Suppose that 1000 is a $3 \times 3 \times 2$ lattice of lattice 100, itself a $2 \times 2 \times 2$ lattice of lattice 10, which is finally a $5 \times 4 \times 3$ lattice of reference volume 1. If, in addition, volume 2 overrides volume 1, then we have:

```
TYPE 1 BOX 1 1 1
TYPE 2 BOX 0.5 0.5 0.5

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1 5 4 3 ENDV
VOLU 100 RESC VOLU 10 2 2 2 ENDV
VOLU 1000 RESC VOLU 100 3 3 2 ENDV
```

There are three levels of numbered lattices ranging from the largest to the smallest lattice:

- depth 1: lattice 1000, consisting of $3 \times 3 \times 2 = 18$ cells replicating 100;
- depth 2: lattice 100, consisting of $2 \times 2 \times 2 = 8$ cells replicating 10;
- depth 3: lattice 10, consisting of $5 \times 4 \times 3 = 60$ cells replicating 1 and 2.

There is a total of $18 \times 8 \times 60$ replications of 1 and 2.

If it is required to change sub-cell (2,1,1) of lattice 100 for cell (3,2,1) of lattice 1000 (depth `depth` = 2) by a volume `num(volu)`=102 (important: this sub-cell (2,2,1) consists of 60 cells of 1 and 2), it is necessary to

indicate the lattice, the volume to be replaced, and the cells for each level, as many times as the requested depth.

The syntax is then (from the smallest lattice to the largest, i.e., in the order of their definition):

```
VOLU 102
     LOWER_CELL DEPTH 2
                LATTICE 100 VOLU 10 CELL 2 1 1
                LATTICE 1000 VOLU 100 CELL 3 2 1
ENDV
```

**Restriction**:
The same restriction applies as in a definition using CELL: it is not possible to retain part of a cell for certain indices and to retain another part for other indices. If CELL and LOWER_CELL are used at the same time, the restrictions that hold for one also hold for the other.

#### 7.10.7.4 Replacing a cell using EXCEPT and VOL_EXCEPT

The user may wish to do one of the following:

- replace a set of cells using the same configuration;
- replace a volume without excluding the volumes that override it (replacement of a cladding, as in the example in 7.10.7.4 on page 97);
- replace a EQUA or ROTATION type volume.

Replacement using CELL or LOWER_CELL would be inadequate in the first case, inappropriate in the second (it would be necessary to redefine the overriding volumes), and impossible in the third.

These problems can be overcome using EXCEPT and VOL_EXCEPT:
By using the command EXCEPT, it is possible to exclude the same volume (or set of volumes) in a number of cells. However, if the cells are excluded using EXCEPT, it is necessary to know which volumes are to be excluded. There are two possible cases:

- redefining a cell by means of CELL or LOWER_CELL when the volume to be excluded is specified during the construction of the new volume and the overriding volumes are excluded by transitivity;
- neither CELL nor LOWER_CELL is subsequently called when it is necessary to specify the volumes to be excluded at the time of their construction using VOL_EXCEPT and the overridden volumes are not automatically excluded.

Note: If a volume is defined using CELL or LOWER_CELL, it must not be included in the list of excluded cells (otherwise the tests will be carried out twice on the same cell).

**Restrictions**:
The restrictions are the same as for CELL and LOWER_CELL: a volume marked by VOL_EXCEPT will always be eliminated from the excluded cells.

Examples:

**Case 1**
A case of two nested lattices.
In the case of a $5 \times 5 \times 1$ block (see Figure 7.21), 20 cells belong to lattice 10 and 5 to lattice 20.
By numbering the lower left-hand cell (1,1,1), the cells of lattice 20 are (2,2,1), (3,3,1), (3,4,1), (4,3,1), and (4,4,1).

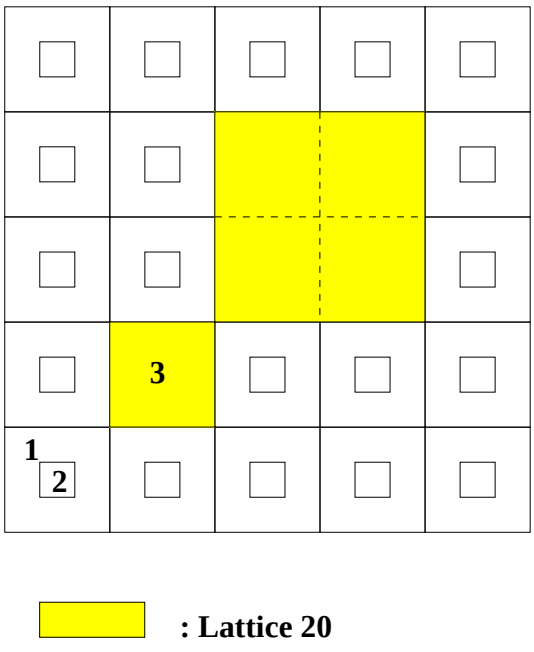



Figure 7.21: Lattice 10 (white) and 20 (gray) on a $5 \times 5 \times 1$ block, 2D-view on *OXY*

The largest lattice (in this case, lattice 10) is used as a reference. It is defined by repeating volume 1 containing volume 2. Note that, by default, the lower left-hand cell is number (1,1,1).

Then, the cells of lattice 20 are excluded (retaining 20 cells would be more expensive in terms of calculation time than excluding 5 cells would be). Cell (2,2,1) is re-defined using the CELL option, the 5 cells being explicitly excluded from lattice 10.

Cell (2,2,1) is re-defined (volume 3, which replaces volume 1). It is used as a reference cell for lattice 20.

Lattice 20 is defined, retaining only 5 cells (including the reference cell). Note that, by default, when a lattice is constructed, this cell is numbered (1,1,1) and it would, therefore, be possible to define lattice 20 by building a $3 \times 3 \times 1$ lattice, whose volume 3 is the reference cell (1,1,1) and for which only the cells (1,1,1), (2,2,1), (2,3,1) (3,2,1), and (3,3,1) are retained.
To simplify the numbering and construction of lattice 20, it is advisable to define a $5 \times 5 \times 1$ lattice with 3 as the reference cell but considered to be cell numbered (2,2,1) (using option BASIS). In this case, the indices of lattices 10 and 20 coincide and the cells to be retained have the same indices as those deleted in lattice 10.

```
TYPE 1 BOX 2 2 2
TYPE 2 BOX 1 1 1

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1 5 5 1
             EXCEPT 5
                    2 2 1
                    3 3 1
                    3 4 1
                    4 3 1
                    4 4 1
ENDV

VOLU 3 CELL 2 2 1 LATTICE 10
             REPLACE 1
ENDV

VOLU 20 RESC VOLU 3 5 5 1
             BASIS 2 2 1
             KEEP 5
                  2 2 1
                  3 3 1
                  3 4 1
                  4 3 1
                  4 4 1
ENDV
```

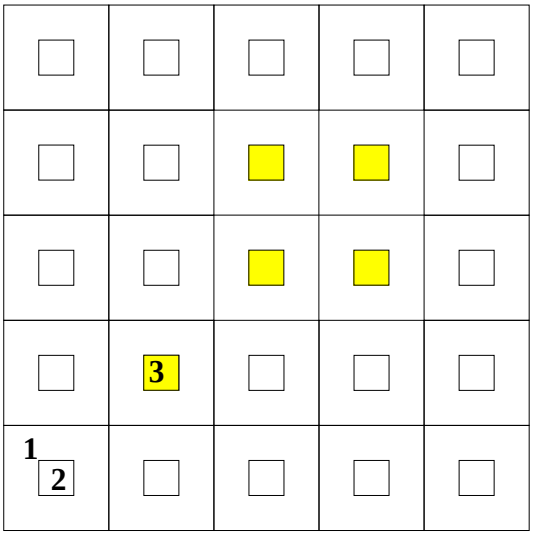
**Case 2**

Suppose that only volume 2 is to be replaced, rather than the entire cell. By using VOL_EXCEPT, it is possible to dispense with the transitivity laws of CELL. In this case, the user must:

- use VOL_EXCEPT to indicate that volume 2 is to be replaced;

- create a virtual volume 12, of the same size as volume 1, which “occupies” the place of cell (2,2,1). This volume is used as the reference cell of the lattice. This is because the translation vectors for the lattice must have the dimensions of the pitch of the first lattice;
- use `V_ORIGIN` to position volume 3 with respect to the origin of volume 12 and specify that volume 3 overrides the virtual volume, which, as a result of transitivity, means that 3 is in the virtual cell and may be translated;
- create a cell from virtual box 12.

It should be noted that the possibility of creating a lattice from the virtual cell only exists because the boundaries of lattice 12 coincide with the existing boundaries. If lattice 10 did not exist, then this construction would not be allowed.




**: Lattice 20**

```
TYPE 1 BOX 2 2 2
TYPE 2 BOX 1 1 1

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 VOL_EXCEPT ENDV

VOLU 10 RESC VOLU 1 5 5 1
             EXCEPT 5
                    2 2 1
                    3 3 1
                    3 4 1
```

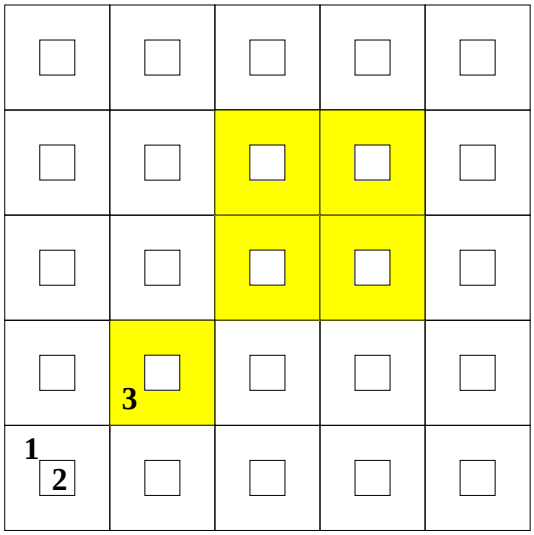
```
                    4 3 1
                    4 4 1
ENDV

VOLU 12 COMBI 1 0 0 0
             M_ORIGIN 2 2 1
             LATTICE 10 FICTIVE
ENDV
VOLU 3 COMBI 2 0 0 0
             V_ORIGIN 12
             SMASH 1 12
ENDV

VOLU 20 RESC VOLU 12 5 5 1
             BASIS 2 2 1
             KEEP 5
                  2 2 1
                  3 3 1
                  3 4 1
                  4 3 1
                  4 4 1
ENDV
```

**Case 3**

Let us now consider the case where volume 1, rather than volume 2, is to be replaced. The user must:

- use `VOL_EXCEPT` to specify that volume 1 is to be replaced ;
- exclude the cells where volume 1 is to be replaced as well ;
- create a virtual volume 12 of the same size as volume 2, centered on cell (2,2,1),
- create a volume 3 of the same size as volume 1, from which volume 12 has been removed (it is not possible to use `12 SMASH 1`, as the removal would be performed after the definition of volume 3 and, therefore, after the overlap tests were run: volume 10 and volume 3 would thus not have a zero intersection);
- create a lattice 20 from box 3.

```
TYPE 1 BOX 2 2 2
```




**: Lattice 20**

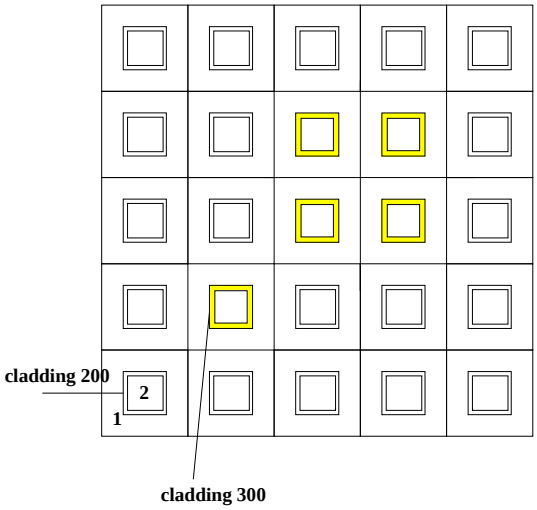
```
TYPE 2 BOX 1 1 1

VOLU 1 COMBI 1 0 0 0 VOL_EXCEPT ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV

VOLU 10 RESC VOLU 1 5 5 1
             EXCEPT 5
                    2 2 1
                    3 3 1
                    3 4 1
                    4 3 1
                    4 4 1
ENDV

VOLU 12 COMBI 2 0 0 0
             M_ORIGIN 2 2 1
             LATTICE 10 FICTIVE
ENDV

VOLU 3 COMBI 1 0 0 0
             VMINUS 1 12
             V_ORIGIN 12
ENDV
```

```
VOLU 20 RESC VOLU 3 5 5 1
             BASIS 2 2 1
             KEEP 5
                  2 2 1
                  3 3 1
                  3 4 1
                  4 3 1
                  4 4 1
ENDV
```

**Case 4**

Let us now consider the case where volume 2 has a cladding 200 that is to be replaced by cladding 300. Here, it is necessary to:

- use VOL_EXCEPT to specify that volume 200 is to be replaced;
- exclude the cell (2,2,1) of lattice 10 as well;
- create a virtual volume 22 of the same size as volume 2, centered on cell (2,2,1);
- create a virtual volume 12 of the same size as volume 1, centered on cell (2,2,1);
- create a volume 300, of the same size as volume 200, from which volume 22 has been removed and which overrides volume 12 (in order to be translated);
- create a lattice from virtual box 12.



```
TYPE 1 BOX 2 2 2
TYPE 2 BOX 1 1 1
TYPE 200 BOX 1.1 1.1 1.1

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 200 COMBI 200 0 0 0 SMASH 1 1 VOL_EXCEPT ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 200 ENDV

VOLU 10 RESC VOLU 1 5 5 1
             EXCEPT 5
                    2 2 1
                    3 3 1
                    3 4 1
                    4 3 1
                    4 4 1
ENDV

VOLU 22 COMBI 2 0 0 0
             M_ORIGIN 2 2 1
                      LATTICE 10
FICTIVE
ENDV

VOLU 12 COMBI 1 0 0 0
             V_ORIGIN 22
FICTIVE
ENDV

VOLU 300 COMBI 200 0 0 0
             V_ORIGIN 12
             SMASH 1 12
             VMINUS 1 22
ENDV

VOLU 20 RESC VOLU 12 5 5 1
             BASIS 2 2 1
             KEEP 5
                  2 2 1
                  3 3 1
                  3 4 1
                  4 3 1
                  4 4 1
```

```
ENDV
```

### 7.10.8 THE VIRTUAL CELL TRICK

It should be noted that a lattice of virtual cells cannot be used to create volumes, except under certain very special circumstances (such as Case 2 above).
However, it is possible to use such lattices to avoid lengthy calculations.
In the case of hexagonal arrays, when the prisms are not contiguous, the lattice operators become less efficient (too many cells have to be eliminated). Since calculating the positions of the centers can, nevertheless, be fairly arduous, the following solution can be used to facilitate the construction:

1. create a virtual volume of the size and position of the lattice cell;
2. create a lattice from the virtual cell;
3. arrange the volumes related to the lattice using the `M_ORIGIN` option.

The code then calculates the coordinates of the centers.

## 7.11 COPIES OF VOLUMES

It is possible for complete patterns to be identical in a geometry definition. If the patterns are not connected, if they are not numerous, or if surface-based scores are required, the lattice approach is not the solution. An option is available that avoids having to cut and paste within the data set (with the high risk of errors). This option consists of copying all the volumes contained in a given volume.
The syntax is as follows:

```
COPY
      num(vol-to-copy)
            X Y Z
       PLUS
      increment
...
ENDC
```

where:

- `num(vol-to-copy)` is the number of the volume to be copied (all the included volumes are also copied);
- `increment` is used to associate a number with the new volumes: if the volume to be copied is numbered 10 and it contains volume 12, and the increment is 100, the volumes duplicated are then numbered 110 and 112;
- `X Y Z` is the center of the volume copied in the geometry reference frame. Note: for a cone, it is the apex and not the center.

Restrictions:
Currently, only combinatorial volumes (conventional volumes or those generated by rotation, but not lattices) can be copied. They must not contain any lattice or surface-based volumes.

## 7.12 2D-VIEW DISPLAY

### 7.12.1 CHOICE OF COLORS

By default, 2D-graph colors are determined by the code. There is a limited number of colors, and it is, therefore, possibile for two adjacent regions to have the same color. To avoid this possible inconvenience and to associate colors with the composition and displays, the user can set volume colors using the `COLOR` command.

```
COLOR
        nb(colors)
        <COLOR(1)> nb(volu)
                volu(1)...volu(n)
        ...
        <COLOR(N)> nb(volu)
                volu(1)...volu(n)
```

where:

- `nbcolors` is the number of colors selected.
- `<COLOR(i)>` is one of the following: `BLACK`, `BLUE`, `BROWN`, `CORAL`, `CYAN`, `GOLD`, `GRAY`, `GREY`, `GREEN`, `MAGENTA`, `NAVY`, `ORANGE`, `PINK`, `PLUM`, `RED`, `SALMON`, `VIOLET`, `WHITE`, `YELLOW`.

Each color name is followed by the number of volumes and a list of their indices.

Note: the syntax is identical to that of the `GEOMCOMP` command, which defines composition-volume combinations. For combinations of colors by composition, the `GEOMCOMP` command can be duplicated and the composition names replaced by color names.

### 7.12.2 2D-CROSS SECTIONAL GRAPH REQUEST

The keyword `GRAPH` is used to create one or more 2D-graphs of the geometry.
The `GRAPH` command is executed directly after it is called. Invoking `GRAPH` between two volume definitions results in the display of a stage in the construction of the geometry; the volume definitions following the `GRAPH` call are not included. It is thus possible, in the same data set, to request 2D-graphs at a number of different angles or stages.
The results can be stored in the form of a PostScript file, which may be encapsulated or non-encapsulated. In this case, the user may choose whether or not the 2D-graph is to be displayed on the screen. This allows the user to obtain a graph of the geometry without stalling the calculation with a screen display.

The syntax is as follows:

```
GRAPH
        Px Py Pz
        Ux Uy Uz
        Vx Vy Vz
                Lu Lv
                Flag
         FILE filename
```

2D-graphs are displayed in a plane containing the point $P$ =(Px,Py,Pz) and defined by two vectors **U**=(Ux,Uy,Uz) and **V**=(Vx,Vy,Vz).

The window is determined by point $P$ (bottom left) and length Lu along **U** and **Lv** along **V**.

Flag is set to:

- 0 for no PostScript file;
- 1 for a PostScript file with a caption in the drawing;
- 2 for a PostScript file without a caption;
- 3 for an encapsulated PostScript file with a caption;
- 4 for an encapsulated PostScript file without a caption;
- 5 for a Postscript file with a caption and with no screen display.
- 6 for a PostScript file without a caption and with no screen display;
- 7 for an encapsulated PostScript file with a caption and with no screen display;
- 8 for an encapsulated PostScript file without a caption and with no screen display.

The option FILE allows for the user to specify the file name associated to the graph : PostScript files are named **"filename.ps"**, and encapsulated PostScript files are named **"filename.eps"**.
By default, PostScript files are named **"graphicN.ps"**, and encapsulated PostScript files are named **"graphicN.eps"** (these can be embedded in text), where $N$ is the number of the graph. They are numbered in the order in which they are requested.

When Flag is set to 0, 1, 2, 3, or, 4, once the display is complete, the user must press the "ENTER" key to continue. To interrupt the calculations at this point, the 2D-graph window can simply be closed. This makes it possible for a "core dump" not to be generated if the following data is missing, for instance.

Note: The geometry is displayed by means of two functions (position and distance calculations) used in the tracking of particles. The sweep is made by a line parallel to axis **U**.
As during calculation, a particle leaving the geometry cannot return to it. Therefore, if the geometry is non-convex (which is not the normal case), the display can "cut" a volume.
In the example in Figure 7.22, line $L$ sweeps the domain (fromleft to right) as follows:

- calculating the point of entry $P(0)$, defining the entrance volume (1, in this case);
- calculating the point of exit $P(1)$ of volume 1;
- determining the new volume (2, in this case);

- calculating the point of exit $P(2)$ from volume 2;
- determining the new volume, in this example, leakage out of the geometry;
- moving on to ray trace the next line.

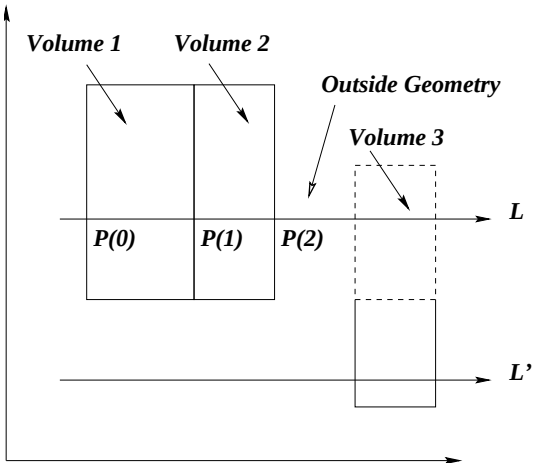



Figure 7.22: The non-convex geometry problem. Undisplayed part in dotted lines

As such, volume 3 is not encountered. However, it is on line $L'$, as there is no gap in the line ray tracing.

## 7.13 GEOMETRY CHECKS

### 7.13.1 COMBINATORIAL GEOMETRY CHECKS

When a combinatorial geometry is defined, the code always performs a test for volume overlapping as the geometry is read.
It is possible to adjust the number of points generated when testing for combinatorial volume non-overlaping. When the keyword "`TEST_GENERATED_POINTS` n" is not used, the default value for n is 200.
It can be set in two ways:

- only once at the beginning of the geometry definition (after the directive `TITLE`)
- or as many times as necessary in the directive `GEOMETRY`, preceding the keyword `VOLUME` (for each combinatorial volume, the number of generated points refers to the previous value of `TEST_GENERATED_POINTS`).

### 7.13.2 REQUESTING CHECKS

No matter how the volumes of the geometry are defined, the user may request checks on either a part or all of the geometry using the command `CHECK_GEOM`. This is particularly useful in surface mode where the default

check (see $ 7.13.1) is not performed. The check is performed in a window defined by coordinates specified by the user.
The syntax, to be placed after the keyword ENDGEOM, is as follows :

```
CHECK_GEOM
        FRAME
                <FRAME_TYPE>
                Ox Oy Oz
                ix iy iz
                (jx jy jz)
                kx ky kz
        WINDOW
                Umin Vmin Wmin
                Umax Vmax Wmax
        SOBOL_POINTS  nb(points)
        LIST  nb(points)
        CHECK_HOLES
END_CHECK_GEOM
```

where:

- the window is defined by specifying the type of coordinates (CARTESIAN, CYLINDER, or SPHERE), the origin **O** and the associated axes **i**, **j**, **k**. The **j** axis is only specified in the case of Cartesian coordinates (see section 4.5 for a complete explanation of frames).
- the window for the checks is defined by two minimum and maximum side points, with coordinates as follows: ($U_{min}$, $V_{min}$, $W_{min}$), and ($U_{max}$, $V_{max}$, $W_{max}$) expressed in the desired type of coordinates (respectively (x,y,z), (r,θ,z) and (r,ϕ,θ)).

This functionality performs a check for overlapping volumes using a number of points randomly sampled in the geometry. The default value for this number of points is 50, but may be modified using the optional keyword SOBOL_POINTS.
The set of sampled points belonging to at least two volumes is displayed. For each of these points, the overlapping volumes are identified. If the list of overlapping points is large, the user can reduce the display by setting the number of elements in the list using the keyword LIST. If at least one point is found to lie within overlapping volumes, the calculation will not be performed.
The keywords SOBOL_POINTS and LIST can be used independently. Nevertheless, if both of the keywords are specified, they must be entered in the order shown above.
Finally, the user may also specify the CHECK_HOLES option, which is used to check for the existence of holes in the geometry. In such a case, the complete list of points not belonging to any volume of the geometry is provided and the calculation is stopped. For this option, the user must ensure that the defined window is indeed included in the geometry.

When the keyword CHECK_HOLES is not used, a check is performed to search for holes, but only a list of approximately ten points is displayed. This allows the user to define a window (for the overlapping check) containing the whole geometry, without stopping the procedure due to the hole checks.

Note : For a combinatorial geometry, the CHECK_GEOM command is especially useful in verifying the convexity of the geometry (absence of holes).

### 7.13.3 1D SCAN OF GEOMETRY

This functionality enables to scan geometry along n paths. For each path, the user has to give the coordinates of the starting point and of the arrival point. For each path, it gives boundary points, crossed volumes, and distances in these volumes. It only requires geometry, as in CHECK_GEOM.
The syntax, to be placed after the keyword ENDGEOM, is as follows :

**SCAN_GEOM_1D**
**PATH** $\mathbf{nb}_{(paths)}$
$\mathbf{X}_{start(path(1))}$ $\mathbf{Y}_{start(path(1))}$ $\mathbf{Z}_{start(path(1))}$
$\mathbf{X}_{stop(path(1))}$ $\mathbf{Y}_{stop(path(1))}$ $\mathbf{Z}_{stop(path(1))}$
$\cdots$ *start(path(i))*
$\cdots$ *stop(path(i))*
$\mathbf{X}_{start(path(n))}$ $\mathbf{Y}_{start(path(n))}$ $\mathbf{Z}_{start(path(n))}$
$\mathbf{X}_{stop(path(n))}$ $\mathbf{Y}_{stop(path(n))}$ $\mathbf{Z}_{stop(path(n))}$
**END_SCAN_GEOM_1D**

### 7.13.4 1D SCAN OF GEOMETRY-COMPOSITION ASSOCIATION

This functionality enables to scan geometry-composition association along n paths. For each path, the user has to give the coordinates of the starting point and of the arrival point. For each path, it gives boundary points, crossed volumes, distances in these volumes and compositions. It requires a complete data file. It gives the results after reading materials.
The syntax, to be placed after the keyword ENDGEOM, is as follows :

```
SCAN_GEOMCOMP
      1D_SCAN
            PATH  nb(paths)
                  X_start(path(1))   Y_start(path(1))   Z_start(path(1))
                  X_stop(path(1))    Y_stop(path(1))    Z_stop(path(1))
                  ... start(path(i))
                  ... stop(path(i))
                  X_start(path(n))   Y_start(path(n))   Z_start(path(n))
                  X_stop(path(n))    Y_stop(path(n))    Z_stop(path(n))
      END_1D_SCAN
END_SCAN_GEOMCOMP
```

### 7.13.5 REQUESTING CHECKS USING TRACKING ROUTINES

This functionality enables to check geometry using tracking routines, while CHECK_GEOM functionality (see $ 7.13.2) uses positioning routines. Therefore, CHECK_GEOM and CHECK_TRACKING are redundant and complementary functionalities that help the user to check the geometry in an exhaustive way. In the case of CHECK_TRACKING, the user will detect the straight-line paths along which tracking is not correct, if any. For example, this can be induced by an overlap between two or more volumes or by a hole in the geometry. The check is performed in a sphere specified by the user, inside which a user-defined number of straight-line paths are sampled and tested for tracking in both directions. The syntax, to be placed after the keyword ENDGEOM, is as follows :

```
      CHECK_TRACKING
*           SPHERE
*           POINT
                  X0 Y0 Z0
                  R
                  nb(paths)
      END_CHECK_TRACKING
```

where:

- SPHERE or POINT indicates the type of test to be performed (see below)
- $X_0$, $Y_0$, and $Z_0$ is the center of the sphere
- R is the radius of the sphere
- nb(*paths*) is the number of straight-line paths to be sampled and tested

The user must choose between a “sphere-sphere" test using SPHERE keyword and a “point-sphere" test using POINT keyword. In the former case, the starting point and the arrival point of the sampled paths belong to the sphere surface. In the latter case, one of the two points is the center of the sphere, allowing the user to focus on a region. If the result of CHECK_TRACKING is positive (indicating that the geometry contains some errors), the code will give the starting point and the arrival point of all paths along which the tracking is not coherent and will try to identify the volume that needs to be fixed (it can be a neighbour of this volume).

Comments:
An error can be detected several times, thus producing a long listing. An error might affect a small spatial region, and thus have negligible consequences on the calculation (this depends also on the features of the system and the desired scores). The impact of the detected errors is therefore to be evaluated according to the distinct system under investigation and general conclusions can not be drawn. In order to support the geometry analysis, one can then use SCAN_GEOM_1D or SCAN_GEOMCOMP in both directions.

Advanced use:
The user can specify multiple CHECK_TRACKING in the same input deck.
The user can run CHECK_TRACKING in parallel mode. In this case, the results of the process with rank 2 in parallel mode will give the same results than the single process in mono-process mode.
An interesting post-processing of the results consists in plotting a 3D graph of all paths along which an error occured: in the case of errors concerning small regions, one can detect many intersecting paths lying within a small region of space, showing the location of the problem.

## 7.14 INFLUENCE OF DATA INPUT ON CALCULATION TIME

Calculation time during simulation (and not only during data verification) depends on the number of operators (SMASH, UNION, etc.). A geometry input in combinatorial form can, therefore, require more time than a surface-based geometry.

It is, however, necessary to find a suitable compromise between the time necessary for the user to define a surface-based geometry, which can be longer (and without the combinatorial definition verifications) and the longer calculation time using the combinatorial approach.

### 7.14.1 CONNECTIVITY MAPS

#### 7.14.1.1 Usage

The optimisation algorithm for resolving geometry calls is based on memorising neighbours of each volume. When particles get out of a volume and the code needs to find the next volume, a search is first done inside a table of nearest neighbours. If it fails, a new search is then done inside all the other volumes (excluding previous searches in simplest cases). When working with large geometries (more than 50000 volumes for example), and if neighbour tables have not been built efficiently, the search may lead to large simulation time.

In the case of combinatorial geometries, the neighbours table is usually efficiently computed during the initialisation. With some surface based geometries, the neighbours table is not very efficiently done during the initialisation.

In any case, when a volume is found by a search in the whole geometry, the volume is then added to the neighbours table so the code is optimising itself during the simulation.

It may be interesting to provide a neighbours table for the code. The table needs not to be complete. In this case, the user must invoke the keyword `USE_CONNECTIVITY_MAP` with the filename. **Important:** the user must place the use call when all volume cited in the file are defined. The most convenient place is in general *at the end of the geometry block*.

Alternatively, the user may store the connectivity map built inside the code at the end of the simulation using the keyword `STORE_CONNECTIVITY_MAP` with the filename. This command has to be placed at the end of the geometry block. Note that the file will be created at the end of the simulation. If the calculation is replayed using the exact same geometry and the geometry contains a very large number of volumes, the second simulation will be much faster.

This can for example be useful when calculating a very large number of scores (so that the calculation will not fit into memory). The user may split the scores in several parts and rerun the same simulation with different scores.

The connectivity map loading has been made recursive, thus it is possible to load multiple connectivity maps. The user can produce some connectivity maps in parallel (that is with independent simulations), and load them all in a second parallel simulation providing some bootstrapping parallelisation effect.

The syntax, for using or creating connectivity maps, is as follows :

**USE_CONNECTIVITY_MAP** <filename>
**STORE_CONNECTIVITY_MAP** <filename>

#### 7.14.1.2 Map Format

Please note this format is subject to change after version 12 of the code, in order to address reading performance issues when there are many volumes (>100000).

The present format is as follows:

```
has_stat|no_stat
iVol0 iNbNei0 iNei0,0 iCnt0,0 iNei0,1 iCnt0,1 ...
iVol1 iNbNei1 iNei1,0 iCnt1,0 iNei1,1 iCnt1,1 ...
...
```

The first line is either `has_stat` or `no_stat` depending on if we give the visiting count (or an integral number increasing with the probability to visit this neighbour). `iVol0` is the identifier of the first volume, followed by a table of its neighbours for the rest of the line. `INei0` is the number of neighbours of `iVol0`. Following this are all the neighbours of `iVol0` one by one, with the identifier (`iNei0,0` for the first) and the number of times this neighbour has been visited (`iCnt0,0`).

It is possible to set the visit count to zero at the cost of a performance decrease (likely slight and forgotten after a few batches). The most important is to have a table that includes the majority of frequently visited neighbours, possibly sorted in descending order of visit counts.

Generally, this table will be sorted in descending order of visits within the first hundred batches to optimize the searches. In cases where the number of neighbours is exceptionally large, sorting this table will be disabled as it would be counterproductive.

### 7.14.2 SURFACE-BASED GEOMETRIES

The keyword `HASH_TABLE` at the beginning of the geometry can be used to speed-up the reading of large surface-based geometries. Very small geometries (requiring for instance less than a second to initialize) may be slower, but large geometries (bigger than 100000 volumes for example) will be faster. Speed-up factors larger than 10 for the initialization time have been observed on large geometries. This option does not affect the transport (simulation time and results).

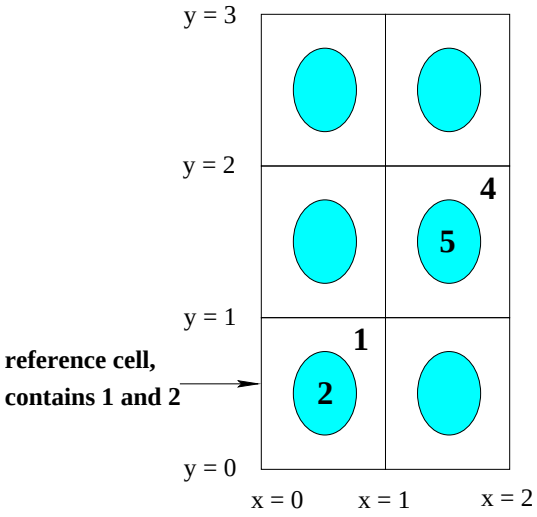
### 7.14.3 COMBINATORIAL GEOMETRIES

Care should also be taken to avoid superfluous operators: not to have one volume override another when it is possible to use the benefits of transitivity and to avoid unnecessary unions when the intersection is empty.

One important implication concerns lattice-based geometries. The user may wish to calculate a surface-based flux or current or a surface-based reaction rate in part of the lattice. It is possible to override the part involved (shown in gray in the diagram) by one or more other volumes, in order to differentiate them.

Example: The above sketch corresponds to the following sequence:

```
TYPE 1 BOX 2 1 1
VOLU 1 COMBI 1 1 0.5 0 ENDV

TYPE 2 SPHERE 0.4
VOLU 2 COMBI 2 1 0.5 0 SMASH 1 1 ENDV

VOLU 3 RESC VOLU 1 2 3 1
ENDV

VOLU 4 COMBI 1 3 1.5 0 SMASH 1 3 ENDV
VOLU 5 COMBI 2 3 1.5 0 SMASH 2 3 4 ENDV
```

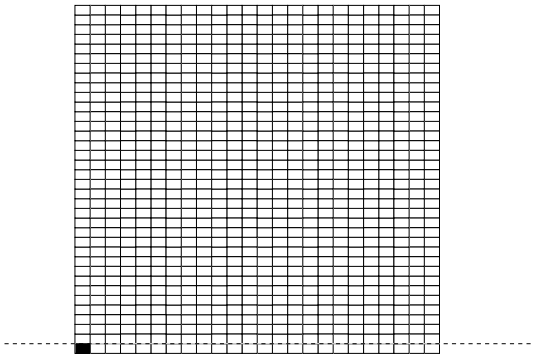



**Lattice 3**

flux between volumes 4 and 5 can be requested (see section on scores).

**It is, however, possible to redefine the cells or parts of them without resorting to using the SMASH operator.** Preference should be given to this approach (i.e., redefinition of the cells).
**With large lattices, use of SMASH is to be avoided.**

Calculation time depends on operators and, in this case, for each point of the lattice, the code tests whether the part involved is not overridden by other volumes. The time lost is considerable if a large number of points are involved. **When the SMASH operator has been used, the user must attempt to split the lattice and define several different lattices**: the lattices that are to be overridden and those that are never overridden. In this manner, the latter group of lattices does not slow down the calculation because of unneccessary tests to determine whether something is being overridden or not.



The example shown in the figure above (limited to a two dimensional example of 24x36 cells for the sake of simplicity) in which the results in cell (1,1) are requested constitutes an extreme case. Rather than having a

complete lattice overridden in a single cell, it is better to define this cell (1,1) separately, then a 1x23 lattice (the first line minus the cell shown in black) with cells of (2,1) to (24,1) and a lattice of 23x36 (above the dotted line (2,1) of cell (2,1) to cell (24,36). This avoids using the operator SMASH completely.

Of course, avoiding the use of SMASH altogether would compromise of the power of RESC, but an effort should be made to limit the number of operators on a lattice (particularly by avoiding setting the bounds of a complete lattice using INTE and VMINUS if the central part of the lattice is to be retained as a "pure" lattice and only set the bounds on the periphery with a second lattice).

Finally, one thing that is common in both surface-based and combinatorial geometries, is that the calculation time depends on the number of potential neighbors the volumes have. For example, if a reactor core is naively defined using a surface-based geometry, the result is a cylinder for the vessel and tens of thousands of cylinders for each cell (3 per cell). A neutron moving through the moderator could then encounter each cell on its trajectory and thus result in the code making tens of thousands of unnecessary intersection calculations (as the probability of a neutron seeing a cell from more than one fuel assembly away is very slight). Calculation time would be equivalent to that obtained using a combinatorial geometry with each cell overriding the vessel.

## Surface-based geometry definition for the criticality example of chapter 5

```
GEOMETRY
TITLE Criticality configuration chap 5 surface-based

SURF 1 PLANEZ 0.
SURF 2 PLANEZ 60.
SURF 3 PLANEZ 185.
SURF 4 PLANEZ 185.3
SURF 5 PLANEZ 213.84
SURF 6 PLANEZ 284.2
SURF 7 PLANEZ 284.8
SURF 8 PLANEZ 300.

SURF 9 CYLZ -15.1 0.  14.55
SURF 10 CYLZ -15.1 0.  14.85
SURF 11 CYLZ 15.1 0.  14.55
SURF 12 CYLZ 15.1 0.  14.85

SURF 13 PLANEX -150.
SURF 14 PLANEX 150.
SURF 15 PLANEY -150.
SURF 16 PLANEY 150.

VOLU 11 EQUA PLUS 3 2 13 15 MINUS 3 3 14 16 FICTIVE ENDV
VOLU 12 EQUA PLUS 3 7 13 15 MINUS 3 8 14 16 FICTIVE ENDV
VOLU 1 EQUA PLUS 5 10 12 3 13 15 MINUS 3 7 14 16
                                  UNION 2 11 12
ENDV


VOLU 2 EQUA PLUS 3 1 13 15 MINUS 3 2 14 16 ENDV

VOLU 31 EQUA PLUS 1 3 MINUS 2 4 10 FICTIVE ENDV
VOLU 32 EQUA PLUS 2 4 9 MINUS 2 6 10 FICTIVE ENDV
VOLU 3 EQUA PLUS 1 6 MINUS 2 10 7
                                  UNION 2 31 32
ENDV

VOLU 4 EQUA PLUS 1 5 MINUS 2 6 9 ENDV
VOLU 5 EQUA PLUS 1 4 MINUS 2 9 5 ENDV
```

```
VOLU 61 EQUA PLUS 1 3 MINUS 2 4 12 FICTIVE ENDV
VOLU 62 EQUA PLUS 2 4 11 MINUS 2 6 12 FICTIVE ENDV
VOLU 6 EQUA PLUS 1 6 MINUS 2 12 7
                                  UNION 2 61 62
ENDV

VOLU 7 EQUA PLUS 1 5 MINUS 2 6 11 ENDV
VOLU 8 EQUA PLUS 1 4 MINUS 2 11 5 ENDV

COMMENT
2D-view xz x from -160 to 160 and z from -10 to 310 in y=0 plane
no Postcript file generated
COMMENT
GRAPH
     -160.  -10.  0.
     1 0 0
     0 0 1
     320.  320.
     0

ENDGEOM
```

Note: by using the UNION operator, it is possible to combine 11, 12, and 13 into volume 1, 31, 32, and 33 into volume 3, and, finally, 61, 62, and 64 into volume 6. The volumes could have been left uncombined, but this is not only interesting for a surface flux or current (the surface is, indeed, defined as the boundary between the two volumes), but above all **the "unnecessary" union would result in a calculation time penalty**. In the present case, we retain the union in order to have the same numbers of volumes as in the combinatorial definition and to provide an example of a union.

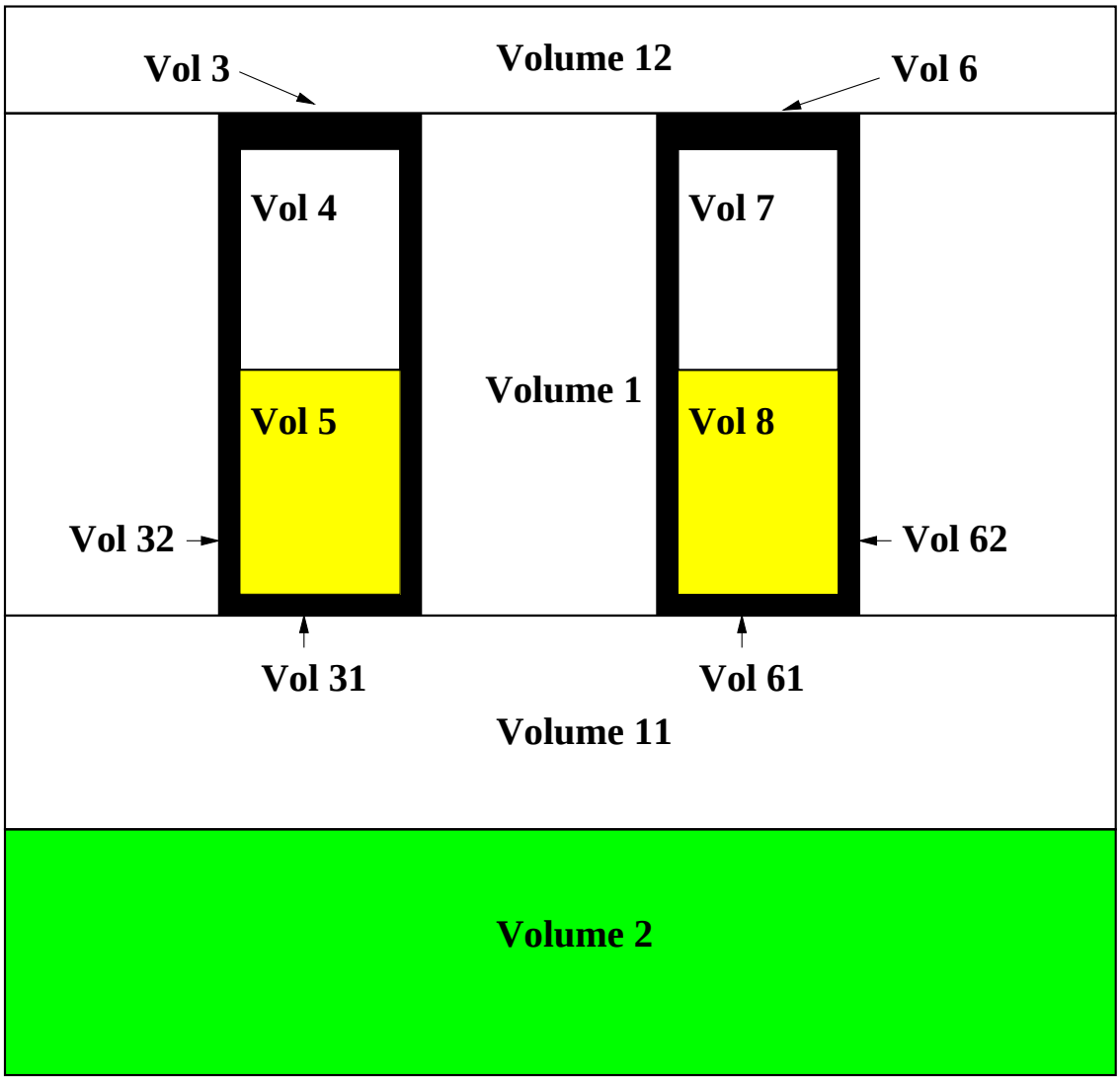



Figure 7.23: Surface-based geometry

## Combinatorial geometry definition for the criticality example of chapter 5

```
GEOMETRY
TITLE Criticality configuration chap 5 combinatorial

TYPE 1 BOX 300 300 300
TYPE 2 CYLZ 14.85 49.90
TYPE 3 CYLZ 14.55 35.18
TYPE 4 CYLZ 14.55 14.27
TYPE 5 BOX 300 300 60

COMMENT
z=0 bottom of the geometry
COMMENT
VOLU 1 COMBI 1 0 0 150 ENDV

COMMENT
concrete volume
COMMENT
VOLU 2 COMBI 5 0 0 30
     SMASH 1 1
ENDV

COMMENT
a container :  3 cylinders
COMMENT
VOLU 3 COMBI 2 -15.1 0 234.75
     SMASH 1 1
ENDV
VOLU 4 COMBI 3 -15.1 0 249.02
     SMASH 1 3
ENDV
VOLU 5 COMBI 4 -15.1 0 199.57
     SMASH 1 3
ENDV

COMMENT
the other container :  3 cylinders
COMMENT
VOLU 6 COMBI 2 15.1 0 234.75
     SMASH 1 1
ENDV
VOLU 7 COMBI 3 15.1 0 249.02
```

```
     SMASH 1 6
ENDV
VOLU 8 COMBI 4 15.1 0 199.57
     SMASH 1 6
ENDV

COMMENT
2D-view xz x from -160 to 160 and z from -10 to 310 in y=0 plane
no Postcript file generated
COMMENT
GRAPH
     -160.  -10.  0.
     1 0 0
     0 0 1
     320.  320.
     0

ENDGEOM
```

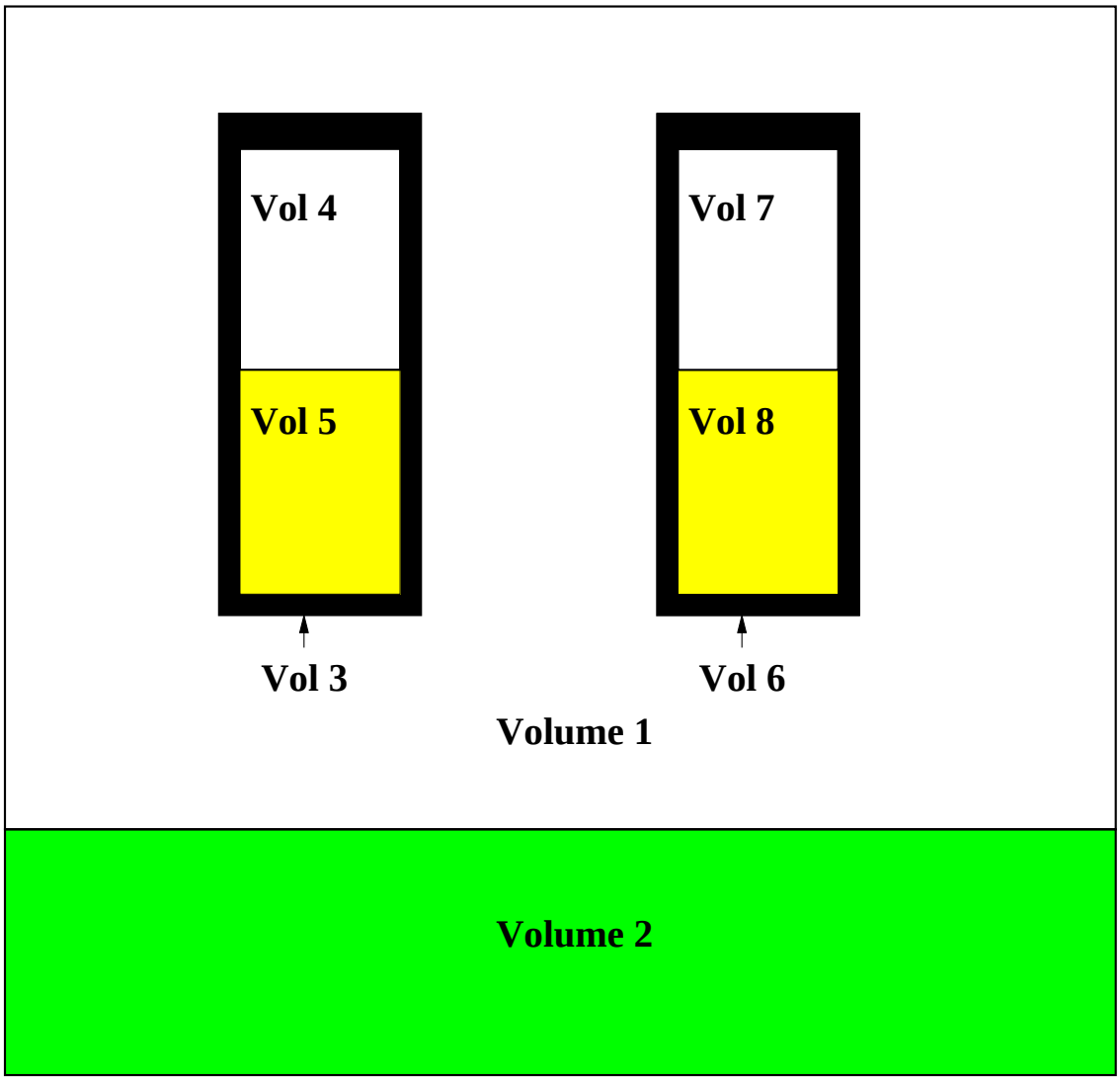



Figure 7.24: Combinatorial geometry

# Chapter 8

# THE VOLSURF COMMAND

For reaction rates and fluxes to be calculated per unit volume, it is necessary to do one of the following:

- request, using the CALCULATION option, that the volumes be assigned values calculated with the TRIPOLI-4® code ;
- specify the values of the volumes using the VOLSURF command.

For currents or fluxes to be calculated per unit area, they must be specified in VOLSURF (TRIPOLI-4® does not calculate areas). An area is determined using two volumes (see the conventions for SCORE in the corresponding section).

```
            VOLSURF
*               SURFACE  nb(surfaces)
                    numvol1(1) numvol2(1) surface(1)
                    ...
                    numvol1(n) numvol2(n) surface(n)
*               VOLUME  nb(volumes)
                    numvol(1) volume(1)
                    ...
                    numvol(n) volume(n)
            END_VOLSURF
```

where:

- nb(surfaces) is the number of surfaces for which the value in $cm^2$ is set (otherwise it is 1 $cm^2$ by default);
- $numvol_1$(i) $numvol_2$(i) surface(i) means that the value surface(i) is assigned to the surface defined by the volumes of numbers $numvol_1$(i) and $numvol_2$(i);
- nb(volumes) is the number of volumes for which the value is set in $cm^3$ (otherwise it is 1 $cm^3$ by default);

- `numvol(i) volume(i)` means that the value `volume(i)` is assigned to the volume numbered `numvol(i)`.

The user may enter only surfaces or may enter only volumes through the `VOLSURF` command.

# Chapter 9

# BOUNDARY CONDITIONS

## 9.1 DEFINING BOUNDARY CONDITIONS

The boundary conditions correspond to a specific sampling of the trajectories through the boundary sides of a volume in the outward direction. They are, therefore, given in pairs (volume, side or surface bounding the volume). Some conditions are not yet implemented (see Table 9.1).

The allowed types of boundary conditions are as follows:

### 9.1.1 LEAKAGE

Particles are abandoned after having passed through the surface (this is the default option for outer surfaces of the geometry).

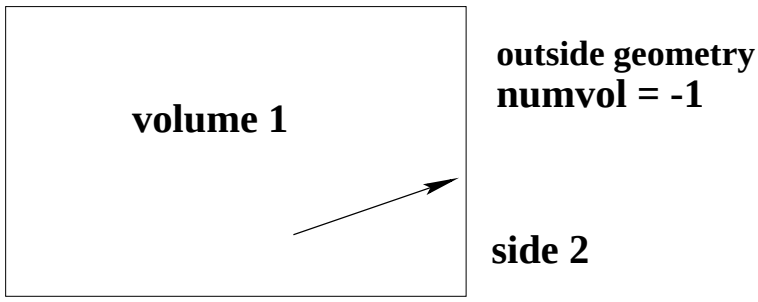



Figure 9.1: Leakage

By default, the particle is in volume 1 and intersects surface 2 (the geometrical boundary, which is indicated by the number -1 for the neighboring volume) then disappears.
In the particular case of a source resulting from a source file (see section 12.6), a leakage condition may need to be applied to a surface in the interior of the geometry. In such a case, this must be specified in the syntax of the boundary conditions for the surfaces involved.

### 9.1.2 REFLECTION

This corresponds to optical reflection on a surface of the volume (see Figure 9.2).

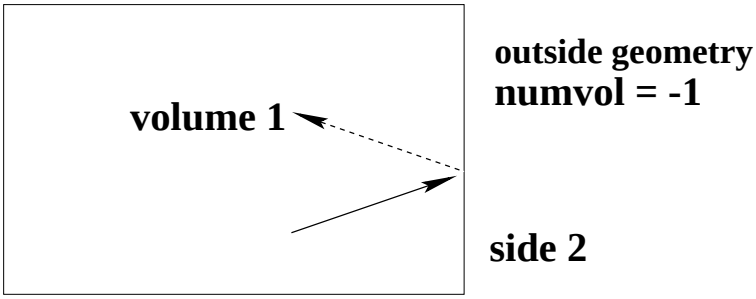



Figure 9.2: Reflection

### 9.1.3 ALBEDO

For all the surfaces on which the reflection boundary conditions can be defined, it is alternatively possible to attribute an albedo boundary condition. The kinematics of the albedo condition is exactly the same as for the reflection condition, i.e., it coincides with a mirror reflection on the surface of the volume as shown in Figure 9.2. However, an albedo value $0 < \beta < \infty$ is applied as a weight multiplier to each particle undergoing this kind of boundary condition. The special case of $\beta = 1$ corresponds to the standard reflection boundary condition. If $\beta < 1$ the particle weight after the boundary condition will be smaller, and for $\beta > 1$ it will be larger. A distinct albedo value $\beta$ can be assigned to each face of the volume. The albedo boundary condition needs thus the albedo value $\beta$ to be specified by the user for every surface where the condition is applied (see below).

### 9.1.4 TRANSLATION

This boundary condition is applicable to a combinatorial volume of one of the following shapes: `BOX`, `PARALLELEPIPEDE`, `HEXAX`, `HEXAY`, `HEXAZ`, `CYLX`, `CYLY`, `CYLZ`, or `CYL`.
Let $(M,\Omega)$ be the position and direction pair of a particle on side f of the box. The translation application transforms pair $(M,\Omega)$ into $(M',\Omega)$ where $M'$ is the point corresponding to $M$ on the side of the box opposite to f. It effectively corresponds to a translation of the constant vector $\mathbf{MM'}$ , which is parallel to an edge and of the same length.
For a cylinder, the situation is similar, except that one of the two cylinder faces must be used. The other face is then reached, with the translation being performed parallel to the axis of the cylinder.

### 9.1.5 ROTATION

This boundary condition should be applicable to any volume (combinatorial, surface based or ROOT). When a particle meet the conditions required for the boundary, it will be rotated according to the rotation elements

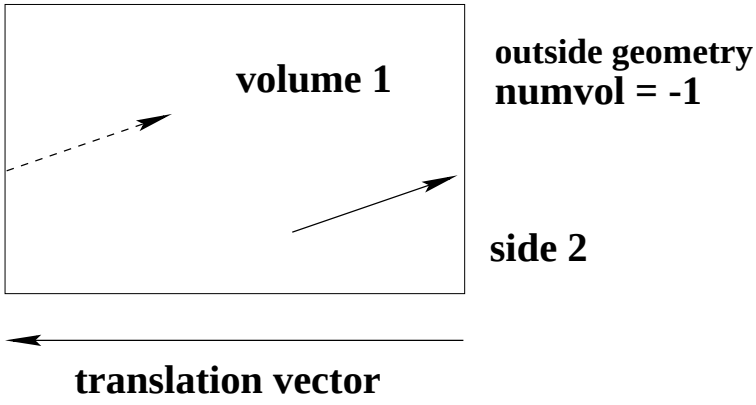



Figure 9.3: Translation

(axis and angle). Currently, only 90 degrees rotations are available. After the rotation, the particle must be inside the geometry but need not be inside the same volume.
The rotation boundary condition needing more specifications than usual boundary conditions follows a different sequence of instruction (see further section).

### 9.1.6 COSINE

This boundary condition will be used (with certain volumes) to simulate an isotropic incoming current in a volume (white boundary condition). For a particle leaving a volume at point $M$, with direction $\Omega$ via a side to which this boundary condition has been applied, the particle is re-emitted from the same point $M$, with a direction of $\Omega'$ The direction $\Omega'$ possesses an angle with respect to the normal inward vector with the cosine of the angle being computed with a uniform probability.

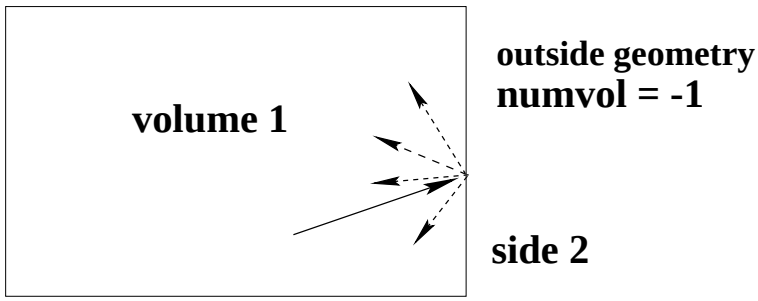



Figure 9.4: Cosine reflection

## 9.2 SEQUENCE OF INSTRUCTIONS

The boundary conditions are defined using the keyword BOUNDARY_CONDITION followed by the number of sides of the geometry that have boundary conditions. There is no need to specify a leakage condition for outer surfaces of the geometry since it is the default boundary condition. For each side, the number of the volume, the name of the boundary condition, and, finally, a number identifying the side are given. For albedo boundary conditions, the specific albedo value β must be provided before the number identifying the face on which the condition is imposed.

The boundary condition instruction sequence is as follows:

**BOUNDARY_CONDITION**
**nb(conditions)**
**# num(vol1) <TYPE_CONDITION> num(face1)**
**# num(vol1) ROTATION <rotation data>**
**# num(vol1) ALBEDO** $< \beta_1 >$ **num(face1)**
...
**# num(voln) <TYPE_CONDITION> num(facen)**
**# num(voln) ROTATION <rotation data>**
**# num(voln) ALBEDO** $< \beta_n >$ **num(facen)**
**END_BOUNDARY_CONDITION**

The possible values for <TYPE_CONDITION> are: REFLECTION, TRANSLATION, COSINUS, or LEAKAGE. The latter may be used when a source resulting from dump files is defined (see section 12.6.2).
For ALBEDO boundary conditions, the albedo value $\beta_i$ for each face $i$ must be given before the face number.

Supplemental data for rotation is given with the following syntax:

**num(voli) ROTATION** $X_c\ Y_c\ Z_c\ u_x\ u_y\ u_z\ \alpha$ **num(facei)**

The values $X_c$, $Y_c$, and $Z_c$ specify the location of any point of the axis. The values $u_x$, $u_y$, $u_z$ specify a vector defining the axis with the previous point. The α value is obviously the angle of the rotation. num(facei) specifies the face or surface from which particle are caught to be rotated.

**Important notes:**

* The boundary condition is oriented and applied only to particles that get out of a specified volume number through a specified surface number. In particular, this implies that a particle inside a volume is not affected by boundary conditions defined to immediate neighboring volumes.

* If the volume is defined by **equations** (surface-based geometry), the number of the side is the number of the surface given in the volume definition.

* If the volume is a **pure combinatorial** type volume (i.e., not altered by any other volume by means of operators), then the number of the side depends on a side numbering convention for the volumes of each shape. The numbering conventions are then:

- for volumes with one side (spheres and toroids) the side number is 1;
- for cones, the number of the base is 1 and that of the body is 2;
- for cylinders with an axis **u**, the lower base is number 1, the upper base is number 2, and the cylindrical side is number 3;
- for boxes, sides are numbered from 1 to 6 in the order -x, +x, -y, +y, -z, +z;
- for parallelepipeds, the sides are numbered from 1 to 6 in the order -u, +u, -v, +v, -w, +w, for **u**, **v**, and **w** the three axes of the parallelepiped;
- for hexagonal prisms, the sides are numbered from 1 to 8. The sides orthogonal to the axis of the prism are numbered 7 and 8 as axis ($X$, $Y$, or $Z$, depending on the case) increases. The other numbers are given in a diagram for a `HEXAZ`. For `HEXAX` and `HEXAY` figures, circular permutation of the axes is required.

It is possible to use the `POSITION` option in the geometry command in order to establish a surface number associated with a boundary point (the surface number is then given to the user).

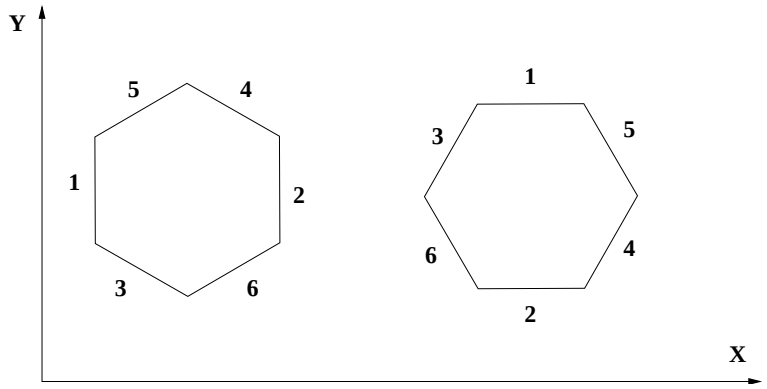



Figure 9.5: Numbering of sides for `HEXAZ` hexagons with angles $\theta = 0$ and $\theta = 90$

If the volume is a **modified combinatorial** volume (i.e., modified by operators), it only has the sides or surfaces of the pure volume, i.e., the volume unmodified by operators apart from rotation and lattice (`SMASH`, etc.).

Important: If the volume is a **lattice** type volume, it is now possible to specify side numbers. The combinatorial definition conventions are re-used.

| VOLUME | ESCAPE(default) | ESCAPE interior | REFLECTION or ALBEDO | TRANSLATION | COSINE |
|---|---|---|---|---|---|
| EQUA (pure) | yes | yes | yes | **no** | yes |
| TORUS[X,Y,Z ] | yes | **no** | yes | **no** | yes |
| COMBI (pure) | yes | yes | yes | **no** | yes |
| SPHERE | yes | yes | yes | **no** | yes |
| BOX | yes | yes | yes | yes | yes |
| PARALLELEPIPEDE | yes | yes | yes | yes | yes |
| HEXA[X,Y,Z ] | yes | yes | yes | yes | yes |
| CYL[X,Y,Z]<br>CYL | yes | yes | yes | yes (bases)<br>**no** (body) | yes |
| CONE | yes | yes | yes | **no** | yes |
| TORUS | yes | **no** | **no** | **no** | **no** |
| RESC (pure) | yes | **no** | yes | **no** | yes |
| RESH (pure) | yes | **no** | yes | **no** | yes |
| ROTATION (pure) | yes | **no** | **no** | **no** | **no** |

Table 9.1: Implemented boundary conditions

A surface can bound a large number of volumes. To avoid having to write a list of all the conditions for all the volumes defined using a surface-based geometry and bounded by this surface, it is possible to use the keyword ALL instead of the volume number. The code determines which volumes contain the surface and assigns the appropriate conditions to them.
Starting from version 10, it is possible to use the keyword ALL_COMPLETE instead of the volume number. The code applies the boundary condition to all the volumes including surface-based volumes, combinatorial type volumes, lattice type volumes and rotation type volumes, even if the surface is crossing the volume. In other words, with this keyword, the surface has the same properties than an additional surface.

## 9.3 ADDITIONAL BOUNDARY SURFACES

When a quarter or eighth of a geometry is to be defined (fuel assembly or reactor for instance), the surfaces at which reflections are to occur may not be natural surfaces if volumes are pure combinatorial types. To obtain the required volume, it would be necessary to use the VMINUS or SMASH operators. To avoid this inconvenience, it is now possible to apply boundary conditions on surfaces that are not part of the volume definition.
In such cases, the user must simply input the opposite of the number of the surface (the surface being defined in the geometry, such as surfaces determining a surface-based volume). The code recognizes a surface with a negative number as an additional surface.
A negative surface number can not be detected with ALL REFLECTION.
However, starting from version 10, this possibility is compatible with the keyword ALL_COMPLETE, see previous section.

**Criticality case of chapter 5**
In the case of the criticality calculation described in chapter 5, all the boundary conditions are leakage, and there is no BOUNDARY_CONDITION command.

# Chapter 10

# DEFINING COMPOSITIONS

Commands are introduced using the keyword `COMPOSITION`.
A composition in the TRIPOLI-4® sense is the cross-section type data used in particle transport for the composition in question, the temperature of the composition, and the name given to the composition by the user. The data necessary for each type of cross-section are then required.
The types of cross-sections currently available in TRIPOLI-4® are:

- pointwise cross-sections derived from evaluation in ENDF format;
- multigroup and homogenized cross-sections (currently the self-shielded cross-sections of APOLLO2 [95] in APOTRIM format and those calculated by TRIPOLI-4® after a homogenization process);
- multigroup cross-sections in GENDF format, created by the GROUPR module of the NJOY code.

If the TRIPOLI-4®-NJOY coupling option is specified when the TRIPOLI-4® code is run (option *-j*, see chapter 1), the temperatures may be any temperatures desired by the user, provided that the files exist for cross-sections at $0K$ temperatures exist. Otherwise, the temperatures may be any temperatures desired by the user, provided that the cross-section files exist at these temperatures (see section 10.4).

The names of the compositions are up to the user's discretion. The names of the isotopes comprising these compositions must be chosen from the names available in the dictionary file or, in the case of a natural element, must include the suffix -NAT. With this option, the existence of the nuclear data file for the natural element is checked. If this check fails, TRIPOLI-4® will use the isotopic abundance file (see chapter 3) to split the natural element into its naturally occurring isotopes. In case the user wants to work, exclusively, with the isotopic abundance file, he has to use -NAT-SPLIT instead.

The general syntax used to define compositions is as follows:

```
COMPOSITION
        nb(compositions)
*           FILE
                <ELECTRONIC_PROPERTIES>
            temperature name(compo)
            filename
*           POINT_WISE
                <ELECTRONIC_PROPERTIES>
            temperature name(compo)
            Composition data
          ...
*           DENSITY
                <ELECTRONIC_PROPERTIES>
            temperature name(compo)
            Composition data
          ...
*           MULTIGROUP_HOMOGENEOUS
            temperature name(compo)
            Composition data
          ...
*           MULTIGROUP
            temperature name(compo)
            Composition data
          ...
*           MULTIGROUP  DENSITY
            temperature name(compo)
            Composition data
          ...
END_COMPOSITION
```

where:

- `FILE` indicates that the definition for the material is located in the file named **filename**, which includes the path, if necessary. A file of this type contains a list of the materials in a format accepted by TRIPOLI-4®. The syntax of such a file corresponds to that of the `COMPOSITION` command; however, it does not use the initial and final keywords `COMPOSITION-END_COMPOSITION`, it does not indicate the number of `nb(compositions)`, and it does not call any compositions from another file;
- `POINT_WISE` is a keyword that initializes pointwise cross-sections in ENDF format by supplying the number of atoms per $cm^3$ (warning: this is true even if the lengths are given in another unit than cm);
- `DENSITY` is a keyword that initializes pointwise cross-sections in ENDF format by entering the density (in $g/cm^3$) and mass fractions or number of atoms;
- `MULTIGROUP_HOMOGENEOUS` is a keyword that initializes the homogenized multigroup cross-sections;

- MULTIGROUP codes the multigroup cross-sections in GENDF format by providing the number of atoms per $cm^3$;
- MULTIGROUP DENSITY is a keyword that initializes the multigroup cross-sections in GENDF format by entering the density (in $g/cm^3$) and the mass fractions;
- temperature is an integer (the temperature is expressed in Kelvin).
- <ELECTRONIC_PROPERTIES> is an optional keyword that defines electron and positron transport properties (see section 10.1)

For pointwise cross-sections, one can define compositions using both formats FILE, POINT_WISE and DENSITY. For multigroup cross-sections, the formats FILE, MULTIGROUP and MULTIGROUP DENSITY can be mixed. Nonetheless, it is not allowed to mix pointwise compositions with multigroup compositions.

## 10.1 POINTWISE CROSS-SECTIONS IN ENDF FORMAT

Compositions with pointwise cross-sections are identified using:

- the number of isotopes making up the composition;
- a list of the isotopes.

For each isotope, it is necessary to specify:

- the name of the isotope is given in the dictionary file or, in the case of a natural element, the name of the isotope is defined in the isotopic natural abundance file (see chapter 3) to which the suffix -NAT is appended (for example C-NAT for natural carbon). The user can force TRIPOLI-4® to work with the isotopic abundance file using -NAT-SPLIT instead of -NAT;
- the number of atoms per $10^{-24}$ $cm^3$ or the density.

All the isotopes with the same composition have the same temperature.

### 10.1.1 DEFINING COMPOSITIONS BY SUPPLYING THE NUMBER OF ATOMS PER $10^{-24}$ $CM^3$

The syntax for a composition of this type is as follows:

**POINT_WISE**
*<ELECTRONIC_PROPERTIES>*
**temperature**
**name(compo)**
**nb(isotopes)**
**isotope(1) nb-atom(1)**
...
**isotope(n) nb-atom(n)**

where:

- *<ELECTRONIC_PROPERTIES>* is an optional keyword that defines electronic transport properties, significant only in `PHOTON ELECTRON POSITRON` calculation mode. It overwrites the default choice of TRIPOLI-4®. The possible keywords are :
  - **GASNTP** for a gas at normal temperature and pressure conditions (0°C - 1 atm)
  - **CONDUCTOR** for an electric conductor
  - **NONPOLARIZABLE** for all other condensed materials (liquid or solid)
- `temperature` is the temperature in kelvin which must be an integer and entered as such without dots or commas (e.g., “300” not “300.”);
- `name(compo)` is the name chosen by the user for the composition to be defined;
- `nb(isotopes)` is the number of isotopes in the composition, followed by a list of the isotopes entered, each with its name `isotope(i)` in the library and the number of atoms `nb-atoms(i)` per $10^{-24}$ cm$^3$ in the composition.

Note: if the name `VOID` is used for a material, the use of a variance reduction technique will establish the value of $k$ at half the minimum of the values of $k$ of the other materials (see chapter 17 on variance reduction techniques).

### 10.1.2 DEFINING COMPOSITIONS BY ENTERING THE DENSITY (IN G/CM$^3$)

The syntax for a composition of this type is as follows:

```
DENSITY
        <ELECTRONIC_PROPERTIES>
        temperature
        name(compo)
        density
        nb(isotopes)
                isotope(1) frac-mass(1)
                ...
                isotope(n) frac-mass(n)
```

or

```
DENSITY
        <ELECTRONIC_PROPERTIES>
        temperature
        name(compo)
        density  NB_ATOM
        nb(isotopes)
                isotope(1) nb_molecular(1)
                ...
                isotope(n) nb_molecular(n)
```

where:

- *<ELECTRONIC_PROPERTIES>* is an optional keyword that defines electronic transport properties, significant only in `PHOTON ELECTRON POSITRON` calculation mode. It overwrites the default choice of TRIPOLI-4®. The possible keywords are :
  - **GASNTP** for a gas at normal temperature and pressure conditions (0°C - 1 atm)
  - **CONDUCTOR** for an electric conductor
  - **NONPOLARIZABLE** for all other condensed materials (liquid or solid)
- `temperature` is the temperature in kelvin which must be an integer and entered as such without dots or commas (e.g., “300” not “300.”);
- `name(compo)` is the name chosen by the user for the composition to be defined;
- `density` is the density;
- **NB_ATOM** allows the user to define the composition as a molecule.
- `nb(isotopes)` is the number of isotopes in the composition, followed by a list of the isotopes entered, each with its name `isotope(i)` in the library and their mass fraction `frac-mass(i)` or, in case of **NB_ATOM** keyword, the number of atoms (or isotopes) `nb_molecular(i)`.

Note: if the name `VOID` is used for a material, the use of a variance reduction technique will establish the value of $k$ at half the minimum of the values of $k$ of the other materials (see chapter 17 on variance reduction techniques).

## 10.2 MULTIGROUP HOMOGENIZED CROSS-SECTIONS

Multigroup homogenized cross-sections can currently only be self-shielded cross-sections derived from APOLLO2 in APOTRIM format and those of TRIPOLI-4® (created using the HOMOGENIZE command).

### 10.2.1 SELF-SHIELDED HOMOGENIZED MULTIGROUP CROSS-SECTIONS FROM APOLLO2 AND TRIPOLI-4®

APOTRIM format compositions are defined using:

- burnup;
- the name of the library containing the self-shielded homogenized cross-sections.

The composition must be specified in an APOTRIM format cross-section file with the same name as that of the TRIPOLI-4® data file, at the same temperature and burnup. Syntax is, therefore, shown in the box below.

```
MULTIGROUP_HOMOGENEOUS
        temperature
        name(compo)  APOLLO2
                burnup
                file(APOLLO)
```

where:

- temperature is the temperature of the composition, expressed in degrees Kelvin. The temperature is an integer and must be entered as such (for example, “300” rather than “300.”). It must correspond to the temperature given in the cross-section file, which is given in degrees Celsius (which must be a relative integer: the difference between the two values must be 273);
- name(compo) is the name of the composition in the file;
- <TYPE_SECTION> equals APOLLO2 or T4HOMOT4;
- burnup is the burnup and must correspond to the value read from the APOTRIM file. This value must be an integer (this value is 0 when TRIPOLI-4® cross-sections are used);
- file is the name of the APOLLO or TRIPOLI-4® cross-section file. If a file is not found in the directory from which the executable is run, it is necessary to add the path. It is recommended that absolute path be used (e.g., */home/user/directory/file*).

Note: The maximum anisotropy order taken into account by TRIPOLI-4® in APOTRIM format files is order 3 in the Legendre polynomial expansion (P3). The method is based on the Nonequally Probable Step Function Representation which is described in reference [77].

## 10.3 MULTIGROUP CROSS-SECTIONS IN GENDF FORMAT

The syntax for a multigroup composition is as follows:

```
MULTIGROUP
        temperature
        name(compo)
        nb(isotopes)
                isotope(1) nb-atome(1)
                ...
                isotope(n) nb-atome(n)
```

or

```
MULTIGROUP  DENSITY
        temperature
        name(compo)
        density
        nb(isotopes)
                isotope(1) frac-mass(1)
                ...
                isotope(n) frac-mass(n)
```

For an explanation of the fields in the above syntax, refer to the pointwise cross-sections case, section 10.1.
Use of multigroup cross-sections requires one of the following command line options:
**-s GENDF** or **-s TABPROB_GENDF** (see chapter 1).

## 10.4 TRIPOLI-4®-NJOY COUPLING

If a cross-section at temperature newtemp is needed and cannot be found in the available libraries or in the directory containing the user-defined cross sections (see chapter 3), TRIPOLI-4® can create the corresponding files automatically. To accomplish this the user must use the NJOY code (version 99), which is called directly by TRIPOLI-4® using the execution option -j (see chapter 1). The input deck provided to TRIPOLI-4® may be a simplified one, used specifically to generate the cross-section files, or else it may be the input deck for the envisioned simulation. TRIPOLI-4® first calls NJOY (BROADR at newtemp K, errthn = 0.001, using reconstructed files at 0K, i.e., using the provided bcd.pendf.0) then TRIPOLI-4® calls its own routines. The created files are put into the directory specified by the user in the path file (see chapter 3) and may be reused in the future.
Once the file at the temperature newtemp has been created in the user repertory, the option -j is no longer needed for subsequent simulations at this temperature. It should be noted that only pointwise neutron cross-sections are created by the option -j. Hence:

- the probability table files produced by the CALENDF code and the multigroup files generated by the GROUPR module of NJOY are not created by the option -j. All calculations using the new temperatures will then be performed without probability tables, even if the option TABPROP is specified.
- The files for bonded elements are not taken into account. Calling a temperature for a bonded element that is not present in the files will result in an error. In order to avoid this, the user can create a link to the file with the closest temperature to the bonded element in question.
- This option should exclusively be run in mono-processor mode. If any run in parallel should accidentally occur, the created cross-section files must be cleaned and the run should be executed again in mono-processor mode.

**Note:** instructions for compiling the NJOY code are provided in the installation files of the TRIPOLI-4® code.

## Defining compositions for the criticality calculation example in chapter 5 with pointwise cross-sections

```
COMPOSITION 4
            POINT_WISE 300 INOX_STEEL
            13
               FE54 3.6805E-3 FE56 5.6188E-2 FE57 1.2943E-3
               FE58 1.7176E-4 CR50 7.4596E-4 CR52 1.3832E-2
               CR53 1.5397E-3 CR54 3.7545E-4 NI58 5.6052E-3
               NI60 2.0697E-3 NI61 8.7556E-5 NI62 2.7564E-4
               NI64 6.8909E-5
            POINT_WISE 300 AIR
            2
               N14 1.900E-5 O16 9.7457E-6
            POINT_WISE 300 SOLUTION_U
            11
               U234 2.5554E-6 U235 2.8247E-4 U236 8.7585E-7
               U238 2.7877E-5 FE54 5.6745E-8 FE56 8.6631E-7
               FE57 1.9955E-8 FE58 2.6481E-9 N14 1.8650E-3
               O16 3.6262E-2 H1 6.1314E-2
            POINT_WISE 300 CONCRETE
            6
               H1 1.3740E-2 O16 4.5908E-2 AL27 1.7380E-3
               SI 1.6608E-2 CA 1.4989E-3 NA23 2.7780E-3
END_COMPOSITION
```

## Defining compositions for the criticality calculation example using multigroup homogenized cross-sections

```
COMPOSITION 4
            MULTIGROUP_HOMOGENEOUS 300 INOX_STEEL APOLLO2
                                   0
                                   /home/am/APOLLO2/resultats/bi.ur2cyl86
            MULTIGROUP_HOMOGENEOUS 300 AIR APOLLO2
                                   0
                                   /home/am/APOLLO2/resultats/bi.ur2cyl86
            MULTIGROUP_HOMOGENEOUS 300 SOLUTION_U APOLLO2
                                   0
                                   /home/am/APOLLO2/resultats/bi.ur2cyl86
            MULTIGROUP_HOMOGENEOUS 300 CONCRETE APOLLO2
                                   0
```

```
                                   /home/am/APOLLO2/resultats/bi.ur2cyl86
END_COMPOSITION
```

## Defining compositions for the criticality calculation example by calling a file

```
COMPOSITION 4
            FILE 300 INOX_STEEL
                 /home/am/catalogue
            FILE 300 AIR
                 /home/am/catalogue
            FILE 300 SOLUTION_U
                 /home/am/catalogue
            FILE 300 CONCRETE
                 /home/am/catalogue
END_COMPOSITION
```

with, for example, the following **/home/am/catalogue** file:

```
POINT_WISE 300 INOX_STEEL
13
   FE54 3.6805E-3 FE56 5.6188E-2 FE57 1.2943E-3
   FE58 1.7176E-4 CR50 7.4596E-4 CR52 1.3832E-2
   CR53 1.5397E-3 CR54 3.7545E-4 NI58 5.6052E-3
   NI60 2.0697E-3 NI61 8.7556E-5 NI62 2.7564E-4
   NI64 6.8909E-5
POINT_WISE 300 AIR
2
   N14 1.900E-5 O16 9.7457E-6
POINT_WISE 300 SOLUTION_U
11
   U234 2.5554E-6 U235 2.8247E-4 U236 8.7585E-7
   U238 2.7877E-5 FE54 5.6745E-8 FE56 8.6631E-7
   FE57 1.9955E-8 FE58 2.6481E-9 N14 1.8650E-3
   O16 3.6262E-2 H1 6.1314E-2
MULTIGROUP_HOMOGENEOUS 300 CONCRETE APOLLO2
   0
   /home/am/APOLLO2/resultats/bi.ur2cyl86
```

# Chapter 11

# DEFINING VOLUME-COMPOSITION ASSOCIATIONS

Volume-composition associations are defined using the keyword `GEOMCOMP`.

This stage of the definition is very simple, but two rules must be followed:

- all the volumes must be associated with a composition, except for virtual (fictive) and lattice volumes;
- no volume must be associated with more than one composition.

Volume-composition associations are defined as follows:

```
GEOMCOMP
        name-compo(1)
                nb-volu(1)
                volu(1)...volu[nb-volu(1)]
        ...
        name-compo(n)
                nb-volu(n)
                volu(1)...volu[nb-volu(n)]
END_GEOMCOMP
```

It is also possible to associate one composition to all the volumes of the geometry. In this case, the volume-composition association is defined as follows:

```
GEOMCOMP
        name-compo
                ALL_VOLUMES
END_GEOMCOMP
```

## Defining volume-composition associations for the criticality calculation example

```
GEOMCOMP
         INOX_STEEL 2 3 6
         SOLUTION_U 2 5 8
         CONCRETE   1 2
         AIR        3 1 4 7
END_GEOMCOMP
```

# Chapter 12

# DEFINING SOURCES

Calculation sources are defined between the keywords `SOURCES_LIST` and `END_SOURCES_LIST`.

The number of sources `nb(sources)` is then given, and each source is defined independently.
The structure is as follows:

```
SOURCES_LIST
    nb(sources)
     NORM  norm(list)
          SOURCE
               Description of first source
          END_SOURCE
          ...
          SOURCE
               Description of last source
          END_SOURCE
END_SOURCES_LIST
```

Adding `NORM norm(list)` indicates the value of the norm for the entire system of `nb(sources)` sources. If this instruction is not used, the norm equals $N_{\text{list}} = \sum_i N_i$ where $N_i$ is the norm (calculated or imposed) of each source.

## 12.1 DEFINING A SOURCE

TRIPOLI-4® sources are factorized as follows:

$$S(\mathbf{r},\Omega,E,t) = C \cdot S(\mathbf{r}) \cdot S(\Omega) \cdot S(E) \cdot S(t)$$

An exception to this scheme is the external source, see paragraph 12.8.
For each source, the spatial $S(\mathbf{r})$, angular $S(\Omega)$, energy $S(E)$, and time $S(t)$ distributions are defined independently. It is also possible to enter a normalization factor $N_i$ for each source $i$, such that

$$\int\int\int\int S_i(\mathbf{r},\Omega,E,t)d\mathbf{r}d\Omega dEdt = N_i.$$

If a global norm $N_{\text{list}}$ has been defined for the system, then the norms for each sources are relative ones. As such, the norm of the source is:

$$N_{\text{source}} = \frac{N_i}{\sum_i N_i} N_{\text{list}}.$$

The syntax used to define a source is as follows:

```
      SOURCE
#           INTENSITY  norm(source)
#           COEFF  coeff(source)
            <PARTICLE_TYPE>
                  Description of spatial distribution: POINT or FACTORIZED
            ANGULAR_DISTRIBUTION
                  Description of angular distribution
            ENERGETIC_DISTRIBUTION
                  Description of energy distribution
            TIME_DISTRIBUTION
                  Description of time distribution
      END_SOURCE
```

where <PARTICLE_TYPE> equals NEUTRON, PHOTON, ELECTRON, or POSITRON. By default, the norm of the source is the product of

$$N_{\text{source}} = I(\text{geometrical}) \cdot I(\text{angular}) \cdot I(\text{energy}) \cdot I(\text{time})$$

where:

- the geometrical intensity $I$(geometrical) is the integral of the spatial distribution $S(\mathbf{r})$ over the source domain definition (intersection of volumes and the definition domain):

$$I(\text{geometrical}) = \int_{V(source)} S(\mathbf{r})d\mathbf{r};$$

- angular intensity $I$(angular) is the integral of the angular distribution $S(\Omega)$:

$$I(\text{angular}) = \int_{4\pi} S(\Omega)d\Omega;$$

- energy intensity $I$(energy) is the integral of energy distribution $S(E)$:

$$I(\text{energy}) = \int S(E)dE;$$

- time intensity $I$(time) is the integral of time distribution $S(t)$:

$$I(\text{time}) = \int S(t)dt.$$

It is possible to impose a normalisation on a source (for source particles) using the keyword `INTENSITY` followed by the norm. The user must then specify the norms of all the sources to ensure a consistent calculation. It is also possible to modify the norm calculated by the code by a coefficient using the keyword `COEFF`. `INTENSITY` and `COEFF` cannot be used simultaneously.

## 12.2 SPATIAL DISTRIBUTION

The source can be:

- a point source (a Dirac distribution);
- a tabulated spatial distribution source (discrete values);
- an analytical spatial distribution source (continuous values).
- a surface spatial distribution source (discrete values).

In the first case, the definition is extremely simple, in the other three it is necessary to specify an associated reference frame.

### 12.2.1 POINT SOURCES

For a point source, it is sufficient to give the coordinates $\mathbf{r}_0 = (X,Y,Z)$ of the source.
Syntax is as follows:

**POINT X Y Z**

Note: The source must not be placed on a boundary of the volume.
There are no other geometrical data (no `VOLUME` to be specified nor `GEOMETRIC_DISTRIBUTION` to be entered).
The geometrical intensity is 1 in this case, the distribution $S(\mathbf{r})$ being a Dirac $\delta(\mathbf{r}-\mathbf{r}_0)$.

### 12.2.2 DISTRIBUTED SOURCES

A factorized distributed source is introduced using the keyword `FACTORIZED`.

#### 12.2.2.1 Definition context

A source is defined (with the exception of the point source case covered above) in a context created by a list of volumes and a definition domain. The spatial extension of the source is the intersection of the volumes and the domain. A particle is, therefore, emitted if it belongs to the definition domain and to a volume on the list. An option `ALL_VOLUMES_INSIDE_MESH` allows any volume inside the domain to be an allowed emitter; this allows for easy flat source definition in criticality calculations.
The domain is defined in a reference frame specific to each source, characterized by:

- an origin $O'$;
- three vectors defining an orthonormal reference frame: **i**, **j**, **k**;
- a reference frame associated with the vectors.

The reference frame can be **Cartesian**: a vector **O'M** is defined by its coordinates $(x,y,z)$, projections of **O'M** along axes **i**, **j**, **k**:

$$\mathbf{O'M} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}.$$

It can also be **cylindrical**, in which case the vector **O'M** is defined by its coordinates $(r,\theta,z)$ where $r$ is the normal of the projection **O'M** on the plane (**i**, **j**), $\theta$ is the angle between the projection (between 0 and $2\pi$) and the vector **i**, and $z$ is the projection of **O'M** along axis **k**:

$$\mathbf{O'M} = r\cos\theta\mathbf{i} + r\sin\theta\mathbf{j} + z\mathbf{k}.$$

Finally, the reference frame can be **spherical**, in which case the vector **O'M** is defined by its coordinates $(r,\phi,\theta)$ where $r$ is the normal of **O'M**, $\phi$ is the angle (between 0 and $2\pi$) between the projection of **O'M** on the plane (**i**, **j**) and the vector **i**, and $\theta$ is the angle between OM and the axis (**i**, **k**) (between 0 and $\pi$):

$$\mathbf{O'M} = r\cos\phi\sin\theta\mathbf{i} + r\sin\phi\sin\theta\mathbf{j} + r\cos\theta\mathbf{k}.$$

See figures 4.1, 4.2 and 4.3.
Note: when a non-unit vector is entered to specify the reference frame, TRIPOLI-4® automatically renormalizes it to 1.

#### 12.2.2.2 Syntax for a factorized source

The reference frame definition is introduced using the keyword `FRAME`, followed by the type of system chosen (`CARTESIAN`, `CYLINDER`, or `SPHERE`), the coordinates of the origin of O', and the vectors defining the axes. It continues with either the keyword `VOLUME` (resp `COMPOSITION`) followed by the number of volumes (resp number of compositions) and a list of volumes (resp list of compositions), or by the keyword `ALL_VOLUMES_INSIDE_MESH`, or by the keyword `ALL_FISSILE_VOLUMES_INSIDE_MESH`. The definition of the spatial distribution is then given along with associated description domain.
It can be followed by the keyword `OPENMP` and the number of processors intended for the calculation using shared memory parallelism. This feature has a very specific usage. It can not currently be mixed with the usual parallel mode of TRIPOLI-4. It can be used to accelerate the calculation of the norm of the factorized

sources in mono-processor mode. The remaining part of the calculation will be in mono-processor. Using shared memory parallelism, this option can only use the cores available on the launching node.

Syntax is as follows for a distributed source:

**FACTORIZED**
**FRAME**
**<FRAME_TYPE>**
**O'$_x$ O'$_y$ O'$_z$**
**i$_x$ i$_y$ i$_z$**
**(j$_x$ j$_y$ j$_z$)**
**k$_x$ k$_y$ k$_z$**
* **VOLUME**
**nb(volumes) volume(1)...volume(n)**
* **COMPOSITION**
**nb(compositions) composition(1)...composition(n)**
* **ALL_VOLUMES_INSIDE_MESH**
* **ALL_FISSILE_VOLUMES_INSIDE_MESH**
***OPENMP nb_procs***
**GEOMETRIC_DISTRIBUTION**
Description of geometrical distribution:
***NO_GEOMSOURCE_WARNING***
***SOBOL_POINTS nb(points)***
TABULATED or ANALYTICAL or TRIANGLE

where <FRAME_TYPE> equals CARTESIAN, CYLINDER, or SPHERE. The line j$_x$ j$_y$ j$_z$ is only required if CARTESIAN is chosen.
Note:
Starting from TRIPOLI-4® version 9, points are sampled uniformly in the Physical (Euclidean) space for all types of frames; previously they were distributed uniformly in the space of parameters **U**, **V**, **W**.
It is possible to use a **virtual** number for the source volume. In this case, the source is the intersection of the virtual volume and the domain. This makes it possible to avoid defining a geometry only for the purpose of defining sources when the domain cannot be given analytically. **Extreme care** is required, however, with this property, since virtual volume definition errors are detected neither in the running of the code nor in the display. Manipulation of the operators must be avoided.

The user must, however, check the volume of the source (volume-domain intersection) using the geometrical intensity, given in the output.

Important special application:
In the case of a lattice, it should be noted that:

- a volume contained in a reference cell is repeated: the numbers of the volumes of the reference cell are used for all the repeated cells of the lattice. It is not possible to distinguish a particular cell, including the reference cell;
- the lattice number **can be used**: this number then corresponds to all the volumes making up the lattice. This is because the lattice case is similar to the virtual volume case. It does, however, include the benefits of tests and a display.

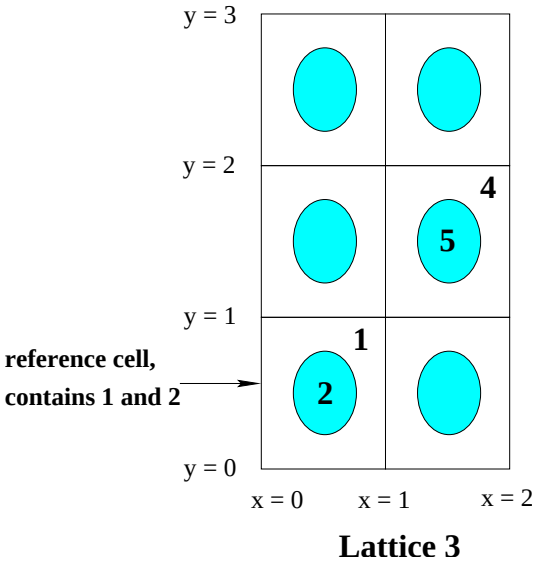



Figure 12.1: Source in a lattice

Example: drawing 12.1 corresponds to the sequence

```
TYPE 1 BOX 2 1 1
VOLU 1 COMBI 1 1 0.5 0 ENDV

TYPE 2 SPHERE 0.4
VOLU 2 COMBI 2 1 0.5 0 SMASH 1 1 ENDV

VOLU 3 RESC VOLU 1 2 3 1
ENDV
```

The volume is defined as the intersection of the domain $(0 < x < 1, 0 < y < 2, -0.5 < z < 0.5)$ and one of the following:

- volumes 1 and 2 (`VOLU 2 1 2` ), as they are repeated in the lattice. If the sphere were to be the only emitting body, this would be replaced by `VOLU 1 2`
- lattice 3 (`VOLU 1 3` ), which in this case represents all the volumes of a lattice

#### 12.2.2.3 Surface, analytical and tabulated distributions

Once the reference frame has been specified, a set of coordinates $(u,v,w)$ is available, in either Cartesian $(x,y,z)$, cylindrical $(r,\theta,z)$, or spherical $(r,\phi,\theta)$ form. A distribution can be defined by tabulated values, by an analytical function or with surface sources.
The optional keyword `SOBOL_POINTS` is used to refine the calculation of the geometric distribution. The default value for the number of points (produced from a sobol generator) used in this calculation is 50000 for the three different sources.
When the intersection between the volumes and the domain of the source is small with respect to the source domain, the code will generate a warning message each time more than 100 consecutive sampled source points are not in the intersection. It is possible to deactivate these messages using the `NO_GEOMSOURCE_WARNING` command.
This may result from an inconsistency between the source domain and the volumes, but it may also result from a very fragmented source structure. In the latter case, it may prove advantageous to enhance the source estimation precision using the command `SOBOL_POINTS`.

**Tabulated distribution.** This is introduced using the keyword `TABULATED`.

A tabulated distribution is a distribution in discrete form in a mesh that describes the definition domain of the source.
The number of neutrons or photons emitted per unit volume in the cell is associated with each cell.

The user must indicate the discretization for each component $(u,v,w)$.

Tabulation can be specified in three different manners:

- the data for intensity $I$(cell) is factorized over three components $I_u$, $I_v$, $I_w$.
  $I(\text{cell}) = I_u(u_{\text{centre}}) \times I_v(v_{\text{centre}}) \times I_w(w_{\text{centre}}) \times \text{Volume}_{\text{emitter}}(\text{cell})$;
- the data for intensity on a component and a plane
  $I(\text{cell}) = I_u(u_{\text{centre}}) \times I_{v,w}(v_{\text{centre}}, w_{\text{centre}})\text{Volume}_{\text{emitter}}(\text{cell})$;
- the data for intensity in space (to be given for each cell)
  $I(\text{cell}) = I_{u,v,w}(u_{\text{centre}}, v_{\text{centre}}, w_{\text{centre}}) \times \text{Volume}_{\text{emitter}}(\text{cell})$.

The data for $I_u \times I_v \times I_w$ , $I_u \times I_{v,w}$, or $I_{u,v,w}$ is in particles per cm$^3$ (and in steradians per MeV per second).
$\text{Volume}_{\text{emitter}}(\text{cell})$ is the intersection volume of the cell and the source volumes.

The intersection of the cell and the source volumes, therefore, homogeneously emits particles with a total intensity of $I$(cell). The non-source part of the cell does not contribute to the emission.

In Figure 12.2, cells $a1$, $b1$, $c1$, $a2$, $b2$, $c2$, $a3$, $b3$, $c3$ emit particles. Only in cell $b2$ are particles emitted at all points (for simplification, the third component is discarded), the other cells only emit particles in their shaded areas. Cells $d1$,$d2$, and $d3$ do not emit any particles.

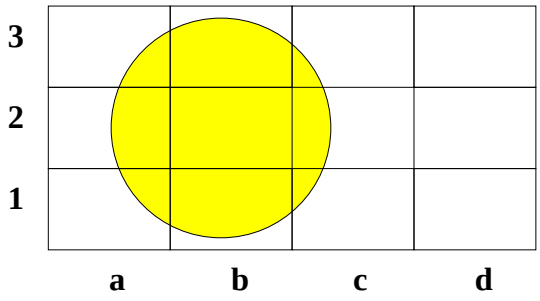



Figure 12.2: The volume is shown in yellow, and the grid represents the domain

The coordinate system can be Cartesian, cylindrical, or spherical in the tabulated case.

**A) Intensity data given on the three components**
This possibility is selected using the keyword: `TYPE F_U_V_W` .
Important: It must be borne in mind that `nb(u)` coordinates along the `u` axis means `nb(u)-1` intervals, hence `nb(u)-1` intensity values along the axis. The same applies to the other coordinates.

**GEOMETRIC_DISTRIBUTION**
***SOBOL_POINTS nb(points)***
**TABULATED**
**TYPE F_U_V_W**
**VAR_U <NAME_U>**
**nb[coord(u)]**
$\mathbf{u}_1 \ldots \mathbf{u}_{nb(u)}$
**VAR_V <NAME_V>**
**nb[coord(v)]**
$\mathbf{v}_1 \ldots \mathbf{v}_{nb(v)}$
**VAR_W <NAME_W>**
**nb[coord(w)]**
$\mathbf{w}_1 \ldots \mathbf{w}_{nb(w)}$
**F_U** $I_u(1) \ldots I_u(nb(u)-1)$
**F_V** $I_v(1) \ldots I_v(nb(v)-1)$
**F_W** $I_w(1) \ldots I_w(nb(w)-1)$

Where `<NAME_U>`, `<NAME_V>` and `<NAME_W>` are equal to `X`, `Y`, or `Z` in Cartesian coordinates, to `R`, `RTHETA`, or `RPHI` in Spherical coordinates, or to `R`, `RTHETA`, or `Z` in Cylindrical coordinates .
The value of the source distribution $S(\mathbf{r})$ in a cell $(i, j, k)$ is given by $I_u(i) \times I_v(j) \times I_w(k)$. When the source distribution is integrated over the cell volume, it is weighted by the portion of volumes selected for source emission (see the definition of `FACTORIZED`, page 142).

Example with Cartesian coordinates:

```
GEOMETRIC_DISTRIBUTION
TABULATED
          TYPE F_U_V_W
               VAR_U X
                     3
                     1 2 3
               VAR_V Y
                     3
                     1.5 3.5 4.5
               VAR_W Z
                     2
                     0 10
               F_U 1 0.5
               F_V 0.8 0.3
               F_W 1
```

**B) Intensity data given for 1 component and 1 plane**

This possibility is selected using the keyword: `TYPE F_U_VW`.

**GEOMETRIC_DISTRIBUTION**
***SOBOL_POINTS nb(points)***
**TABULATED**
**TYPE F_U_VW**
**VAR_U <NAME_U>**
**nb[coord(u)]**
$\mathbf{u}_1 \ldots \mathbf{u}_{nb(u)}$
**VAR_V <NAME_V>**
**nb[coord(v)]**
$\mathbf{v}_1 \ldots \mathbf{v}_{nb(v)}$
**VAR_W <NAME_W>**
**nb[coord(w)]**
$\mathbf{w}_1 \ldots \mathbf{w}_{nb(w)}$
**F_U** $I_u(1) \ldots I_u(nb(u)-1)$
**F_VW** $I_{vw}(1,1) \ldots I_{vw}(nb(v)-1, nb(w)-1)$

Where `<NAME_U>`, `<NAME_V>`, and `<NAME_W>` are equal to `X`, `Y` or `Z` in Cartesian coordinates, to `R`, `RTHETA`, or `RPHI` in Spherical coordinates, or to `R`, `RTHETA`, or `Z` in Cylindrical coordinates .

The value of the source distribution $S(\mathbf{r})$ in a cell $(i,j,k)$ is given by $I_u(i) \times I_{vw}(j,k)$. When the source distribution is integrated over the cell volume, it is weighted by the portion of volumes selected for source emission (see the definition of `FACTORIZED`, page 142).

This is similar to the F_U_V_W case, but instead of nb(v)-1 values of F_V and nb(w)-1 values of F_W, the [nb(v)-1]×[nb(w)-1] values of F_VW are given. The values F_VW(v,w) are to be entered in the following order (with constant W and V varied, see Figure 12.3):
F_VW(1,1) F_VW(2,1)...F_VW(nb(v)-1,1) then
F_VW(1,2) F_VW(2,2)...F_VW(nb(v)-1,2) then
...
F_VW(1,nb(w)-1) F_VW(2,nb(w)-1)...F_VW(nb(v)-1,nb(w)-1)

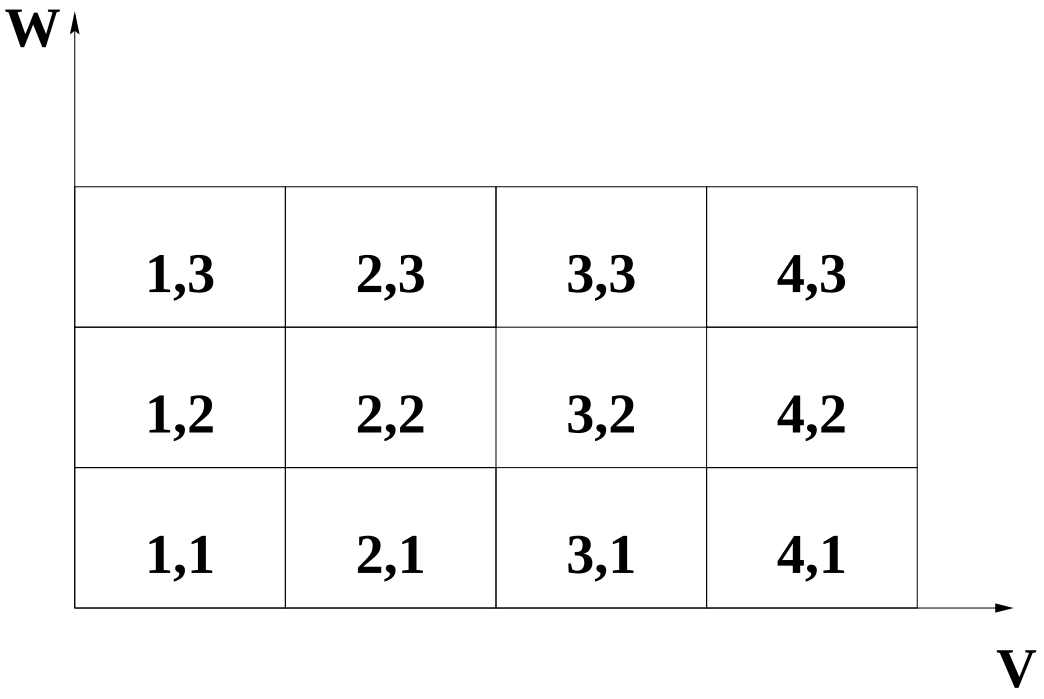



Figure 12.3: The data are entered from left to right and from bottom to top: (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), etc.

**C) Intensity data given per cell**

This possibility is selected using the keyword: `TYPE F_UVW`.

**GEOMETRIC_DISTRIBUTION**
***SOBOL_POINTS nb(points)***
**TABULATED**
**TYPE F_UVW**
**VAR_U <NAME_U>**
**nb[coord(u)]**
$\mathbf{u}_1 \ldots \mathbf{u}_{nb(u)}$
**VAR_V <NAME_V>**
**nb[coord(v)]**
$\mathbf{v}_1 \ldots \mathbf{v}_{nb(v)}$
**VAR_W <NAME_W>**
**nb[coord(w)]**
$\mathbf{w}_1 \ldots \mathbf{w}_{nb(w)}$
**F_UVW** $I_{uvw}(1,1,1) \ldots I_{uvw}(nb(u)-1, nb(v)-1, nb(w)-1)$

Where `<NAME_U>` , `<NAME_V>` , `<NAME_W>` are equal to `X` , `Y`, or `Z` in Cartesian coordinates, to `R`, `RTHETA`, or `RPHI` in Spherical coordinates, or to `R`, `RTHETA`, or `Z` in Cylindrical coordinates .

The value of the source distribution $S(\mathbf{r})$ in a cell $(i, j, k)$ is given by $I_{uvw}(i, j, k)$. When the source distribution is integrated over the cell volume, it is weighted by the portion of volumes selected for source emission (see the definition of `FACTORIZED`, page 142).

Once again, this is similar to the `F_U_V_W` case, but instead of `nb(u)-1` values of `F_U`, `nb(v)-1` values of F_V and `nb(w)-1` values of `F_W`, the

`[nb(u)-1]`×`[nb(v)-1]`×`[nb(w)-1]`

values of `F_UVW` are given. The values `F_UVW(u,v,w)` are to be entered in the following order (see Figures 12.4 and 12.5):

```
F_UVW(1,1,1) F_VW(2,1,1)...F_VW(nb(u)-1,1,1)
F_UVW(1,2,1) F_VW(2,2,1)...F_VW(nb(u)-1,2,1)
...
F_UVW(1,nb(v)-1,1) F_VW(2,nb(v)-1,1)...F_VW(nb(u)-1,nb(v)-1,1)
```

then

```
F_UVW(1,1,2) F_VW(2,1,2)...F_VW(nb(u)-1,1,2)
F_UVW(1,2,2) F_VW(2,2,2)...F_VW(nb(u)-1,2,2)
...
F_UVW(1,nb(v)-1,2) F_VW(2,nb(v)-1,2)...F_VW(nb(u)-1,nb(v)-1,2)
```

then...

```
F_UVW(1,1,nb(w)-1) F_VW(2,1,nb(w)-1)...F_VW(nb(u)-1,1,nb(w)-1)
F_UVW(1,2,1) F_VW(2,2,1)...F_VW(nb(u)-1,2,1)
```

...

```
F_UVW(1,nb(v)-1,nb(w)-1)
F_VW(2,nb(v)-1,nb(w)-1)...F_VW(nb(u)-1,nb(v)-1,nb(w)-1)
```

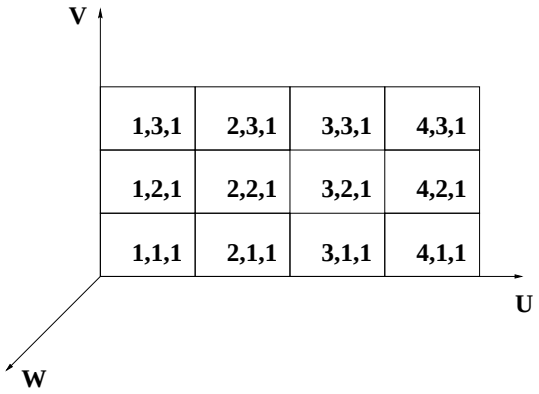



Figure 12.4: First from left to right and from bottom to top (first sweep along **U**, line by line). Top view

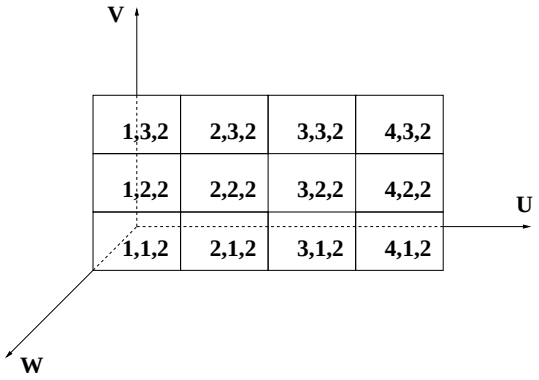



Figure 12.5: Then from left to right and from bottom to top (first sweep along **U**, line by line). Top view

**Helpers for syntax entry.** In order to simplify the data entry in the sources and thus to simplify the consistency between the mesh of sources and the mesh of the KIJ matrix feature (optional), two keywords has been added: `UPTO_LIN` and `REPEAT`.
The use of "*x* `UPTO_LIN` *n y*" replaces in input the part "`UPTO_LIN` *n*" by *n* values linearly interpolated between *x*, excluded and *y* included. It leads, thus, to $n+1$ values regularly disposed between *x* and *y* included.
The use of "*x* `REPEAT` *n*" replaces in input the part "`REPEAT` *n*" by *n* values of *x*, thus leading to $n+1$ times the value *x*.
For example, we could express a source with a uniform density on a grid of 11x11x9 (=1089) cells between [0, 275] in *x*, [0, 275] in *y*, and [0, 100] in *z* using the following syntax:

```
TYPE F_UVW
```

```
VAR_U X 12 0 UPTO_LIN 11 275
VAR_V Y 12 0 UPTO_LIN 11 275
VAR_W Z 10 0. UPTO_LIN 9 100.
F_UVW  1. REPEAT 1088
```

**Analytical distribution.** An analytical distribution is a continuous distribution in the source definition domain. This distribution $S(\mathbf{r})$ is defined by an algebraic formula and a definition domain for the formula. The syntax is as shown in the box, below

```
GEOMETRIC_DISTRIBUTION
    ANALYTICAL
        FUNCTION = ... ;
        DOMAIN = ... ;
```

and using the following variable names

- in a Cartesian reference frame: `X` , `Y` , `Z`;
- in a cylindrical reference frame: `R`, `RTHETA`, `Z` (`RTHETA` must be between 0 and $2\pi$);
- in a spherical reference frame: `R`, `RPHI`, `RTHETA` (`RPHI` must be between 0 and $2\pi$ and `RTHETA` between 0 and $\pi$).

Notes:
The code does not check the consistency of the definition domain with the algebraic formula. This task is left to the user. In each reference frame, formulas and domains are expressed using variables.
The source distribution $S(\mathbf{r})$ is given, not its volume-integrated value.
The default number of points used in the geometrical source distribution is now 50000, whereas it was 5000 in version 4.4 (and earlier versions) of TRIPOLI-4®. The processing of the factorized source initialization for an analytical distribution is thus more precise, but may also be relatively slow. For comparisons with earlier versions, set the `SOBOL_POINTS` value to 5000. See in subsection 12.2.2 how to change the `SOBOL_POINTS` default value.

Important: If an expression begins with a $-$ sign, a 0 must be placed before it: `EXP(0-X)`, for $\exp(-x)$ for instance.

The allowed operators in the formula expression are (`LOG` is the natural logarithm):
`+` , `-` , `*`, `/`, `SH`, `CH`, `SIN`, `COS`, `EXP`, `LOG`, `SQRT`, `()`.
The relationships accepted in domains definitions are: $<$ and $>$.
Inputs are separated by commas.
Examples
Source in $\exp[-(x+y)]$ (Cartesian coordinates):

```
GEOMETRIC_DISTRIBUTION
    ANALYTICAL
        FUNCTION= EXP(0-(X+Y));
        DOMAIN=0<X<1,0<Y<2,0<Z<1;
```

Source in $\exp(-r)$ (cylindrical coordinates):

```
GEOMETRIC_DISTRIBUTION
     ANALYTICAL
          FUNCTION= EXP(0-R);
          DOMAIN=0<R<1,0<RTHETA<6.28,0<Z<1;
```

Source in $\exp(-r)$ (spherical coordinates):

```
GEOMETRIC_DISTRIBUTION
     ANALYTICAL
          FUNCTION= EXP(0-R);
          DOMAIN=0<R<1,0<RPHI<6.28,0<RTHETA<3.14;
```

Note: Care must be taken not to omit the semi-colons after the FUNCTION and DOMAIN commands.

**Surface distribution.** A surface distribution can be defined thanks to a tessalation process. The idea consists in building a surface source by splitting this source into (small) triangles. Therefore, to declare a surface source, a file with an arbitrary name (called here surface_source_file.txt) has to be present together with TRIPOLI-4 input file in the current directory. This surface file is given here:

```
t4 triangular mesh file
version 1.0
NUMBER_OF_TRIANGLES 2
0.  30.  30.  0.  -30.  30.  0.  -30.  -30.
0.  30.  30.  0.  30.  -30.  0.  -30.  -30.
```

where the two first lines contain arbitrary comments, the third line contains the keyword NUMBER_OF_TRIANGLES followed by the number of triangles used to describe the source (here 2) and all the other lines contain the triangles definitions thanks to the 9 coordinates of their angles. Those coordinates are the one of the three vertices, each one being described by its position in (x,y,z). In the previous example, if one consider the first triangle, its first vertex is located in (x=0.,y=30.,z=30.), its second vertex is located in (x=0.,y=-30.,z=30.) and its last vertex is located in (x=0.,y=-30.,z=-30.). Fig. 12.6 illustrates this example. Once this surface source file defined, the syntax used in the TRIPOLI-4 input file is (with the SOBOL_POINTS value set to 5000):

```
GEOMETRIC_DISTRIBUTION
     SOBOL_POINTS 5000
     TRIANGLE
     FILE surfacic_source_file.txt
```

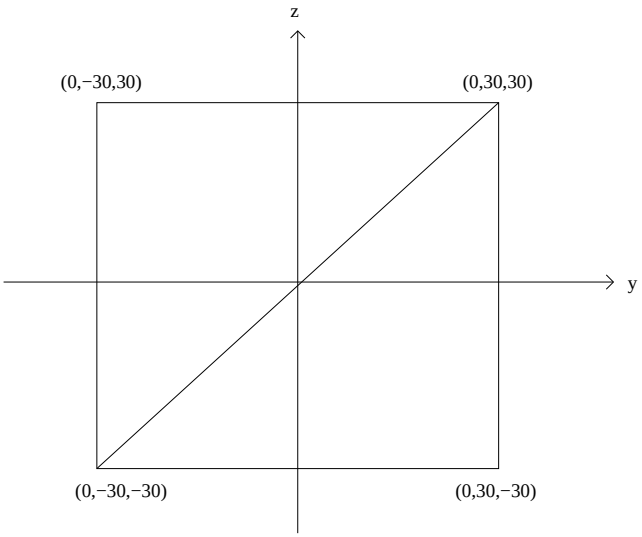



Figure 12.6: Square source made by two triangles

Note: The intensity is uniform over all triangles of a same source. To declare triangles with different intensities, different sources have to be declared.

## 12.3 ANGULAR DISTRIBUTION

There are five different types of angular distribution (Dirac, isotropic, rays, tabulated and analytical). An explanation of variables and angles definitions for the different types of frame can be found in section 4.5.

### 12.3.1 DIRAC

This is the single emission direction expressed in the form of a vector of coordinates $(\omega_x, \omega_y, \omega_z)$ given in the geometrical reference frame in the case of a point source, or given in the source frame in the case of a spatially distributed source. The syntax is as follows:

**ANGULAR_DISTRIBUTION**
**MONO_DIR**
$\omega_x$ $\omega_y$ $\omega_z$

The distribution $S(\Omega)$ is a Dirac distribution, whose integral over the solid angle (angular intensity) is equal to 1.

### 12.3.2 UNIFORM DISTRIBUTION

Important: The distribution $S(\Omega)$ is a constant such that the angular intensity equals $4\pi$.
This corresponds to an isotropic emission, written as follows:

```
ANGULAR_DISTRIBUTION
ISOTROPIC
```

### 12.3.3 RAY DISTRIBUTION

This option can be used to define a source distributed on a list of discrete angles. It is introduced using `DISCRETE_PROBABILITY`.

#### 12.3.3.1 Discrete $\mu$

The source is defined as a function of $\mu = cos(\theta)$, where the variable $\theta$ is the angle relative to axis $Oz$:

- for a point source: axis $OZ$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **k** (see figure 4.3).

The syntax is as follows :

```
ANGULAR_DISTRIBUTION
    DISCRETE_PROBABILITY
        DISCRETE_MU
        nb(angles)
            μ0 I0
            ...
            μnb(angles)−1 Inb(angles)−1
```

The angular distribution is the sum of Dirac distributions whose total intensity is 1.
$S(\mu) = \sum_{i=0}^{nb} I_i \delta(\mu - \mu_i) / \sum_{i=0}^{nb} I_i$

#### 12.3.3.2 Discrete $\phi$

The source is defined as a function of $\phi$. $\phi$ is the angle relative to axis $Ox$:

- for a point source: axis $OX$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **i** (see figure 4.3).

The syntax is as follows :

**ANGULAR_DISTRIBUTION**
**DISCRETE_PROBABILITY**
**DISCRETE_PHI**
**nb(angles)**
$\phi_0$ $\mathbf{I}_0$
...
$\phi_{nb(angles)-1}$ $\mathbf{I}_{nb(angles)-1}$

The angular distribution is the sum of Dirac distributions whose total intensity is 1.
$S(\phi) = \sum_{i=0}^{nb} I_i \delta(\phi - \phi_i) / \sum_{i=0}^{nb} I_i$

### 12.3.4 TABULATED DISTRIBUTIONS

This is introduced using (`TABULATED` or) `MU_TABULATED` or `PHI_TABULATED`.

#### 12.3.4.1 $\mu$ tabulated distribution

Tabulation provides the discretized distribution over an interval of $\mu = cos(\theta)$. Here, the variable $\theta$ is the angle with respect to axis $Oz$:

- for a point source: axis $OZ$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **k** (see figure 4.3).

#### 12.3.4.2 Syntax

The principle of definition is identical to that for the input of a tabulated response function, as described in Section 13.4, option `DATA`, containing a precise definition. The variable $\mu = \cos\theta$ replaces the energy, and the angular distribution values replace the cross-sections. The syntax is, therefore, as follows (the $\mu$ must be given in increasing order) :

**ANGULAR_DISTRIBUTION**
**MU_TABULATED**
***SOURCE_WEIGHTING***
**nb(points) nb(inter)**
**<MODE_INTERPOLATION(1)> num[limsup(1)]**
**...**
**<MODE_INTERPOLATION(nbinter)> num[limsup(nbinter)]**
$\mu_0$ **value**$_0$ $F_0$
**...**
$\mu_{nb(points)-1}$ **value**$_{nb(points)-1}$ $F_{nb(points)-1}$

Notes:

- Starting from TRIPOLI-4v8, new keyword `MU_TABULATED` is available allowing the user to be explicit about the chosen angle. However, keyword `TABULATED` is still available (designating $\mu$ angle).
- The keyword `SOURCE_WEIGHTING` is optional and only available for the interpolation mode `IN_NO_INTERPOLATION` (histogram). It allows for source angle biasing: the sampling of certain $\mu$ angle ranges of the tabulated distribution can be enhanced in this way. When this option is requested, $\mathtt{F}_x$ parameters have to be input at the end of the present command block. Note that the biased distribution can then be checked by requesting a `FIXED_SOURCE_EMISSION` score (see chapters 13 and 15). Standard variance reduction techniques (see chapter 17) are not compatible with the `SOURCE_WEIGHTING` option, whereas source energy biasing (see subsection 12.4.7) and source angle biasing can be requested together.
- $\mathbf{F}_x$ values have to be given only when the keyword `SOURCE_WEIGHTING` is present in the command block. These values are multiplicative factors weighting the natural probabilities of the angle distribution. For example a factor 10 makes the sampling of the given angle range 10 times more probable than with the natural probability. These factors have to be defined for each angle range `x` of the tabulated distribution and are 1 by default if the keyword `SOURCE_WEIGHTING` is not used. To deal with the resulting lower weights of certain particles, the Russian roulette threshold of the simulation may be lowered (see chapter 22, an order of magnitude being $0.8/\mathrm{F}_{max}$ if the biggest factor defined $\mathrm{F}_{max}$ is bigger than 1).
- `<MODE_INTERPOLATION>` belongs to the list {`IN_NO_INTERPOLATION`, `IN_LIN_LIN`, `IN_LIN_LOG`, `IN_LOG_LIN`, `IN_LOG_LOG`};
- `nb(points)` is the number of points, and `nb(inter)` the number of different interpolation ranges;
- Mode `<MODE_INTERPOLATION(1)>` applies from $\mu_0$ to $\mu_{num[limsup(1)]}$;
- Mode `<MODE_INTERPOLATION(j)>` applies from $\mu_{num[limsup(j-1)]}$ to $\mu_{num[limsup(j)]}$ where $num[limsup(i)]$ is an integer, the index to the point in the list of given ($\mu_i$,value$_i$) pairs pairs (Caution: the numbering begins at 0).

- The value of the last couple ($\mu_{nb(points)-1}$, **value**$_{nb(points)-1}$) is untapped when <MODE_INTERPOLATION> is IN_NO_INTERPOLATION.

It is the angular distribution $S(\Omega)$ which is tabulated, and not the angular intensity.

#### 12.3.4.3 $\phi$ tabulated distribution

Tabulation provides the discretized distribution over an interval of $\phi$. Here, the variable $\phi$ is the angle with respect to axis $Ox$:

- for a point source: axis $OX$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **i** (see figure 4.3).

#### 12.3.4.4 Syntax

The principle of definition is identical to that for the input of a tabulated response function, as described in Section 13.4, option DATA, containing a precise definition. The variable $\phi$ replaces the energy, and the angular distribution values replace the cross-sections. The syntax is, therefore, as follows (the $\phi$ must be given in increasing order).

**ANGULAR_DISTRIBUTION**
**PHI_TABULATED**
***SOURCE_WEIGHTING***
**nb(points) nb(inter)**
**<MODE_INTERPOLATION(1)> num[limsup(1)]**
**...**
**<MODE_INTERPOLATION(nbinter)> num[limsup(nbinter)]**
$\phi_0$ **value**$_0$ $F_0$
**...**
$\phi_{nb(points)-1}$ **value**$_{nb(points)-1}$ $F_{nb(points)-1}$

Notes:

- The keyword SOURCE_WEIGHTING is optional and only available for the interpolation mode IN_NO_INTERPOLATION (histogram). It allows for source angle biasing (the meaning is similar as for the previous $\mu$ tabulated distribution).
- $\mathbf{F}_x$ values have to be given only when the keyword SOURCE_WEIGHTING is present in the command block and their meaning is similar as for the previous $\mu$ tabulated distribution).
- <MODE_INTERPOLATION> belongs to the list {IN_NO_INTERPOLATION, IN_LIN_LIN, IN_LIN_LOG, IN_LOG_LIN, IN_LOG_LOG};

- `nb(points)` is the number of points, and `nb(inter)` the number of different interpolation ranges;
- Mode `<MODE_INTERPOLATION(1)>` applies from $\phi_0$ to $\phi_{num[limsup(1)]}$;
- Mode `<MODE_INTERPOLATION(j)>` applies from $\phi_{num[limsup(j-1)]}$ to $\phi_{num[limsup(j)]}$ where $num[limsup(i)]$ is an integer, the index to the point in the list of given ($\phi_i$,value$_i$) pairs pairs (Caution: the numbering begins at 0).
- The value of the last couple ($\phi_{nb(points)-1}$, **value**$_{nb(points)-1}$) is untapped when `<MODE_INTERPOLATION>` is `IN_NO_INTERPOLATION`.

It is the angular distribution $S(\Omega)$ which is tabulated, and not the angular intensity.

### 12.3.5 2D TABULATED DISTRIBUTION

This is introduced using `2D_TABULATED`. 2D Tabulation provides the discretized distribution over a grid of $\mu = cos(\theta)$ and of $\phi$.

Here, the variable $\theta$ is the angle with respect to axis $Oz$:

- for a point source: axis $OZ$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **k** (see figure 4.3).

The variable $\phi$ is the angle with respect to axis $Ox$:

- for a point source: axis $OX$ of the geometrical reference frame;
- for a spatially distributed source, the axis carried by **i** (see figure 4.3).

#### 12.3.5.1 Syntax

The syntax is as follows :

**ANGULAR_DISTRIBUTION**
**2D_TABULATED**
$\mathbf{nb}_{(bounds(\mu\ grid))}$ $\mathbf{nb}_{(bounds(\phi\ grid))}$
$\mathbf{bound}_{(1(\mu\ grid))} \dots \mathbf{bound}_{(n(\mu\ grid))}$
$\mathbf{bound}_{(1(\phi\ grid))} \dots \mathbf{bound}_{(n(\phi\ grid))}$
$\mathbf{value}_{(\mu_0,\phi_0)}$ $\mathbf{value}_{(\mu_0,\phi_1)} \dots$ $\mathbf{value}_{(\mu_0,\phi_{(nb(bounds(\phi\ grid))-1)})}$
$\mathbf{value}_{(\mu_1,\phi_0)}$ $\mathbf{value}_{(\mu_1,\phi_1)} \dots$ $\mathbf{value}_{(\mu_1,\phi_{(nb(bounds(\phi\ grid))-1)})}$
...
$\mathbf{value}_{(\mu_{(nb(bounds(\mu\ grid))-1)},\phi_0)}$ $\mathbf{value}_{(\mu_{(nb(bounds(\mu\ grid))-1)},\phi_1)} \dots$ $\mathbf{value}_{(\mu_{(nb(bounds(\mu\ grid))-1)},\phi_{(nb(bounds(\phi\ grid))-1)})}$

Note:

- **value**($\mu_x$,$\phi_x$) is the angular distribution corresponding to the cell ($\mu_x$,$\phi_x$).

### 12.3.6 ANALYTICAL DISTRIBUTION

The distribution $S(\Omega)$ is expressed using an algebraic formula described in a manner similar to that used for geometrical distribution. In each reference frame, the formulas and domains are expressed using variables: `PHI` and `THETA` (or `MU` = cos(`THETA` )).
These variables correspond to the angles $\phi$ (between 0 and $2\pi$) and $\theta$ (between 0 and $\pi$) defined for a spherical reference frame, whatever the reference frame chosen. The axes are those of reference frame $OX, OY, OZ$ in the case of a point source. Furthermore:
`PHI` is the angle with respect to $Ox$:

- for a point source: axis $OX$ of the geometrical reference frame;
- for a spatially distributed source: the axis carried by **i** (see reference frame figures)

`THETA` is the angle with respect to $Oz$:

- for a point source: axis $OZ$ of the geometrical reference frame;
- for a spatially distributed source: the axis carried by **k** (see reference frame figures).

Notes:
In the case of variables $\theta$ and $\phi$, the integral of the sources is calculated as:

$$I(\text{angular}) = \int_{\phi=0}^{2\pi}\int_{\theta=0}^{\pi} S(\theta,\phi)d\theta d\phi$$

where $S(\theta,\phi)$ is given by the user.
An isotropic distribution is, therefore, defined by a function $S(\theta,\phi) = \sin\theta$, and not $S(\theta,\phi) = 1$.

In the case of variables $\mu = \cos\theta$ and $\phi$, the integral of the sources is calculated as:

$$I(\text{angular}) = \int_{\phi=0}^{2\pi}\int_{\mu=-1}^{1} S(\mu,\phi)d\mu d\phi$$

Here the isotropic distribution is defined by a function $S(\mu,\phi) = 1$.

A distinction must be made between `PHI` and `THETA` on the one hand and `RPHI` and `RTHETA` on the other. The latter are only used when the definition reference frame is spherical. In the case of a cylindrical reference frame, the user must distinguish between the cylindrical definition of angle $\theta$ for spatial distribution (`RTHETA`) and the spherical definition for angular distribution (`THETA` ): **these are not the same angles**.

The operators in the formula expression are the following (`LOG` is the natural logarithm):
+ , -, * , /, `SH`, `CH`, `SIN`, `COS`, `EXP`, `LOG`, `SQRT`, `()`. The acceptable relationships in domain definitions are: $<$, $>$.

Important: If an expression begins with a $-$ sign, a 0 must be placed before it: `EXP(0-X)`, for $\exp(-x)$ for instance.

**ANGULAR_DISTRIBUTION**
**ANALYTICAL**
**FUNCTION = . . . ;**
**DOMAIN = . . . ;**

Example (source in sin(θ), ϕ between 0 and 2π, θ between 0 and π):

```
ANGULAR_DISTRIBUTION
    ANALYTICAL
        FUNCTION= SIN(THETA);
        DOMAIN=0<PHI<6.28,0<THETA<3.14;
```

## 12.4 ENERGY DISTRIBUTION

Distributions can be expressed in seven different ways (Dirac, line spectrum, band emission, Gauss Spectrum, Watt spectrum, Maxwell spectrum, tabulated spectrum, and analytical spectrum).

### 12.4.1 DIRAC

This corresponds to a mono-kinetic source, given by a real number. The syntax is as follows:

```
ENERGETIC_DISTRIBUTION
    SPECTRUM
        MONOKINETIC
            E0
```

The energy distribution is a Dirac $S(E) = \delta(E - E_0)$, whose intensity is 1.

### 12.4.2 LINE SPECTRUM

This corresponds to a list of Dirac distributions and expressed using a series of real number pairs (Energy-Intensity). The syntax is as follows:

```
ENERGETIC_DISTRIBUTION
    SPECTRUM   RAY_SPECTRUM
        nb(of lines)
            E0 I0
            ...
            E_nb(lines)-1 I_nb(lines)-1
```

The energy distribution is the sum of Dirac distributions.
$S(E) = \sum_{i=0}^{nb} I_i \delta(E - E_i)$

### 12.4.3 UNIFORM DISTRIBUTION

This corresponds to a uniform energy emission distribution between two energy values that are specified by the user. The syntax is as follows:

```
ENERGETIC_DISTRIBUTION
    SPECTRUM
        BAND
            Esup Einf
```

The energy distribution is $S(E) = U(E_{sup}, E_{inf})$, such that $S(E) = 1$ if the energy is between the bounds, and $S(E) = 0$ otherwise. The intensity, therefore, equals $E_{sup} - E_{inf}$.

### 12.4.4 GAUSS SPECTRUM

This corresponds to a Gauss energy emission distribution at the energy $E_{mean}$ with a standard deviation $\sigma$. The energy distribution is $S(E) = \mathcal{N}(E_{mean}, \sigma)$. $\mathcal{N}$ is the normal distribution centered on $E_{mean}$ with the standard deviation $\sigma$. $E_{mean}$ and $\sigma$ are specified by the user using the following directives:

$$\mathcal{N}(E_{mean}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(E-E_{mean})^2}{2\sigma^2}} = \frac{1}{\sqrt{\pi}a} e^{-\frac{(E-E_{mean})^2}{a^2}}$$

The syntax is as follows:

```
ENERGETIC_DISTRIBUTION
    SPECTRUM
        GAUSS_SPECTRUM
            a Emean
```

### 12.4.5 WATT SPECTRUM

This corresponds to a distribution expressed in the form of a Watt spectrum, i.e.,

$$S(E) = C \cdot e^{-aE} \mathrm{sh}\sqrt{bE}$$

where $C = 2e^{-\frac{b}{4a}} \frac{a^{\frac{3}{2}}}{\sqrt{\pi b}}$.

```
ENERGETIC_DISTRIBUTION
    SPECTRUM
        WATT_SPECTRUM
            PARAM  Esup Einf a b
```

It is possible to input the a and b coefficients (both expressed in $MeV^{-1}$) of the spectrum (warning: it is common to find "$\frac{1}{a}$" in other codes instead of the definition of "$a$"), and the energy range for the particle emission. For further information on the subject refer to [1].

By default:
Esup=20 MeV ,
Einf =1.E-11 MeV,
a=1.02 $MeV^{-1}$ and
b=2.286 $MeV^{-1}$.
This corresponds to a fast $^{235}$U spectrum.

Other common examples of choice for a and b parameters are as follows:
a=1.012 $MeV^{-1}$ and b=2.249 $MeV^{-1}$ for a thermal $^{235}$U spectrum
a=1.035 $MeV^{-1}$ and b=2.842 $MeV^{-1}$ for a thermal $^{239}$Pu spectrum
a=1.02 $MeV^{-1}$ and b=2.12 $MeV^{-1}$ for a $^{238}$U spectrum
a=1.02 $MeV^{-1}$ and b=2.328 $MeV^{-1}$ for a $^{233}$U spectrum
a=1.02 $MeV^{-1}$ and b=1.496 $MeV^{-1}$ for a $^{232}$Th spectrum.

It should be noted that the option of *fission spectrum from the evaluation* may also be a way to define a Watt spectrum (see subsection 12.4.9).

### 12.4.6 MAXWELL SPECTRUM

This corresponds to a distribution expressed as a Maxwell spectrum between energy bounds Esup and Einf.

$$S(E) = C \cdot e^{-\frac{E}{\theta}} \sqrt{E}$$

where $C = 1$

It is necessary to use $\theta = kT$, with θ being the temperature of the spectrum translated into MeV. The energy bounds are Esup and Einf.

**ENERGETIC_DISTRIBUTION**
**SPECTRUM**
**MAXWELL_SPECTRUM**
**θ $E_{sup}$ $E_{inf}$**

### 12.4.7 TABULATED SPECTRUM

This corresponds to a distribution whose values are given at a certain number of points and interpolated between them. The syntax used to define such a spectrum corresponds to that used for tabular angular distri-

bution (based on the input of a response function with DATA, as described in full in Chapter 13). Distribution is given as a function of energy (**by order of increasing energy**).

**ENERGETIC_DISTRIBUTION**
**SPECTRUM INTERPOLATED**
***SOURCE_WEIGHTING***
***GROUP_PROBABILITY***
**nb(points) nb(inter)**
**<MODE_INTERPOLATION(1)>**
**num[limsup(1)]**
**...**
**<MODE_INTERPOLATION(nbinter)>**
**num[limsup(nbinter)]**
$\mathbf{E}_0$ **value**$_0$ $F_0$
**...**
$\mathbf{E}_{nb(points)-1}$ **value**$_{nb(points)-1}$ $F_{nb(points)-1}$

- the keyword SOURCE_WEIGHTING is optional and only available for the interpolation mode IN_NO_INTERPOLATION (histogram). It allows for source energy biasing: the sampling of certain energy groups of the tabulated distribution can be enhanced in this way. When this option is requested, $F_x$ parameters have to be input at the end of the present command block. Note that the biased distribution can then be checked by requesting a FIXED_SOURCE_EMISSION score (see chapters 13 and 15). Standard variance reduction techniques (see chapter 17) are not compatible with the SOURCE_WEIGHTING option, whereas source energy biasing and source angle biasing (see subsection 12.3.4) can be requested together.

- the keyword GROUP_PROBABILITY is optional and only available for the interpolation mode IN_NO_INTERPOLATION (histogram). It is used in order to enter directly the values in particles/s. SOURCE_WEIGHTING and GROUP_PROBABILITY options can be used together.

- $\mathbf{F}_x$ values have to be given only when the keyword SOURCE_WEIGHTING is present in the command block. These values are multiplicative factors weighting the natural probabilities of the energy distribution. For example a factor 10 makes the sampling of the given group 10 times more probable than with the natural probability. These factors have to be defined for each energy group x of the tabulated distribution and are 1 by default if the keyword SOURCE_WEIGHTING is not used. To deal with the resulting lower weights of certain particles, the Russian roulette threshold of the simulation may be lowered (see chapter 22, an order of magnitude being $0.8/\mathrm{F}_{max}$ if the biggest factor defined $\mathrm{F}_{max}$ is bigger than 1).

- <MODE_INTERPOLATION> belongs to list {IN_NO_INTERPOLATION, IN_LIN_LIN, IN_LIN_LOG, IN_LOG_LIN, IN_LOG_LOG};

- nb(points) is the number of points, and nb(inter) the number of different interpolation ranges;

- Mode <MODE_INTERPOLATION(1)> applies to $E_0$ at $E_{num[limsup(1)]}$;
- Mode <MODE_INTERPOLATION(j)> applies to $E_{num[limsup(j-1)]}$ at $E_{num[limsup(j)]}$ where $num[limsup(i)]$ is the index in the list of given ($E_i$,value$_i$) pairs. (Caution: numbering begins at 0).
- The value of the last couple ($\mathbf{E}_{nb(points)-1}$, $\mathbf{value}_{nb(points)-1}$) is untapped when <MODE_INTERPOLATION> is IN_NO_INTERPOLATION.

If the keyword GROUP_PROBABILITY is not used, the tabulated values must be expressed in particles/MeV/s.
Warning: The units conventions for TRIPOLI-3 correspond to use of the keyword GROUP_PROBABILITY.

If the user wants to use a spectrum such as $S(E) \propto \frac{1}{E}$, the most accurate way is to define a tabulated source with two points and the IN_LOG_LOG interpolation.

Example (source in $\frac{1.}{E}$ between $10^{-5}$ and 10 MeV):

```
ENERGETIC_DISTRIBUTION
      SPECTRUM    INTERPOLATED
     2 1
     IN_LOG_LOG 1
     1.E-5 1.E5
     10.  0.1
```

### 12.4.8 ANALYTICAL DISTRIBUTION

The distribution is expressed by an algebraic formula identical to that used for geometrical distribution. Expression of the formulas and domains involves the use of the variable E.
The operators used in the formula expression are (LOG is the natural logarithm):
+ , - , *, /, SH, CH, SIN, COS, EXP, LOG, SQRT, ().
The relationships accepted in domain definitions are: $<$ and $>$.

Note: If an expression begins with a $-$ sign, a 0 must be placed before it: EXP(0-X), for $\exp(-x)$ for instance.

```
ENERGETIC_DISTRIBUTION
      SPECTRUM
            ANALYTICAL
                  FUNCTION = ... ;
                  DOMAIN = ... ;
```

**WARNING !** If the user wants to use a spectrum such as $S(E) \propto \frac{1}{E}$, with a domain going over several orders of magnitude, the sampling of the analytical dirtibution is not accurate to do this. In this case, it is advised to use a tabulated source with the `IN_LOG_LOG` interpolation as specified in the example above.

Example (source in $\exp(-E)$ between 1 and 2 MeV):

```
ENERGETIC_DISTRIBUTION
     ANALYTICAL
          FUNCTION= EXP(0-E);
          DOMAIN=1<E<2;
```

### 12.4.9 FISSION SPECTRUM FROM THE EVALUATION

The distribution is the fission spectrum read in the evaluation for a given nuclide at 300 K and a given incident energy. If the energy is not directly tabulated in the evaluation an interpolation is made according to the interpolation scheme also given in the evaluation.
This option is available only when the fission spectrum is given as a tabulated law or a Maxwell law distribution in the evaluation ("LF1" and "LF7" laws of the "file 5" in the ENDF/B format). Otherwise a message error is produced by the code.
The syntax is as follows:

**ENERGETIC_DISTRIBUTION**
**SPECTRUM**
**FISSION_FROM_EVALUATION isotope**
* ***THERMAL***
* ***FAST***
* ***ENERGY*** $E_0$

where:

- `isotope` is one of the nuclide names given in the dictionary file (see chapter 3)
- `THERMAL` is equivalent to `ENERGY 2.5E-6`
- `FAST` is equivalent to `ENERGY 2.`
- $E_0$ is an energy chosen by the user in MeV. It is not an energy bound of the spectrum, but an incident energy for fission, as tabulated in the evaluation.

## 12.5 TIME DISTRIBUTION

Time distribution can be expressed in five different ways (Dirac, uniform, Gauss, tabulated, or analytical).
**Warning**: particle sources extending into negative times are allowed, but it is the user's responsibility to lower the value of the initial simulation time correspondingly (see keyword `TIME_INF` in Chapter 22). TRIPOLI-4

cannot automatically detect the problem and will silently kill any particle that undergoes a collision below the initial time.

### 12.5.1 DIRAC

This corresponds to a source at a given time input as a real number. The time distribution is a Dirac $S(t) = \delta(t - t_0)$, and the Dirac value is 1 when $t = t_0$.
The syntax is as follows:

**TIME_DISTRIBUTION**
**DIRAC $\mathbf{t_0}$**

Important example:
In the case of a problem that is not time dependent, the following is selected:
`TIME_DISTRIBUTION DIRAC 0`

### 12.5.2 UNIFORM DISTRIBUTION

This corresponds to uniform time distribution between two values specified by the user. The syntax is as follows:

**TIME_DISTRIBUTION**
**UNIFORM**
**$\mathbf{t}_{inf}$ $\mathbf{t}_{sup}$**

The time distribution is $S(t) = U(t_{sup}, t_{inf})$, such that $S(t) = 1$ if the time is within the bounds, and $S(t) = 0$ otherwise. The intensity, therefore, is equal to $t_{sup} - t_{inf}$.

### 12.5.3 GAUSS DISTRIBUTION

This corresponds to a Gauss time distribution centred around time $t_{\text{mean}}$ with a standard deviation $\sigma$. The time distribution is

$$S(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-t_{\text{mean}})^2}{2\sigma^2}} = \frac{1}{\sqrt{\pi}a} e^{-\frac{(t-t_{\text{mean}})^2}{a^2}}$$

The quantities $t_{\text{mean}}$ and $a$ are specified by the user using the following syntax:

**TIME_DISTRIBUTION**
**GAUSS**
**a $\mathbf{t_{mean}}$**

### 12.5.4 TABULATED DISTRIBUTION

This corresponds to a distribution whose values are given at a certain number of points and interpolated between them. The syntax used to define this type of distribution is similar to that of a response function expressed in the form of data (Section 13.4), or energy or angular tabulated distributions (with the times being given in increasing order).

**TIME_DISTRIBUTION**
**INTERPOLATED**
**nb(points) nb(inter)**
**<MODE_INTERPOLATION(1)>**
**num[limsup(1)]**
**...**
**<MODE_INTERPOLATION(nbinter)>**
**num[limsup(nbinter)]**
$\mathbf{t}_0$ $\mathbf{value}_0$
**...**
$\mathbf{t}_{nb(points)-1}$ $\mathbf{value}_{nb(points)-1}$

- `<MODE_INTERPOLATION>` belongs to list {`IN_NO_INTERPOLATION`, `IN_LIN_LIN`, `IN_LIN_LOG`, `IN_LOG_LIN`, `IN_LOG_LOG`};
- `nb(points)` is the number of points, and `nb(inter)` the number of different interpolation ranges;
- Mode `<MODE_INTERPOLATION(1)>` applies from $t_0$ to $t_{num[limsup(1)]}$;
- Mode `<MODE_INTERPOLATION(j)>` applies from $t_{num[limsup(j-1)]}$ to $t_{num[limsup(j)]}$ where $num[limsup(i)]$ is an integer, the index to the point in the list of given pairs (Caution: the numbering begins at 0).
- The value of the last couple ($\mathbf{t}_{nb(points)-1}$, $\mathbf{value}_{nb(points)-1}$) is untapped when `<MODE_INTERPOLATION>` is `IN_NO_INTERPOLATION`.

### 12.5.5 ANALYTICAL DISTRIBUTION

The distribution is expressed by an algebraic formula of the same type as that used for geometrical distribution. Expression of formulas and domains involves the use of variable `T`.
The operators in the expression of the formula are (`LOG` is the natural logarithm):
`+` , `-` , `*`, `/`, `SH`, `CH`, `SIN`, `COS`, `EXP`, `LOG`, `SQRT`, `()`.
Relationships accepted in domain definitions are: $<$ and $>$.

Note: If an expression begins with a $-$ sign, a 0 must be placed before it: `EXP(0-X)`, for $\exp(-x)$ for instance.

```
TIME_DISTRIBUTION
    ANALYTICAL
        FUNCTION = ... ;
        DOMAIN = ... ;
```

Example (source in $\exp(-t)$, $t$ from 0 to 30 seconds)

```
TIME_DISTRIBUTION
    ANALYTICAL
        FUNCTION= EXP(0-T);
        DOMAIN=0<T<30;
```

## 12.6 SOURCES TAKEN FROM SOURCE FILES

### 12.6.1 VOLUME PARTICLE SOURCES

Volume particle sources can be dumped using the STORE_*_IN_FILE directives in the simulation block (see section 22.20).
Each file corresponds to events (sources, collisions, leaks) throughout the simulation for a given list of batches. We can store the source events from a first calculation and restart them in a second calculation.
The command used in the second calculation is as follows:

```
SOURCES_LIST
    1
    SOURCE
        NEUTRON
        USE_FILE
*           ASCII
*           BINARY
            firstrun.t4.sources  LIST n i0 ... in-1
    END_SOURCE
END_SOURCES_LIST
```

where $n$ is the number of batches stored in the first run, $i_0$ ... $i_{n-1}$ are the batch numbers and firstrun.t4.sources is the prefix of the source event files.
The user must maintain the statistical normalization of the batches. If the first run used $n$ particles per batch, and $p$ restart files are loaded, the second run must use $n.p$ particles per batch (cf. keyword SIZE in the SIMULATION block, see chapter 22).

This allows, for example, preparing several storage files at the end of a first simulation of several batches apart (to reduce correlations) on a single processor and then resuming with a second calculation in parallel with a roughly converged source of larger batches. This minimizes the computational effort for the source convergence in criticality and maximizes performance of the parallelism of the second run.

### 12.6.2 SURFACE PARTICLE SOURCES

#### 12.6.2.1 Syntax

This type of source is used to perform simulations on " local " geometries after the characteristics of the particles have been stored in source files during a simulation on the " global " geometry (using the `STORAGE` command, or else the `SCORE` command with the `STORAGE` option). By re-using the source files during a local simulation, the source particles are directly injected using the batch by batch source. This may follow the order in which the batches were stored or else a random sampling of batches. This "zooming " approach is used in certain situations in order to obtain statistics where the global simulation alone does not yield satisfactory results. It also decreases calculation time when performing a parametric study concerning the geometry of the local simulation. Two figures illustrating this approach are presented at the end of this section.
The syntax used in defining a surface source taken from a source file is as follows:

```
SOURCES_LIST
        1
        SOURCE
                INTENSITY 1
                <PARTICLE>
                SOURCE_STORAGE  LIST  nb(fichiers)
                        file-name(1)
                        ...
                        file-name(n)
                <MODE>
                TIME_DISTRIBUTION
                        Description of time distribution
        END_SOURCE
END_SOURCES_LIST
```

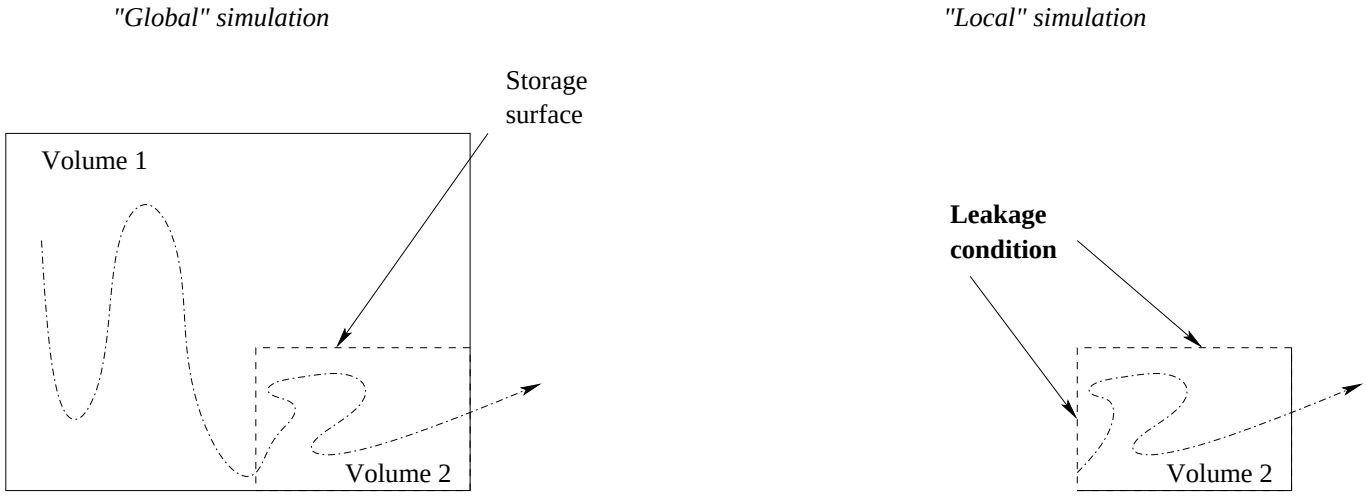
- a single source must be defined: nb(sources) equals 1
- the norm for sources given after the keyword `INTENSITY` is 1 (the correct value for this variable is actually read directly from the header of the source files)
- `<PARTICLE>` may be `NEUTRON`, `PHOTON`, or `ALL_PARTICLE` (the last option must be used if the data dump was created with these particles: neutrons and photons stored in the same source file)

- the file names file-name(i) are the complete names (including the suffixes when several files have been created due to exceeding the size limit for the source files). There must not be more than 256 files.
- restart in parallel mode should invoke a filename with the following syntax `source_filename.%SIMRANK%th_task_<istock>_.stock` . See section 12.6.2.3 for more details.
- `<MODE>` equals:
  - `ORDERED` n_discard, if the source of each batch is to be constituted following the order in which the batches were stored.
    The integer n_discard is the number of batches to be skipped at the beginning of the first file. This number must be greater than or equal to 1 (the first batch read will, therefore, be the n_discard + 1 $^{th}$. This first batch read may be empty or factitious). If biasing was used in the storage phase, it is recommended that n_discard be set greater than or equal to 2.
  - `REPLICATE` n, if each source particle is to be split in n particles with a statistical weight divided by n. In this case the batches are read sequentially and in order (as in the `ORDERED` option). This option will increase the simulation time but allows to have a better precision in the whole phase space, especially when only few particles have been stored.
  - `RANDOM`, if the batches are to be chosen at random. Warning: Extra care must be taken in this mode as the sources may be sampled multiple times leading to an underestimated standard deviation. The user is advised not to use this mode or to check he will not ask for more batches than available in the source file.
- the syntax used in defining the time distribution is the same as for other types of sources.

**Notes:**

- In order for the source particles coming from the source files to be created strictly inside the geometry, these particles are "pushed" forward by the code by a quantity $\varepsilon_{geo}$.
- In order not to count particles twice, the surface where the stored particles are re-introduced should have a leakage boundary condition. This is the default boundary condition in TRIPOLI-4® for surfaces at the boundary of the geometry, but it is also possible to use a leakage boundary condition on a surface inside the geometry (see chapter 9);
- In the `SIMULATION` command, the simulation mode must be set to **SHIELDING** (default simulation mode in TRIPOLI-4®);

#### 12.6.2.2 Zooming approach principle




#### 12.6.2.3 Parallel mode

The parallel version of this feature will restart each stored file on each processor as if the calculation was not interrupted. That means that the sources produced on the simulator *i* in the first pass will be used on the simulator *i* in the second pass.
In this case, the number of processors in the second pass should be less or equal than the number of processors in the first pass.
In previous versions, the user may be tempted to use all the sources produced in the first pass to restart a second pass (with the `RANDOM` option or not). It is considered potentially harmful as it can introduce correlations between the samples leading to underestimated variances.
In parallel, the syntax is almost identical except the source filename should be specified with the following syntax `source_filename.%SIMRANK%th_task_<istock>_.stock`. The special token `%SIMRANK%` in the filename will be replaced with the index of the task.

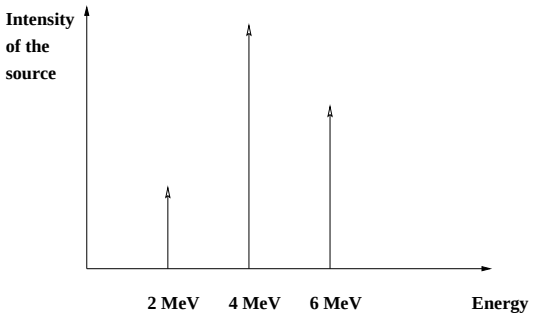
## 12.7 COMPLETE SOURCE EXAMPLES

### 12.7.1 ISOTROPIC POINT SOURCE WITH NO TIME DEPENDENCE

A one particle per second source is required. A temporal Dirac is chosen. The source is, therefore, normalized to one particle (the integral of the Dirac is 1). It is placed at the origin of the reference frame.

#### 12.7.1.1 Line spectrum

A source whose relative strength is known is given at 3 energies: a relative strength of 1 at 2 MeV, a relative strength of 3 at 4 MeV, and a relative strength of 2 at 6 MeV. The spectrum is then the sum of three Diracs, and the source has to be divided into three contributions.




The total source is written as follows (the first source has a normalisation of 1, the second a normalisation of 3 and the third a normalisation of 2)

```
SOURCES_LIST
     3
     SOURCE INTENSITY 1 NEUTRON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM MONOKINETIC 2
            TIME_DISTRIBUTION DIRAC 0.
     END_SOURCES
     SOURCE INTENSITY 3 NEUTRON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM MONOKINETIC 4
            TIME_DISTRIBUTION DIRAC 0.
     END_SOURCES
     SOURCE INTENSITY 2 NEUTRON
```

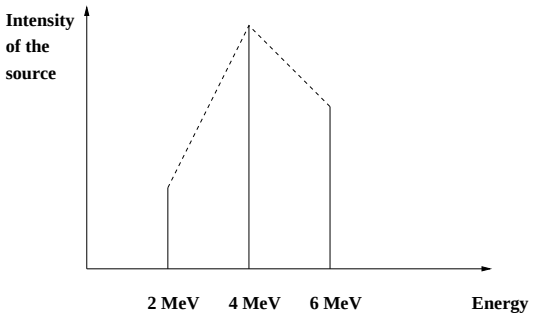
```
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM MONOKINETIC 6
            TIME_DISTRIBUTION DIRAC 0.
     END_SOURCES
END_SOURCES_LIST
```

A more concise formulation of this is obtained using a line spectrum.

```
SOURCES_LIST
     1
     SOURCE INTENSITY 1 NEUTRON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM RAY_SPECTRUM 3
                      2 1
                      4 3
                      6 2
            TIME_DISTRIBUTION DIRAC 0.
     END_SOURCES
END_SOURCES_LIST
```

#### 12.7.1.2 Interpolated spectrum

If the user wishes to interpolate the spectrum between these points linearly, a single INTERPOLATED source is selected. There are three interpolation points and a single interpolation type (linear-linear), which ends at the last energy (the last energy bound is number 2 because the numbering starts at 0).




```
SOURCES_LIST
     1
     SOURCE INTENSITY 1 NEUTRON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM INTERPOLATED 3 1
                               IN_LIN_LIN 2
                               2 1
                               4 3
                               6 2
            TIME_DISTRIBUTION DIRAC 0.
     END_SOURCES
END_SOURCES_LIST
```

### 12.7.2 DISTRIBUTED ISOTROPIC SOURCE WITHOUT TIME DEPENDENCE

Suppose we wish to simulate a one-dimensional cylindrical case (for instance, a comparison using a SN code), in which the source is "linear", i.e., contained within a cylinder with a small radius. TRIPOLI-4® requires a three-dimensional geometry. For simplicity, an isotropic mono-kinetic source can be considered. To ensure that the domain is not larger than the emission volume, it is best to consider the case where it is the domain that sets the bounds for the source and that allows for the geometry definition. In any case, the domain must not be substantially larger than the volume. The implicit convention in the example is that volume 1 contains the domain, and the z-axis geometry is bounded by $-10 < z < 10$.

Example: Radius of cylinder 1: 0.05 cm and the following definition:

```
SOURCES_LIST
     1
     SOURCE INTENSITY 1 NEUTRON
            FACTORIZED
                      FRAME CYLINDER
                    0 0 0
                    1.  0.  0.
                    0.  0.  1.
                      VOLUME 1 1
            GEOMETRIC_DISTRIBUTION
                      ANALYTICAL
                    FUNCTION = 1.;
                    DOMAIN = 0.< R < 0.05, 0 < RTHETA < 6.2832,-10.< Z < 10 ;
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM MONOKINETIC 1
            TIME_DISTRIBUTION
                      DIRAC 0.
     END_SOURCES
END_SOURCES_LIST
```

But certainly not radius of cylinder 1: 0.05 cm and the following defintion

```
SOURCES_LIST
     1
     SOURCE INTENSITY 1 NEUTRON
            FACTORIZED
                      FRAME CYLINDER
                    0 0 0
                    1.  0.  0.
                    0.  0.  1.
                      VOLUME 1 1
            GEOMETRIC_DISTRIBUTION
                      ANALYTICAL
                    FUNCTION = 1.;
                    DOMAIN = 0.< R < 5, 0 < RTHETA < 6.2832,-10.< Z < 10 ;
            ANGULAR_DISTRIBUTION
                      ISOTROPIC
            ENERGETIC_DISTRIBUTION
                      SPECTRUM MONOKINETIC 1
            TIME_DISTRIBUTION
                      DIRAC 0.
```

```
    END_SOURCES
END_SOURCES_LIST
```

For the calculation of the source normalization, points are quasi-randomly sampled throughout the definition domain. But only the points belonging to the intersection of the domain and volume 1 will be used. If this intersection is too small, the number of points retained will be small, and the volume calculation (and hence the source normalization) may be inaccurate.

## Defining the sources for the criticality calculation example of chapter 5

```
SOURCES_LIST
  2
  SOURCE NEUTRON
    POINT -15.1 0.  199.57
    ANGULAR_DISTRIBUTION
      ISOTROPIC
    ENERGETIC_DISTRIBUTION
      SPECTRUM WATT_SPECTRUM
    TIME_DISTRIBUTION
      DIRAC 0.
  END_SOURCES
  SOURCE NEUTRON
    FACTORIZED
      FRAME CYLINDER
        15.1 0.  185.3
        1.  0.  0.
        0.  0.  1.
      VOLUME 1 8
    GEOMETRIC_DISTRIBUTION
      ANALYTICAL
        FUNCTION = 1.;
        DOMAIN = 0.  < R < 14.55, 0 < RTHETA < 6.2832, 0.  < Z < 28.54;
    ANGULAR_DISTRIBUTION
      ISOTROPIC
    ENERGETIC_DISTRIBUTION
      SPECTRUM WATT_SPECTRUM
    TIME_DISTRIBUTION
      DIRAC 0.
  END_SOURCES
END_SOURCES_LIST
```

## 12.8 EXTERNAL SOURCES

### 12.8.1 DESCRIPTION

The feature external source allows the user to supply any source using a source function written in C or Fortran language and dynamically linked to TRIPOLI-4® at the beginning of the run. The user can define a source as complex as desired, with energy-angle-position correlations (as in bremsstrahlung showers for example). This feature is reserved to advanced Tripoli users as they have to provide their own sampling algorithm.

The general syntax is as follows:

```
        SOURCE
#               INTENSITY  norm(source)
#               COEFF  coeff(source)
               <PARTICLE_TYPE>
                       EXTERNAL  source.so
                       NO_GEOMSOURCE_WARNING
                       SOBOL_POINTS  nb(points)
                       VOLUME  nb(volumes)
                               volume(1) ... volume(n)
                       PARAM  nb(params)
                               param(1) ... param(n)
        END_SOURCE
```

where :

**INTENSITY** same meaning than previously (see section 12.1)

**COEFF** same meaning than previously (see section 12.1)

<PARTICLE_TYPE> can be NEUTRON, PHOTON, ELECTRON, POSITRON or ALL_PARTICLE
If it is ALL_PARTICLE, no rejection test on particle type is carried out

**EXTERNAL source.so** source.so is the name of the dynamic loading library containing the source function. Its path must be included in *$LD_LIBRARY_PATH* environment variable.

**NO_GEOMSOURCE_WARNING** optional keyword, if present it disables low sampling efficiency test (by default, the calculation is stopped if efficiency $< 10\%$)

**SOBOL_POINTS nb(points)** optional, sets the number of sampling tests to *n*b(points), default=1000

**VOLUME nb(volumes) volume(1) ... volume(n)** optional, sets the volumes in which sources must be sampled and calculates a sampling efficiency.

If not present, all volumes are accepted.

**PARAM nb(param) param(1) ... param(n)(n)** List of numerical parameters transmitted to source function ( $max = 10$ ).

The function name must be necessary source and the call arguments have to be :

**int *ipt** type of particle to create

- 1 = neutron
- 2 = photon
- 3 = electron
- 4 = positron

**double *x** x position of the new particle created
**double *y** y position of the new particle created
**double *z** z position of the new particle created
**double *u** direction cosine along x
**double *v** direction cosine along y
**double *w** direction cosine along z
**double *e** particle energy
**double *we** particle statistical weight
**double *t** particle time
**double *param** list of parameters from input TRIPOLI-4® file
**double (*randtri)(void)** pointer to TRIPOLI-4®'s pseudo-random-number generator

By default, at each function call, the variables are initialized :

- **ipt=1** (neutron)
- **x=y=z=0.**
- **u, v, w** isotropic
- **e=14 MeV**
- **we=1.**
- **t=0.**

Compilation options depend on the operating system, in some cases (eg AIX), it may be necessary to call fortran functions via C function interfaces. During the TRIPOLI-4® source initialization phase, some verification calculations are provided to the user (means, standard deviations, sampling efficiency). They are performed on a sample of 1000 particles by default.

### 12.8.2 EXAMPLE

This example defines a linear source from x=-10 cm to 0, in volumes 10 and 11 (possible rejection) of 1.33 MeV photons. The direction is inside a cone with an opening angle of ± 5 degrees along z. Source location on x, particle type and energy are set from TRIPOLI-4® input data file.
For the sake of reproducibility, the pseudo-random numbers are drawn from TRIPOLI-4®'s generator.
Listings 12.1 and 12.2 show identical source functions written in C and Fortran.

The source definition in the input data file is as follows :

```
SOURCES_LIST
     1
     SOURCE ALL_PARTICLE
            EXTERNAL source.so
            VOLU 2 10 11  // source must be in volume 10 or 11 (or it will be resampled)
            PARAM 4 -5.0 10.  2 1.33  // center, length, particle (2=photon), energy (MeV)
     END_SOURCES
END_SOURCES_LIST
```

NOTES :

- The Fortran source subroutines used for MCNP are compatible with those of TRIPOLI-4® after updating the call arguments and random number generator. It is not necessary to recompile the TRIPOLI-4® code.
- Be careful if point estimators (FLUX_PT) are used in the response list. The direct contribution from the sources is assumed to be isotropic.
- When used in parallel calculation, if one considers that the sequence of random source sampling is of importance on scores in relation to the transport part, it is recommended to use the TRIPOLI-4® random-number generator passed as parameter **randtri**. Alternatively, the user has to manage the initialization of source function random generator for each process.

```c
// compilation on linux os via "gcc source.c -fPIC -shared -o source.so"
#include <stdlib.h>
#include <math.h>

void source(int *ipt,double *x,double *y,double *z,double *u,double *v,double *w,
            double *e,double *we,double *t,double *param, double (*randtri)(void)) {
  double phi,costheta,sintheta,cos5 ;

  // by default x=0, y=0, z=0
  *x= (param[0]-param[1]*0.5) + randtri()*param[1] ; // param[0]=center,  param[1]=length

  // (u,v,w) isotropic by default
  cos5=cos(5./360.*6.283185);
  phi=randtri()*6.283185 ;
```

```
  costheta=randtri()*(1.0 - cos5)  + cos5 ;
  sintheta=sqrt(1.0 - costheta*costheta);
  *u=costheta;
  *v=sintheta*sin(phi);
  *w=sintheta*cos(phi);

  // by default ipt=1 (neutron), e = 14 MeV, we = weight = 1, t = time = 0
  *ipt=param[2]; // particle 1=neutron, 2=photon, 3=electron, 4=positron
  *e=param[3];   // energy (MeV)
  return ;
}
```

Listing 12.1: Example of C function source.

```
c compilation on linux os via "gfortran -fno-underscoring source.f -fPIC -shared -o source.so"
       subroutine source(ipt,x,y,z,u,v,w,e,we,t,param,randtri)
       double precision x,y,z,u,v,w,e,we,t,param,randtri
       double precision phi,costheta,sintheta,cos5
       dimension param(*)

c by default x=0, y=0, z=0
       x= (param(1) - param(2)*0.5) + randtri()*param(2) ! param(1)=center,  param(2)=length

c (u,v,w) isotropic by default
       cos5 = cos(5./360.*6.283185)
       phi=randtri()*6.283185
       costheta=randtri()*(1.0 - cos5) + cos5
       sintheta=sqrt(1.0 - costheta*costheta);
       u=costheta
       v=sintheta*sin(phi)
       w=sintheta*cos(phi)

c by default ipt=1 (neutron), e = 14 MeV, we = weight = 1, t = time = 0
       ipt=param(3)  ! particle 1=neutron, 2=photon, 3=electron, 4=positron
       e=param(4)    ! energy (MeV)
       end
```

Listing 12.2: Example of Fortran function source.

## 12.9 MULTI-PARTICLE SOURCES

Standard TRIPOLI-4 sources generate one particle per history. For the purpose of calculating average quantities like fluxes or reaction rates, the correlations between source particles are immaterial; even if the Monte Carlo simulation does not account for them, the calculated quantities are only sensitive to the singles distributions (energy spectra, angular distributions, etc.). However, for the calculation of some observables, correlations between source particles are crucial.
One example of such observables is given by the distribution of events with respect to the energy deposited in a gamma spectrometer. For instance, consider the simultaneous emission of 1.17 and 1.33 MeV photons arising from the decay of $^{60}$Co. If both photons are absorbed by the spectrometer, the detector will see the sum of their energies. Clearly, a simulation where each photon energy appears independently of the other can

```
SOURCES_LIST 2
      NORM 1
      SAME_HISTORY
      SOURCE
            PHOTON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION SPECTRUM MONOKINETIC 1.17
            TIME_DISTRIBUTION DIRAC 0.
      END_SOURCE
      SOURCE
            PHOTON
            POINT 0 0 0
            ANGULAR_DISTRIBUTION ISOTROPIC
            ENERGETIC_DISTRIBUTION SPECTRUM MONOKINETIC 1.33
            TIME_DISTRIBUTION DIRAC 0.
      END_SOURCE
END_SOURCES_LIST
```

Figure 12.7: Example of SOURCES_LIST block that generates uncorrelated photons from the decay of $^{60}$Co.

never reproduce the occurrence of the sum peak.
Multi-particle sources are a TRIPOLI-4® feature that make it possible to model correlated or uncorrelated creation of multiple particles in the same history.

### 12.9.1 UNCORRELATED SOURCE PARTICLES

The simplest way to activate this feature is to use the SAME_HISTORY keyword in the SOURCES_LIST; this tells TRIPOLI-4 to seed histories by sampling one particle from each defined source, instead of one particle from a randomly-chosen source. When using SAME_HISTORY, all the sources in SOURCES_LIST must have the same intensity.
An example SOURCES_LIST producing two uncorrelated $^{60}$Co gamma rays is shown in Figure 12.7.

### 12.9.2 CORRELATED SOURCE PARTICLES

More complex multi-particle sources (e.g. sources emitting correlated particles) can be introduced using external sources (see Section 12.8 above) with the MULTIPLE keyword, like in Figure 12.8.
Here multi_ext.so is a shared library that must contain an external C++ source function (see Section 12.8) with the following name and signature:

```
SOURCES_LIST 2
      SOURCE
            PHOTON
            EXTERNAL multi_ext.so
            MULTIPLE
      END_SOURCE
END_SOURCES_LIST
```

Figure 12.8: Example of SOURCES_LIST block that generates correlated source particles by calling an external source function.

```
extern "C"
void source_multiple(std::vector<int> &ipts,  std::vector<double> &xs,
                     std::vector<double> &ys, std::vector<double> &zs,
                     std::vector<double> &us, std::vector<double> &vs,
                     std::vector<double> &ws, std::vector<double> &es,
                     std::vector<double> &wes, std::vector<double> &ts,
                     const double * const params, void (*randtri)(void));
```

The source_multiple function may take parameters through the params array, may call the randtri() function to generate pseudo-random numbers and must fill the argument vectors ipts, xs, etc. with the characteristics of the sampled particles. The meaning of the arguments are the same as in the case of traditional single-particle external sources (function source, Section 12.8). Note that the source file for a multi-particle external source cannot be written in C or Fortran.
An example of source file for a multi-particle external source is shown in Listing 12.3.

```
// compilation on linux os via "g++ source_multiple.cc -fPIC -shared -o source.so"
#include <vector>
#include <cmath>
#include <iostream>

const double pi = 3.14159265359;

typedef double random_fct_type(void);
typedef void source_multiple_fct_type(std::vector<int> &,     std::vector<double> &,
                                      std::vector<double> &, std::vector<double> &,
                                      std::vector<double> &, std::vector<double> &,
                                      std::vector<double> &, std::vector<double> &,
                                      const double * const, random_fct_type);

/* The angular distribution is taken to be
 *
 *   W(theta) = N ( 1 + a2 cos(theta)^2 + a4 cos(theta)^4 )
 *
 * with a2=9e-3 and a4=8e-3. The normalisation constant is
 *
 *   N = 8 / (8 + 4 a2 + 3 a4)
 *
 * and the maximum occurs at 0, pi:
 *
 *   W(0) = W(pi) = norm * ( 1 + a2 + a4 )
 *
 * Since the CDF cannot be analytically inverted, we sample the PDF by
 * rejection.
 */

const double a2 = 9e-2;
const double a4 = 8e-2;
const double norm = 8. / (8. + 4.*a2 + 3.*a4);

double dist(const double cosx)
{
  const double cosx2 = cosx*cosx;
  const double cosx4 = cosx2*cosx2;
  return norm * (1 + a2 * cosx2 + a4 * cosx4);
}

const double distmax = dist(1.);

template<typename T> struct Vec {
  T x;
  T y;
  T z;
  Vec(const T &ax, const T &ay, const T &az) : x(ax), y(ay), z(az) {}
  T dot(const Vec<T> &other) const { return x*other.x + y*other.y + z*other.z; }
  Vec<T> vector(const Vec<T> &other) const {
    return Vec( y * other.z - z * other.y,
                z * other.x - x * other.z,
```

```
                x * other.y - y * other.x );
  }
  Vec<T> operator*(const T &a) const { return Vec<T>(a*x, a*y, a*z); }
  Vec<T> operator+(const Vec<T> &other) const { return Vec<T>(x+other.x, y+other.y, z+other.z); }
};


extern "C" {
  void source_multiple(std::vector<int> &ipts,  std::vector<double> &xs,
                       std::vector<double> &ys, std::vector<double> &zs,
                       std::vector<double> &us, std::vector<double> &vs,
                       std::vector<double> &ws, std::vector<double> &es,
                       std::vector<double> &wes, std::vector<double> &ts,
                       const double * const params, random_fct_type randtri);
}

void source_multiple(std::vector<int> &ipts,  std::vector<double> &xs,
                     std::vector<double> &ys, std::vector<double> &zs,
                     std::vector<double> &us, std::vector<double> &vs,
                     std::vector<double> &ws, std::vector<double> &es,
                     std::vector<double> &wes, std::vector<double> &ts,
                     const double * const params, random_fct_type randtri) {

  const double e1 = 1.17;
  const double e2 = 1.33;
  const double x = 0.;
  const double y = 0.;
  const double z = 0.;
  const double costheta1 = std::min(std::max(2.*randtri()-1., -1.), 1.);
  const double sintheta1 = std::sqrt((1. - costheta1)*(1. + costheta1));
  const double phi1 = 2. * pi * randtri();
  const Vec<double> v1 =
    Vec<double>(sintheta1 * std::cos(phi1),
                sintheta1 * std::sin(phi1),
                costheta1);

  double costheta12, xi;
  do { // rejection loop
    costheta12 = std::min(std::max(2.*randtri()-1., -1.), 1.);
    xi = distmax * randtri();
  } while(xi >= dist(costheta12));
  const double sintheta12 = std::sqrt((1. - costheta12)*(1. + costheta12));
  const double phi12 = 2. * pi * randtri();
  const Vec<double> v12 =
    Vec<double>(sintheta12 * std::cos(phi12),
                sintheta12 * std::sin(phi12),
                costheta12);

  Vec<double> v2(0., 0., 0.);
  if(sintheta1==0.) { // degenerate case
    v2 = v12;
  } else {
    Vec<double> k = Vec<double>(0., 0., 1./std::fabs(sintheta1)).vector(v1);
```

```
    // Rodrigues' rotation formula
    v2 = v12 * costheta1 + k.vector(v12) * sintheta1 + k * (k.dot(v12)*(1. - costheta1));
  }

  ipts.push_back(2); // photon
  xs.push_back(x); ys.push_back(y); zs.push_back(z);
  us.push_back(v1.x); vs.push_back(v1.y); ws.push_back(v1.z);
  es.push_back(e1); wes.push_back(1.); ts.push_back(0.);

  ipts.push_back(2); // photon
  xs.push_back(x); ys.push_back(y); zs.push_back(z);
  us.push_back(v2.x); vs.push_back(v2.y); ws.push_back(v2.z);
  es.push_back(e2); wes.push_back(1.); ts.push_back(0.);
}
```

Listing 12.3: Example C++ source listing for a multi-particle external source. The source produces two correlated photons from the decay of $^{60}$Co; the angle $\theta$ between the photon emission directions is assumed to follow the distribution $W(\theta) = N \cdot (1 + a_2 \cos^2\theta + a_4 \cos^4\theta)$, with $N = 8/(8 + 4a_2 + 3a_4)$ being a normalization constant.

## 12.10 SOURCES FOR ACTIVATION CALCULATIONS

In the second step of Rigorous Two-Step (R2S) activation calculations, a special activation source must be defined. Please refer to Section 41.2.2 for further details.

## 12.11 GEANT4 PARTICLE SOURCES

The TRIPOLI-4/GEANT4 coupling makes it possible to directly use particle sources from a GEANT4 simulation in TRIPOLI-4. For further details, please consult Section 42.5.

## 12.12 SPONTANEOUS FISSION SOURCES WITH THE FIFRELIN FISSION MODEL

A spontaneous fission source can be defined using fission event files produced by the FIFRELIN fission model code (see also Section 22.1 for details on the coupling between TRIPOLI-4® and FIFRELIN codes by external files). The syntax is as follows:

```
SOURCE
      ALL_PARTICLE  SPONTANEOUS_FISSION  FIFRELIN  za
            USE_FILE  format n(files)
            path_to_file_1
            ...
            path_to_file_n
            RADIUS  R
            HEIGHT  h
            TIME_INF  T_inf
            TIME_SUP  T_sup
END_SOURCE
```

with the same conventions after `FIFRELIN` keyword as for induced fission (see Section 22.1). The keyword `ALL_PARTICLE` has to be used for the particle type, so as to sample fission neutrons and photons. The emission point coordinates are sampled in the sphere of radius $R$ and center $x = 0$, $y = 0$, $z = h$. Emission times are between $T_{inf}$ and $T_{sup}$.

# Chapter 13

# DEFINING RESPONSE TYPES

Calculating a score requires knowledge of the following:

- the type of the response function (flux, current, reaction rate, ...);
- the energy, angle, or time grid on which the results are required;
- the points, surfaces, or volumes of the geometry in which the response is to be computed, and the chosen estimator.

The tasks are divided among three modules: the `RESPONSES` module, which defines the type of the responses, the `GRID_LIST` module, which defines the grids, and the `SCORE` module, which specifies the geometrical data and the estimators and uses the data previously described in the `RESPONSES` and `GRID_LIST` modules. The responses are defined between the `RESPONSES` and `END_RESPONSES` keywords.

## 13.1 THE AVAILABLE RESPONSES

Responses are physical quantities calculated by the code during a simulation.
Responses can be of the following types:

- a flux (in neutronics, this corresponds to a density multiplied by a speed);
- a current;
- a dose equivalent rate H*(10), 1997 (ICRP-74 standard) [45];
- a dose equivalent rate ANS, 1977 [3] (this is for backward compatibility only);
- a reaction rate (to be specified);
- a macroscopic reaction rate (to be specified);
- an energy deposition in matter;
- a kinematic limit;

- a Displacement Per Atom (DPA);
- a gas production;
- a fission spectrum;
- a production of particles of a given type (for example a photon source or an electron source);
- a histogram of deposited energy per source particle (gamma detector spectrum for example);
- a histogram of deposited charge per source particle;
- a total deposited energy per source particle (gamma dose for example);
- a fixed source emission;
- a histogram of number of fission neutrons sampled per fission;
- a histogram of the sum of fission neutron energies per fission;
- a (Z,A) distribution of the fission products (after emission of prompt neutrons).

The units are for:

- volume flux: by default, the volume-integrated flux values are given in the number of particles times the unit length per unit time (neutron·cm/s); if the volume is given, the values will be in the number of particles per unit area per unit time (neutron/(cm$^2$·s));
- surface flux: by default, the surface-integrated flux values are given in the number of particles per unit time (neutron/s); if the surface area is given, the values will be given in the number of particles per unit area per unit time (neutron/(cm$^2$·s));
- point flux: the values are in number of particles per unit area per unit time (neutron/(cm$^2$·s));
- current: same units as for surface flux;
- energy deposition in the material or kinematic limits: MeV/s;
- dpa: (displaced atoms)/s;
- reaction rate or macroscopic reaction rate: same as flux, but with the conversion factor (flux, rate);
- dose equivalent rate H*(10): $\mu$Sv/h if the volume is given;
- dose equivalent rate ANS: rem/h if the volume is given;
- production of particles and gas production: particle_type/s (for example photon/s for a photon source, or alpha particles/s for gas production);
- detector spectrum : counts/s;
- charge deposition spectrum: the score result is given as charge deposition (in units of the elementary charge) per unit time and unit volume: charge/(s cm$^3$);

- fixed source emission: the same units as for the source definition;
- histogram of number of fission neutrons: neutron/s;
- (Z,A) distribution: the same units as for the source definition.

Each response corresponds to a type of particle (NEUTRON, PHOTON, ELECTRON, or POSITRON).
However, the concept of flux as used in neutron and photon transport is not meaningful for charged particles. One exception is DEPOSITED_SPECTRUM and DEPOSITED_SPECTRUM_WEIGHTED_ENERGY if PHOTON, ELECTRON and POSITRON are included in the simulation. There is no difference between deposited energy by these particles, then the response particle must be the list PHOTON ELECTRON POSITRON.
Responses are defined sequentially after giving the number of total responses. There can be several reaction rates. The syntax is as follows:

**RESPONSES**
**nb(responses)**
*NAME* <*name1*>
<**type1**> <**particle1**>
<*additional data*>
...
*NAME* <*nameN*>
<**typeN**> <**particleN**>
<*additional data*>
**END_RESPONSES**

Available responses <types> are:

1. FLUX
2. DOSE: equivalent to DOSE_H*(10) (DOSE_ANS77 is still available)
3. CURRENT
4. WEIGHTED_COURANT (see sections 13.4 and 13.5 for the additional data)
5. REACTION (see sections 13.2 to 13.4 for the additional data)
6. MACROSCOPIC_REACTION_RATE (see section 13.6 for more information)
7. DEPOSITED_ENERGY (see section 13.7 for more information)
8. KINEMATIC_LIMITS (see section 13.8 for more information)
9. DPA (see section 13.9 for more information)
10. GAS_PRODUCTION (see section 13.10 for more information)

11. FISSION_SPECTRUM (see section 13.11 for more information)
12. PRODUCTION (see the following notes for more information)
13. DEPOSITED_SPECTRUM
14. DEPOSITED_SPECTRUM_WEIGHTED_ENERGY
15. DEPOSITED_CHARGE (see section 13.13 for more information)
16. FIXED_SOURCE_EMISSION (see section 13.12 for more information)
17. NU_FISSION
18. FISSION_N_SUM_SPECTRUM
19. FISSION_ZA_DISTRIBUTION

It is possible to assign a name to the responses (using the keyword NAME).
Additional data are required only for reaction rates and weighted currents (WEIGHTED_COURANT).

Notes:

- The flux per unit area (or current) cannot be calculated for all of the surfaces (see SCORE);
- For the DOSE_H*(10) response, only particle types NEUTRON or PHOTON can be specified;
- The DEPOSITED_SPECTRUM response is used to calculate the spectrum of electromagnetic deposited energy in a detector. This response can be used with or without the electromagnetic shower option (in the SIMULATION command, see chapter 22), however if PHOTON, ELECTRON and POSITRON are included in SIMULATION command, then all these particles have to be included also in response particle data whereas if only PHOTON is present, then only PHOTON is needed in response.
  Note also that if a variance reduction technique is needed together with this specific response, the only available technique is the AMS technique (see Chapter 18).
- The DEPOSITED_SPECTRUM_WEIGHTED_ENERGY response is used to calculate electromagnetic deposited energy per source particle in a detector ($\gamma$ dose). It's a physical reference calculation of $\gamma$ dose without tabulated response function but may be time consuming. The same DEPOSITED_SPECTRUM remarks apply to DEPOSITED_SPECTRUM_WEIGHTED_ENERGY. The integral values over all energy domain of DEPOSITED_SPECTRUM_WEIGHTED_ENERGY and DEPOSITED_ENERGY must be equal (in $\gamma e^{\pm}$ problems) but the energy grid meaning differs: it is incident energy in DEPOSITED_ENERGY and deposited energy in DEPOSITED_SPECTRUM_WEIGHTED_ENERGY.
- The PRODUCTION response gives the production of secondary particles of a given type 2 (neutron, photon, electron or positron), produced by particles of a given type 1 (neutron, photon, electron or positron). Additional data must be defined with the following syntax:

**PRODUCTION <particle_type_2> FROM <particle_type_1> <spectrum_type>**

where <spectrum_type> is either SECONDARY_SPECTRUM or INCIDENT_SPECTRUM, corresponding to results requested in an energy grid relatively to the secondary particle or to the incident particle.
Note that:

- keywords PHOTON_REPARTITION and ELECTRON_REPARTITION (available from earlier versions than 7) are still available and are equivalent to the syntax : PRODUCTION PHOTON FROM NEUTRON SECONDARY_SPECTRUM and PRODUCTION ELECTRON FROM PHOTON SECONDARY_SPECTRUM.

- when <particle_type_2> = <particle_type_1>, all reactions with one or several outgoing particles of the required type are taken into account (for example, for the case <particle_type_2> = <particle_type_1> = NEUTRON, all scattering and fission neutrons are collected).
- if the THICK_TARGET_BREMSSTRAHLUNG option is used, photons produced by Bremsstrahlung are collected by PHOTON as <particle_type_1> of the PRODUCTION response (and not ELECTRON, which has to be used in the frame of the full electromagnetic shower only).

- For scores associated with the PHOTON_REPARTITION response function (or its equivalent with the PRODUCTION keyword), the only gamma data taken into account are those given in the evaluations associated with **the collided nucleus** undergoing interactions processed in simulations (gamma productions associated with possible products of reactions are not taken into account).

- The NU_FISSION and FISSION_N_SUM_SPECTRUM responses are used to calculate a histogram of the number of fission neutrons per fission event, and a histogram of the sum of fission neutron energies per fission event, as sampled during the simulation. These responses are of interest when fission is sampled from a fission model (see Section 22.1). The distribution of the neutron multiplicity is then different from the two-group histogram of integer numbers surrounding the mean value provided by the evaluated data, produced when no fission model is used. The particle type must be NEUTRON. These responses are available only with the analog simulation mode for neutrons, with or without a fission model (see Section 22.1).

- The FISSION_ZA_DISTRIBUTION response is available with analog neutron transport with a fission model (LLNL_FISSION or FIFRELIN in the SIMULATION command block, see Section 22.1). The Z and A values of the fission products, after emission of prompt neutrons, are given by the fission model to TRIPOLI-4®. When the TRIPOLI-4® results are edited, the distribution is provided in a 2D (Z,A) mesh defined by the user in the SCORE command block (see Section 15.7). The particle type of the RESPONSES command block must be NEUTRON.

## 13.2 ADDITIONAL DATA FOR REACTION RATES

For a reaction rate, it is also necessary to provide data concerning the reaction rate to be calculated. First, the nucleus name `name(nucleus)` is given after the keyword `NUCLEUS`. This name must be one of the nucleus names in the library, unless the reaction is defined by `DATA`. This data may be given in three different forms:

- A nucleus in a composition and one or more interactions (introduced using `COMPOSITION`).
  This corresponds to a macroscopic reaction rate. If several interactions are required, they will be combined to give the reaction rate sought;
- A virtual nucleus and one or more interactions (introduced using `FICTIVE`). This corresponds to a microscopic reaction rate. Here again, the reaction rate is the sum of the reaction rates of the different interactions requested. A virtual nucleus is defined as a nucleus which is not found in any of the combinations in the problem. The temperature must be indicated, in the form of an integer.

  Important: in the above two cases, the code seeks the nucleus in the ENDF files. But to be able to access it, it is necessary for the particle concerned (`NEUTRON` , `PHOTON`, `ELECTRON`, or `POSITRON`) to be declared in the `SIMULATION` command. Therefore, if a neutron reaction is requested for a gamma only problem, the code will fail, even if the response is not used;
- A nucleus (any name is acceptable) and cross-section data tabulated with an interpolation mode. In this case, the user inputs the cross-section to be used in the reaction rate calculation. The cross-section value data are stored internally according to an NJOY cross-section model.

The syntax is, therefore, as follows:

```
     REACTION <PARTICLE_TYPE>
          NUCLEUS name(nucleus)
*              COMPOSITION name(compo)
                    INTERACTION nb(interactions)
                    code_interactions(1) ... code_interactions(n)
*              FICTIVE temperature
                    INTERACTION nb(interactions)
                    code_interactions(1) ... code_interactions(n)
*              ( COEFF coeff) DATA
                    DATA additional data
*              ( COEFF_NB_ISOTOPES name-compo) DATA
                    DATA additional data
```

where:

- `<PARTICLE_TYPE>` is `NEUTRON` or `PHOTON`.
- The available interaction codes are given in Tables 13.1 and 13.2.
- For the `DATA` choice, it is possible to multiply the values by a coefficient `coeff`. In the absence of the words `COEFF coeff`, the coefficient equals 1.

With COEFF_NB_ISOTOPES followed by the name of a composition, the coefficient is calculated internally and equals the number of atoms of the isotope in this composition for $10^{-24}$ cm$^3$.

Important warning for International Reactor Dosimetry Files (IRDF) reaction rates:
When requesting an IRDF reaction rate (from the related dosimetry nuclear data files), it is recommended to use DATA (instead of FICTIVE) option. Otherwise, if the energy grid of the reaction of interest is different from the total cross-section energy grid (which may happen only for these specific dosimetry files), cross-section values may be linearly interpolated on the missing energies of the total cross-section energy grid, which could lead to different reaction rate results.

Notes for Table 13.1:

- code 105 refers to the sum of all other neutron interactions that are not mentioned by another code (in order to be consistent with the total neutron cross-section).
- codes 40 and 41 refer to nuclide transmutation, outside the unresolved range for code 40, in the unresolved range for code 41. Both reaction rates can be requested with or without probability tables. Reaction rate 41 is always zero when probability tables are missing. When probability tables are used, the response function has zero values for energies outside the unresolved range. Conversely for response 40, when probability tables are used, the response function has zero values for energies in the unresolved range. The non-zero values are equivalent to the summation of reaction rates 58, 24, 25 and 26. Summation of both reaction rates 40 and 41 can be useful.
- code 42 refers to the sum of the following neutron scattering interactions: elastic, total inelastic (discrete and continuous), (n,n$\alpha$), (n,np) and the three thermal scattering interactions (codes 30 to 32).
- reaction rates available only from version 7 on (interaction codes 242000 to 242016, 53000 to 53049, 54000 to 54049, 55000 to 55049, 56000 to 56049, 57000 to 57049, 245, 543 and 552) cannot be requested with the execution option *-k* which disables the reading of cross-sections for these interactions (see chapter 1).

Notes for Table 13.2:

- code 142 refers to the sum of all other photonuclear interactions that are not mentioned by another code (in order to be consistent with the total photonuclear cross-section).
- code 145 refers to the neutron production rate from photonuclear reactions, defined as the sum of the $\nu_i * \sigma_i$ for the interactions $i$ producing photoneutrons (note that neutrons produced by photofission are not taken into account).
- the request of photonuclear reaction rates needs the keyword PHOTO_NUCLEAR in the SIMULATION command block (see chapter 22).

| NEUTRON INTERACTION | CODE |
|---|---|
| **Total** | 104 |
| **Elastic scattering** | 21 |
| **Discrete inelastic scattering** | 22 |
| **Level *i* disc. inel. scat., *(i) from 01 to 40*** | 220(i) |
| **Continuous inelastic scattering** | 23 |
| **n $\rightarrow N\cdot$n** | 24 |
| **n $\rightarrow$ 2n** | 242 |
| **Level i disc. n $\rightarrow$ 2n, *(i) from 00 to 16*** | 2420(i) |
| **n $\rightarrow$ 3n** | 243 |
| **n $\rightarrow$ 4n** | 244 |
| **n $\rightarrow$ 2n + $\alpha$** | 245 |
| **n $\rightarrow$ n + 2$\alpha$** | 272 |
| **n $\rightarrow$ n + $\alpha$** | 25 |
| **n $\rightarrow$ n + p** | 26 |
| **Discrete and continuous inelastic scattering** | 28 |
| **Coherent elastic scattering** | 30 |
| **Incoherent elastic scattering** | 31 |
| **Incoherent inelastic scattering** | 32 |
| **Fission** | 33 |
| $\nu\cdot\sigma_f$ | 34 |
| $\nu_{prompt}\cdot\sigma_f$ | 35 |
| $\nu_{delayed}\cdot\sigma_f$ | 36 |
| **Nuclide transmutation outside URR** | 40 |
| **Nuclide transmutation in URR** | 41 |
| **Total neutron scattering** | 42 |
| **n $\rightarrow$ $\gamma$ absorption** | 52 |
| **n $\rightarrow$ p absorption** | 53 |
| **Level *i* disc. n $\rightarrow$ p absorption, *(i) from 00 to 49*** | 530(i) |
| **n $\rightarrow$ 2p absorption** | 532 |
| **n $\rightarrow$ t absorption** | 54 |
| **Level *i* disc. n $\rightarrow$ t absorption, *(i) from 00 to 49*** | 540(i) |
| **n $\rightarrow$ t + 2 $\alpha$ absorption** | 543 |
| **n $\rightarrow$ $\alpha$ absorption** | 55 |
| **Level *i* disc. n $\rightarrow$ $\alpha$ absorption, *(i) from 00 to 49*** | 550(i) |
| **n $\rightarrow$ 2$\alpha$ absorption** | 552 |
| **n $\rightarrow$ $^3$He absorption** | 56 |
| **Level *i* disc. n $\rightarrow$ $^3$He absorption, *(i) from 00 to 49*** | 560(i) |
| **n $\rightarrow$ d absorption** | 57 |
| **Level *i* disc. n $\rightarrow$ d absorption, *(i) from 00 to 49*** | 570(i) |
| **Total absorption** | 58 |
| **Other interactions** | 105 |

Table 13.1: Data file interaction codes for neutron reactions.

| PHOTON INTERACTION | CODE |
|---|---|
| **Total** | 106 |
| **Coherent scattering** | 100 |
| **Incoherent scattering** | 101 |
| **Photoelectric effect** | 102 |
| **Pair production** | 103 |
| **Total photonuclear** | 130 |
| $\gamma \rightarrow N \cdot$**n** | 131 |
| $\gamma \rightarrow$ **2n** | 1312 |
| $\gamma \rightarrow$ **3n** | 1313 |
| $\gamma \rightarrow$ **n + p** | 132 |
| $\gamma \rightarrow$ **n +** $\alpha$ | 133 |
| $\gamma \rightarrow$ **n +** $2\alpha$ | 1332 |
| **Discrete and continuous** $\gamma \rightarrow$ **n** | 135 |
| **Discrete** $\gamma \rightarrow$ **n** | 136 |
| **Level *i* disc.** $\gamma \rightarrow$ **n, *(i) from 00 to 40*** | 1360(i) |
| **Continuous** $\gamma \rightarrow$ **n** | 137 |
| **Photofission** | 138 |
| $\nu \cdot \sigma_{photofission}$ | 139 |
| $\nu_{prompt} \cdot \sigma_{photofission}$ | 140 |
| $\nu_{delayed} \cdot \sigma_{photofission}$ | 141 |
| **Other photonuclear interactions** | 142 |
| **Photoneutron production rate** | 145 |

Table 13.2: Data file interaction codes for photon (photoatomic and photonuclear) reactions.

## 13.3 SUM OF REACTION RATES

It is possible to define a reaction rate as the sum of `nb(taux)` reaction rates on different nuclei. The syntax is as follows.

**REACTION** <PARTICLE_TYPE>
**NUCLEI nb(rate)**
**name(nucleus$_1$)**
Additional data
for nucleus 1
**name(nucleus$_2$)**
Additional data
for nucleus n

The additional data for a given nucleus are `COMPOSITION` etc., `FICTIVE` etc. or `DATA` etc. It is thus possible to sum the reaction rates of different nuclei. It is also possible to mix the different types of nuclei (`FICTIVE` for one nucleus, `DATA` for another, etc. But care must be exercised as `FICTIVE` gives microscopic cross-sections and `COMPOSITION` gives macroscopic cross-sections, which must not be mixed!).

- It is possible to use `NUCLEI 1` instead of `NUCLEUS`;
- Rather than requesting, for instance:

```
REACTION NEUTRON
NUCLEI 2
O16 FICTIVE 300 INTERACTION 1 22
O16 FICTIVE 300 INTERACTION 1 23
```

it is better to write:

```
REACTION NEUTRON
NUCLEUS
O16 FICTIVE 300 INTERACTION 2 22 23
```

This is because certain incompatibilities are detected in the second case and not in the first (for instance, the total cross-section cannot be added to a partial cross-section of the same material, see Tables 13.1 and 13.2).

## 13.4 DATA ADDITIONAL DATA

The keyword `DATA` introduces a cross-section definition.

Note: The cross-sections are input in the same manner as the tabulated source.
When defining cross-sections, the following information is mandatory:

| Interpolation mode | Associated code |
|---|---|
| **Histogram** | IN_NO_INTERPOLATION |
| **Linear** | IN_LIN_LIN |
| **Linear-log** | IN_LIN_LOG |
| **Log-linear** | IN_LOG_LIN |
| **Log-log** | IN_LOG_LOG |

Table 13.3: Possible Interpolation Types

- the number of `nb(points)` pairs defining the cross-sections for the (energy, cross-section) pairs;
- the number of intervals `nb(inter)` for which it is necessary to specify the type of interpolation (if two intervals are defined using the same interpolation mode, but they are discontinuous, then they are counted as two distinct intervals);
- the interpolation modes for the intervals;
- the cross-section values at the given energies.

The modes of interpolation and their codes are given in Table 13.3.

For a histogram input, the cross-section value in a `[E(inf),Esup]` interval is the one corresponding to `E(inf)`.
The meanings of the modes of interpolation are as follows (where $y$ is the cross-section data and $x$ the energy):

- `IN_NO_INTERPOLATION`
$$y = constant;$$
- `IN_LIN_LIN`
$$y = a * x + b;$$
- `IN_LIN_LOG`
$$y = a * \log(x) + b;$$
- `IN_LOG_LIN`
$$\log(y) = a * x + b;$$
- `IN_LOG_LOG`
$$\log(y) = a * \log(x) + b.$$

The user must indicate how to make the intervals corresponding to interpolation mode j coincide with the energies supplied.
Let `E(i)` be a list of the energies (in MeV) for which the cross-section is known, `i` varying between 0 and `nb(points)`-1. The list, with the cross-section values associated with each energy, must be in increasing order. After giving the mode of interpolation for interval `j`, the index of the upper energy limit `num[limsup(j)]` is given for this interval (the energy must, of course, belong to the list of `E(i)` and `num[limsup(j)]` must be between 1 and `nb(points)`-1).

- The first interpolation interval extends from `E(0)` to `E{num[limsup(1)]}`;
- The `j`-th interpolation interval extends from `E{num[limsup(j-1)]}` to `E{num[limsup(j)]}`;
- The last interpolation interval extends from `E{num[limsup(nbinter-1)]}` to `E{num[limsup(nbinter)]} = E[nb(points)-1]`.

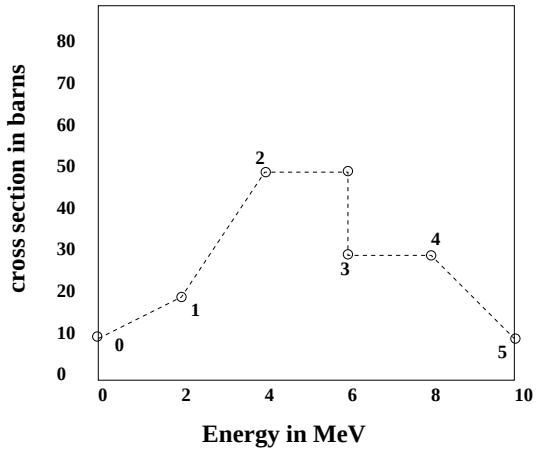
Finally, the `(E(i), section(i))` pairs are given, with `i` varying from 0 to `nb(points)-1`.
The syntax is as follows:

**DATA**
**nb(points) nb(inter)**
**<MODE_INTERPOLATION(1)> num[limsup(1)]**
**...**
**<MODE_INTERPOLATION(nbinter)> num[limsup(nbinter)]**
$\mathbf{E}_0$ **section**$_0$
**...**
$\mathbf{E}_{nb(points)-1}$ **section**$_{nb(points)-1}$

A file can also be used. It must contain all of the data of the previous syntax, starting with nb(points). The syntax is then:

**DATA**
**FILE filename**
where filename is the name of the data file.

## Example of a cross-section definition given using the keyword `DATA`




Thus, if the user wishes to define the cross-section shown in the figure above, six points need to be entered: (0,10), (2,20), (4,50), (6,30), (8,30),(10,10) (`nb(points)=6`). There are three (`nb(inter)=3`) interpolation intervals, i.e., (with number `num[limsup(j)]` of the last point in bold).

- interval [0,4] (from point 0 to point **2**), for which linear interpolation between the points is required;
- interval [4,8] (from point 2 to point **4**) with histogram interpolation;
- interval [8,10] (from point 4 to point **5**) with linear interpolation once again.

This gives:

```
DATA
6 3
IN_LIN_LIN 2
IN_NO_INTERPOLATION 4
IN_LIN_LIN 5
0.  10.
2.  20.
4.  50.
6.  30.
8.  30.
10.  10.
```

## 13.5 ADDITIONAL DATA FOR A WEIGHTED CURRENT

It is possible to weight each current tally by a function of the energy, defined by the user. The syntax is as follows:

**WEIGHTED_COURANT** <PARTICLE_TYPE>
**DATA**
*Additional data*

The command block DATA is the same as for the reaction rates (see section 13.4). Nevertheless, in the (energy, section) pairs, the "section" here actually refers to any variable weighting the current.
A special case of this kind of response is the calculation of energy balance: for that, the user only has to give to the "section" variable the same values as for the energy.

## 13.6 MACROSCOPIC REACTION RATES

Sometimes, computing macroscopic reaction rates including several isotopes and reactions might become cumbersome. For this reason, the response MACROSCOPIC_REACTION_RATE allows the user to compute some predefined macroscopic reaction rates by direcly averaging over the reactions contributing to the desired rate. The MACROSCOPIC_REACTION_RATE response is thus a macroscopic reaction rate summing all the isotopes in a given composition and all the reactions for a specific type.

The particle type must be NEUTRON, followed by the desired reaction type: TOTAL (all possible reaction types), CAPTURE (all non fissile absorptions), TOTAL_FISSION (all fission events), PROMPT_NUSIGF (prompt neutron production events), DELAYED_NUSIGF (delayed neutron production events), YIELD_SIGMA_N2N (yield times $(n,2n)$ events), YIELD_SIGMA_N3N (yield times $(n,3n)$ events) YIELD_SIGMA_N4N (yield times $(n,4n)$ events).
**WARNING !** The user is advised not to use macroscopic reaction rates in very diluted media or with very thin volumes (fission chambers, homogenized materials of very small volumes). In these cases, the collision-based estimator for macroscopic reaction rates might display a slow convergence, and the user should preferentially use the standard method to calculate reaction rates, as given in Sec. 13.2.

## 13.7 DEPOSITED ENERGY

This response computes the Kinetic Energy Release in MAtter (KERMA), i.e., the energy $E_D$ deposited by charged particles (emitted following collisions induced by a given particle type) in the traversed medium.
The particle type following the keyword DEPOSITED_ENERGY can be NEUTRON, PHOTON, ELECTRON or POSITRON: this defines the particle that is responsible for a specific contribution to the energy deposition.
The deposited energy $E_D$ depends on whether other particles than the requested particle type are simulated: by convention, it is assumed that, if a particle is transported, its associated energy is subtracted from the energy balance, since it will be deposited elsewhere. On the contrary, if a particle is not simulated, its energy is assumed to be deposited locally at the collision event. For instance, when the particle type is NEUTRON,

- in pure neutron tranport, the energy carried by the photons emitted in nuclear reactions is assumed to be deposited locally the reaction event: $E_D = E_D^{neutron}$
- in coupled neutron-photon tranport, the energy carried by the photons emitted in a given reaction is subtracted from the energy balance: $E_D = E_D^{neutron} - E_D^{photon}$.

If electrons and positrons are not simulated, their energy is analogously assumed to be deposited locally (this energy is usually smaller than that of neutrons or photons). For photon energy deposition, only the sum of the energies deposited by PHOTON, ELECTRON and POSITRON is meaningful.
When photons are transported, by default only photoatomic reactions contribute to the calculation of deposited energy. The user may be interested in including the contribution of photonuclear reactions to deposited energy: this is achieved by the use of the keyword PHOTO_NUCLEAR in the SIMULATION block.
For scores associated with the DEPOSITED_ENERGY response function, it is important to note that the energy grid associated with the energy deposition response function corresponds to the energy of the incident particle inducing the reaction. Therefore, the deposited energy varies as a function of the energy of the incident particle.

### 13.7.1 REACTION Q-VALUES FOR ENERGY DEPOSITION

The nuclear reaction Q-values (other than fission or photofission Q-values) are by default computed (from mass differences) by reading in the file giving the atomic weights of the nuclei. For some isotopes, however, this is not possible (or leads to wrong results): this is the case for

- $(n,2n)$ reaction of $^{9}$Be: this leads to a resonant nucleus $^{8}$Be that immediately decays to 2α. For a proper calculation of the deposited energy, the associated Q-value must be read in the evaluation file
- the Q-values of metastable nuclei (e.g., $^{110M}$Ag) can not be computed based on mass differences and must be read in the evaluation file
- the Q-values of natural elements (e.g., $C-NAT$) can not be computed based on mass differences and must be read in the evaluation file.

For $(n,2n)$ reaction of $^{9}$Be, the user can resort to the dictionary file and specify a Q-value flag 2 so as to read the reaction Q-value in the evaluation file (see chapter 3). This is the option by default in the dictionary file. Analogously, for metastable isotopes, the user can resort to the dictionary file and specify a Q-value flag 1 so as to read the reaction Q-values in the evaluation file (see chapter 3). This is the option by default in the dictionary file. Finally, for natural elements, TRIPOLI-4® reads the Q-values in the evaluation file.

### 13.7.2 FISSION ENERGY

For neutron-induced fission, we adopt the convention that in `SHIELDING` mode the secondary fission neutrons (prompt and delayed) are not transported, so that their energy is assumed to be deposited locally, whereas in `FIXED_SOURCES_CRITICALITY` or `CRITICALITY` the secondary neutrons are transported and their energy is therefore subtracted from the energy balance.
By default, the fission energy is given by the sum of prompt and delayed contributions, so as to preserve the same conventions as in TRIPOLI-4® version 7.
Finally, the user interested in computing the fission energy alone can do so by resorting to the optional keyword `FROM_FISSION` in the response `DEPOSITED_ENERGY`. This keyword can be used in addition to either `ONLY_PROMPT_ENERGY`, `ONLY_PROMPT_GAMMA_ENERGY`, or `RESCALE_DELAYED_FISSION_PHOTONS`, that are global options of the `SIMULATION` block (see Sec. 22).
Neutron-induced fission Q-values are by default read in the file giving the energies released by fissile isotopes during fission for the different particles emitted (see chapter 3). If this information is missing, to perform a deposited energy calculation the user must provide himself a fission Q-value (in MeV) for the isotope. For photofissions (if any) a photofission Q-value must be provided as well to perform a deposited energy calculation when the keyword `PHOTO_NUCLEAR` is specified in the `SIMULATION` block. This is achieved by resorting to a separate block in the form
This block can be placed at any position in the input file. Here `NEUTRON` is the keyword that defines neutron-induced fissions, and nb_isotopes_for_neutron_induced_fissions is the number of isotopes for which fission energy is provided by the user. Then, isotopei is the name of the *i*th isotope and fission_energyi is the corresponding fission energy in MeV. Analogously, `PHOTON` is the keyword that defines photofissions, and nb_isotopes_for_photofissions is the number of isotopes for which photofission energy is provided by the user. In absence of precise information, a fission (or photofission) Q-value of 200MeV should provide a rough order-of-magnitude of the energy release in matter.

```
QFISSION
                NEUTRON nb_isotopes_for_neutron_induced_fissions
                        isotope1 fission_energy1
                        isotope2 fission_energy2
                ...
                PHOTON nb_isotopes_for_photofissions
                        isotope1 fission_energy1
                        isotope2 fission_energy2
                ...
END_QFISSION
```

## 13.8 KINEMATIC LIMITS

The response KINEMATIC_LIMITS is provided to guide the user in the estimation of the deposited energy in specific coupled neutron-photon calculations. When the user demands the energy deposited by neutrons in such coupled calculations, the energy carried away from photons is subtracted, so that two major problems may arise: first, the quantity to be estimated is generally affected by large fluctuations (which stems from being basically the difference between the available energy, $E+Q$, and the energy of the photons, which are usually of comparable order of magnitude); second, inconsistencies in the nuclear data libraries may become more apparent in this configuration, since energy or momentum balance errors (typically in the secondary photon productions) are emphasized when taking these differences. The KINEMATIC_LIMITS response allows the users to compute the energy deposited by neutrons in coupled neutron-photon calculations without actually having to simulate photons, based on the laws of kinematics: as such, this response may

- provide a robust estimation of the deposited energy, only minimally dependent on nuclear data
- serve as a tool for nuclear data analysis/verification.

In the former case, the user computes kinematic limits as an alternative to the DEPOSITED_ENERGY response in a coupled neutron-photon transport, which is based on energy balance and thus dependent on nuclear data libraries. In the latter, a comparison between the KINEMATIC_LIMITS and the DEPOSITED_ENERGY responses (on the very same configuration) provides the user with test of consistency of the nuclear data library under examination.
The particle type must be NEUTRON, and it must be followed by the kinematic limit type, which is UPPER_LIMIT (upper boundary) or LOWER_LIMIT (lower boundary). The basic coherence verification is that the DEPOSITED_ENERGY response for neutrons in a coupled neutron-photon transport lies above the LOWER_LIMIT and below the UPPER_LIMIT of the KINEMATIC_LIMITS response. Whenever these boundaries are not respected, the deposited energy response obtained from DEPOSITED_ENERGY in coupled neutron-photon transport might become questionable because of possible inconsistencies in nuclear data libraries. Remark that these constraints hold only for the responses as a function of the incident neutron energy, and do not generally apply also to energy-integrated values. Observe that KINEMATIC_LIMITS depends only on neutron transport parameters, and this response will be the same whether neutron only or neutrons and photons

are simulated: we stress that KINEMATIC_LIMITS yields the energy deposited by neutrons as if photons were transported.
For scores associated with the KINEMATIC_LIMITS response function, it is important to note that the energy grid associated with the KINEMATIC_LIMITS response corresponds to the neutron incident energy, exactly as in the case of DEPOSITED_ENERGY.
The $(n,2n)$ reaction of $^{9}$Be leads to a resonant nucleus $^{8}$Be that immediately decays to 2α. For a proper calculation of the kinematic limit, the user can resort to the dictionary file and specify a Q-value flag 2 so as to read the reaction Q-value in the evaluation file and at the same time adopt special kinematics for this reaction (see chapter 3). This is the option by default in the dictionary file.

## 13.9 DPA

This response computes the number of displaced atoms per unit time (DPA), in agreement with the specifications in the HEATR module of the NJOY code [36, 35]. The available energy $E_a$ per unit time, i.e., the energy that is available to create an atomic displacement is computed via the model proposed by Norgett, Robinson and Torrens (NRT). The NRT model roughly says that $E_a$ is proportional to the primary knoked-on atom (PKA) energy, and has a nonlinear saturation at higher energies.
DPA response is meaningful only for neutrons. If coupled neutron-photon calculations are performed, the response will be the same as in the case of a pure neutron transport: photonuclear reactions are ignored. The effects of $(n,\gamma)$ reactions are nonetheless taken into account through the residual nucleus recoil energy.
The particle type following the keyword DPA must be NUCLEUS.
For DPA calculations, the user must specify the displacement (threshold) energies $E_d$ associated to any material composition for which a DPA is requested. In its simplest definition (linear response), the DPA response is an available energy $E_a$ per unit time, divided by $2E_d$ : the result

$$DPA = \frac{E_a}{2E_d} \tag{13.1}$$

carries therefore the units of (displaced atoms)/s.
The displacement energies $E_d$ (in MeV) must be provided with a separate block in the form

```
DISPLACEMENT_ENERGY
        COMP1 nb_isotopes_for_comp1
                isotope1 displacement_energy1
                isotope2 displacement_energy2
        ...
        COMPN nb_isotopes_for_compN
                isotope1 displacement_energy1
                isotope2 displacement_energy2
        ...
END_DISPLACEMENT_ENERGY
```

This block can be placed at any position in the input file. Here *COMPi* is the name of the *i*th composition, and *nb_isotopes_for_compi* is the number of isotopes for this composition for which a displacement energy is provided; then, isotopei is the name of the *i*th isotope, and displacement_energyi is the corresponding displacement energy in MeV. If, for a given isotope of a composition for which a DPA is requested, the displacement energy $E_d$ is not given, it is assumed that this isotope does not contribute to the creation of displacements.

| **Element** | $E_d$ [eV] | **Element** | $E_d$ [eV] |
|---|---|---|---|
| *C* graphite | 28-31 | *C* diamond | 80 |
| *Si* | 11-22 | *Ge* | 12-30 |
| *GaAs* | 9 (*Ga*) 9 (*As*) | *ZnS* | 10 (*Zn*) 15 (*S*) |
| *CdS* | 7 (*Cd*) 9 (*S*) | *Cu* | 19-29 |
| *Al* | 19-27 | *Ni* | 23-33 |
| *Pt* | 33-44 | *SiC* | 45-90 |
| *MgO* | 64 (*Mg*) 60 (*O*) | $Al_2O_3$ | 18 (*Al*) 76 (*O*) |
| $MgAl_2O_4$ | 56 (*O*) | *ZnO* | 40 (*Zn*) 57 (*O*) |
| $UO_2$ | 40 (*U*) 20 (*O*) | *BeO* | 76 (*O*) |
| *Fe* | 17-44 | *Mo* | 33-65 |
| *Zn* | 19-29 | *Au* | 35-49 |
| *Be* | 31 | *Co* | 40 |
| *Mg* | 25 | *Zr* | 40 |
| *Nb* | 40 | *Ca* | 40 |
| *V* | 40 | *Ta* | 90 |
| *Cr* | 40 | *W* | 90 |
| *Mn* | 40 | *Pb* | 25 |

Table 13.4: Typical values for displacement energies $E_d$.

A modified saturation function for DPA calculations can be obtained by resorting to the optional keyword KINCHIN_PEASE_MODEL, following which the modified Kinchin Pease model is applied instead of the linear response defined above: in this case, the number of displacements per atom is defined as

- $DPA = 0$ for $E_a < E_d$,
- $DPA = 1$ for $E_a < 2.5E_d$,
- $DPA = 0.8\,E_a/(2E_d)$ otherwise.

Typical values of the displacement energies $E_d$ are given in Table 13.4 (we refer the reader to the HEATR documentation for more information [35, 36]). In absence of precise information, a displacement energy of $E_d = 25 \cdot 10^{-6}$MeV should provide a rough order-of-magnitude for performing DPA calculations.
The $(n,2n)$ reaction of $^9$Be leads to a resonant nucleus $^8$Be that immediately decays to 2α. For a proper calculation of the DPA, the user can resort to the dictionary file and specify a Q-value flag 2 so as to read the reaction Q-value in the evaluation file and at the same time adopt special kinematics for this reaction (see chapter 3). This is the option by default in the dictionary file.
For scores associated with the DPA response function, it is important to note that the energy grid associated with the DPA response corresponds to the neutron incident energy. Therefore, DPA vary as a function of the energy of the incident neutron.

By default, the DPA response is summed over all atoms in the composition, and carries the units of total displacements in the volume (per unit time). The optional keyword NORMALIZE_TO_ONE_ATOM can be used to express DPA in units of displacements per atom (per unit time), by normalizing to the number density in the volume.

## 13.10 GAS PRODUCTION

The production of light particles (the isotopes of hydrogen and helium, also called gas particles) may affect the structural properties of traversed materials, and are of utmost importance for radiation shielding (especially in the case of tritium). The GAS_PRODUCTION response allows the user to compute the production rate of such particles, coming from neutron and photon contributions. In both cases, we consider direct reactions (a gas particle in the exit reaction channel) as well as indirect reactions (the gas particle is the recoil nucleus itself: this happens only for light target nuclei). Photons can induce gas production only via photonuclear reactions, which demand the optional PHOTO_NUCLEAR keyword in the SIMULATION block; otherwise, the gas production rate from photon reactions will be zero.
The particle type following the keyword GAS_PRODUCTION must be NEUTRON or PHOTON. The user can specify the gas particle type by using the keyword: NEUTRON ($^1H$), DEUTERON ($^2H$), TRITON ($^3H$), HELIUM ($^3He$), or ALPHA ($^4He$).
For fissile nuclei, gas production typically comes from ternary fission leading to light nuclei: special nuclear data for ternary fission ($MF8$, $MT454$) have been included in the Q-fission folder for convenience.
The $(n,2n)$ reaction of $^9$Be leads to a resonant nucleus $^8$Be that immediately decays to 2α. For a proper calculation of the gas production, the user can resort to the dictionary file and specify a Q-value flag 2 so as to adopt special kinematics for this reaction (see chapter 3). This is the option by default in the dictionary file.
For scores associated with the GAS_PRODUCTION response function, it is important to note that the energy grid associated with the GAS_PRODUCTION response corresponds to the neutron or photon incident energy. Therefore, the gas production varies as a function of the energy of the incident particle.

## 13.11 FISSION SPECTRUM

The response FISSION_SPECTRUM allows the user to compute the fission energy spectrum, averaged over all isotopes in a given composition. The particle type following the keyword FISSION_SPECTRUM must be NEUTRON. The kind of desired spectrum is to be defined by the user: available keywords are TOTAL (sum of prompt and delayed contributions), PROMPT (only prompt component) and DELAYED (only delayed component). The integral of the TOTAL sums up to the PROMPT plus DELAYED components. Moreover, the integral PROMPT component equals the total prompt β (prompt from prompt plus prompt from delayed), whereas the integral DELAYED component equals the total delayed β (delayed from prompt plus delayed from delayed). Prompt and delayed components are expressed in units of [$10^5$ pcm]. See the section 22.6 for more details about effective-β calculations.

## 13.12 FIXED SOURCE EMISSION

The response FIXED_SOURCE_EMISSION enables to check the energy and/or angle source distribution as simulated by the code (the associated energy and/or angle grids have to be defined in the GRID_LIST command block, see chapter 14). This response is available for fixed source simulations only. The origin of the delivered information differs from the source normalization information given in the results file at the end of the initialization step : the information delivered by the present response is collected by an estimation calculated from the real sampling of source particles during the simulation. When source biasing options are used (see subsections 12.3.4 and 12.4.7), this response allows the user to check the resulting simulated source distribution. The syntax is as follows:

**FIXED_SOURCE_EMISSION** <**particle_type**> <**mode**>

where <particle_type> is the particle type and <mode> is either DENSITY or FREQUENCY.
When the keyword DENSITY is chosen, the distribution of the sampled source is estimated. The mean value can be compared with the source norm, as given by the code in the initialization step (and output in the results file as "SIMULATION INTENSITY"). If no variance reduction technique and no source biasing is used, the variance of the score associated to this response is null.
When the keyword FREQUENCY is chosen, the information delivered is the frequency of the sampling and enables to check for example that certain energy groups enhanced by the user through a source energy biasing (see subsection 12.4.7) are sampled more frequently. If no variance reduction technique and no source biasing is used, both options DENSITY and FREQUENCY are equivalent. Otherwise the mean value of the score associated with the FREQUENCY option response can be compared with the biased source norm, as given by the code in the initialization step (and output in the results file as "BIASED SIMULATION INTENSITY").

## 13.13 DEPOSITED CHARGE

The DEPOSITED_CHARGE response is used to calculate the spectrum of charge deposition in a given volume. Note that this makes sense only if charged particles (ELECTRON and POSITRON) are transported at all; no charge deposition can occur otherwise.
The particle type for this response must be ATOM.
For scores based on the DEPOSITED_CHARGE response function, the associated grid has the meaning of charge-deposition grid. Therefore, if one wishes to estimate the average deposited charge in a given volume, the grid must span negative and positive values, or else the final result will be biased. It is recommended to use unit bins centered on integer values.
In addition, each bin of the resulting histogram contains the product of the charge deposition frequency and the bin charge. Specifically, if a bin ranging from $-2.5$ to $-1.5$ contains a score result of $-0.6$ charge/(cm$^3$ source), it means that the simulation resulted in a $-2$ charge deposition in 30% of the histories (assuming unit volume for the sake of simplicity). This is done so that the average charge deposition can be straightforwardly calculated as the integral of the histogram. The average values are available for the user in the score output. Note also that the “score/lethargy” output values have no meaning for this response.
Finally, note that charged source particles are not tallied by this response function.

## 13.14 SPECIAL OPTION TO EXCEPT A PART OF THE CONTRIBUTION FOR SECONDARY PARTICLES

For a given response, the user has the possibility to except the tally contributions of secondary particles produced by certain interactions (for example the user can request a photon response in a coupled neutron-gamma simulation with a neutron source, excepting the contribution of the photons produced by fission). Note that those secondary particles are nevertheless simulated and can contribute to other tallies for which this option is not activated. The syntax in the RESPONSES command block for the current response is:

```
        <response_type> <particle_type> <additional_data>
#               NOT_INDUCED_BY_INTERACTION  nb(interactions) code_1 ... code_n
#               INDUCED_BY_INTERACTION  nb(interactions) code_1 ... code_n
```

where the nb(interactions) interaction codes are chosen among those listed in Tables 13.1 and 13.2.
When the keyword NOT_INDUCED_BY_INTERACTION is chosen, the secondary contributions of the following interactions are excepted, whereas when the keyword INDUCED_BY_INTERACTION is chosen, secondary contributions are taken into account only for the interactions mentioned.
Possible particle types are NEUTRON and PHOTON.
Calculations in several steps are not compatible with this option (e.g. KEEP_RESULT option, Green function files, collision files, source files ...).

**Response definition (optional) for the criticality calculation example of chapter 5**

```
RESPONSES 4

NAME FLUX_NEUTRON
FLUX NEUTRON

NAME U238_TOTAL_REACTION_RATE_NEUTRON
REACTION NEUTRON
NUCLEUS U238 COMPOSITION SOLUTION_U
INTERACTION 1 104

NAME O16_TOTAL_REACTION_RATE_NEUTRON
REACTION NEUTRON
NUCLEI 2
O16 FICTIVE 300 INTERACTION 1 22
O16 FICTIVE 300 INTERACTION 1 23

NAME BIDON_REACTION_RATE_NEUTRON
```

```
REACTION NEUTRON
NUCLEUS BIDON
DATA
6 3
IN_LIN_LIN 2
IN_NO_INTERPOLATION 4
IN_LIN_LIN 5
0.  10.  2.  20.
4.  50.  6.  30.
8.  30.  10.  10.

END_RESPONSES
```

# Chapter 14

# DEFINING GRIDS

Grids are defined between the keywords `GRID_LIST` and `END_GRID_LIST`.

Grids need to be specified for tallying and homogenization. A grid is a set of energy, angle, or time intervals, a response function is calculated for each of those intervals. A grid can also be used for ranges of number of fission neutrons.

A list of grids is defined using:

- the number of grids
- the data for each grid

A grid is defined using:

- the name of the grid
- the number of bounds for the grid
- the values of the bounds

For a pre-defined grid, the number of bounds and their values need not be indicated. The grids are referred to by their names: ***3_GRS*** (3 groups), ***SAILOR*** (47 groups), ***APOLLO_94_GAMMA*** (94 groups from 5 keV to 20 MeV, for gamma transport), ***APOLLO_99*** (99 groups), ***GAM2*** (100 groups), ***APOLLO_172*** (172 groups), ***APOLLO_281*** (281 groups, SHEM) and ***TRIPOLI_315*** (315 groups). The TRIPOLI-4® code adds a group above the highest energy of the grid up to $10^4$ MeV and a group below the lowest energy of the grid down to $10^{-11}$ MeV (if necessary) (in order to overlap the simulation domain in every case).

All of the intervals comprising an energy grid must cover the simulation domain of the problem (`[Einf,Esup]` by default $10^{-11}$ MeV - 20 MeV for neutron). Otherwise, the TRIPOLI-4® code fails. For known grids, the additional high-energy and low energy groups guarantee that the upper and lower boundary of the simulation domain are within the grid.

The grids are defined for any type of particle (`NEUTRON`, `PHOTON`, `ELECTRON` and `POSITRON`).

Linear or log spacing grids can be defined automatically using the keywords IN_LIN or IN_LOG. For example $v1$ IN_LIN $n$ $v2$ defines $n$ values linearly interpolated between $v1$ and $v2$, the total number of values is then $n+2$.
The sequences $v1$ IN_LIN $n$ $v2$ and $v1$ IN_LOG $n$ $v2$ can be inserted everywhere in grid, the total bound number must take into account all the interpolated values.

The grids for energy, angle, time and number of fission neutrons are defined in the same way. A keyword is not necessary to define the type of grid.

Important: there is one more bound than the number of intervals. A grid can be input with increasing or decreasing values, but the list must be **monotonic** (always increasing or always decreasing).

The grid definition syntax is, therefore, as follows:

**GRID_LIST**
**nb(grids)**
**name(grid$_1$)**
**nb(bounds)**
**bound(1)...bound(n)**
...
**name(known_grid$_i$)**
...
**name(grid$_j$)**
**nb(bounds)**
**bound(j$_1$)** ...
**bound(j$_k$) IN_LIN m$_k$ bound(j$_{k+m_k+1}$)** ...
**bound(j$_l$) IN_LOG m$_l$ bound(j$_{l+m_l+1}$)** ...
**bound(j$_n$)**
...
**name(grid$_n$)**
**nb(bounds)**
**bound(1)...bound(n)**
**END_GRID_LIST**

## Defining grids for the criticality calculation of chapter 5

```
GRID_LIST 6
          GRID_FLUX 10
                    20.  5.  1.  0.1 0.01
                    1.E-3 1.E-4 1.E-5 1.E-6 1.E-11
          GRID_ABSORPTION 6
                    20.  4.086771E-2 9.119000E-3
                    1.433817E-3 1.101044E-4 1.E-11
          GRID_TIME 5
```

```
                1.  0.1 0.01 1.E-5 1.E-7
        GRID_ANGLE 5
                -1.  0.5 0.  0.5 1.
        GAM2
        GRID_NU 11
                0.  0.9999 1.9999 2.9999 3.9999
                4.9999 5.9999 6.9999 7.9999 8.9999 9.9999
        GRID_INTERPOL 15
                1.E-11 1.E-10 IN_LOG 4 1.  1.5 2.  IN_LIN 5 20.
END_GRID_LIST
```

The third grid in the above example is a time grid. It is, of course, not subject to the energy simulation domain constraints.

Angle grids such as the grid GRID_ANGLE are given in terms of cosines and must cover the domain [-1, 1]. Cosine is the angle between particle's vector ω and the normal vector of the surface crossed by the particle. The normal vector is the one used to build the surface. It means that it could be the opposite direction of the out going normal direction of the volume that surface belongs.

Grids such as GRID_NU are used for ranges of number of fission neutrons and are not subject to the energy simulation domain constraints since they should encompass the zero value (they are used with the `WHOLE_GRID` keyword instead of the standard `GRID` keyword for energy grid, see section 15.7).

The last grid GRID_INTERPOL is equivalent to :

```
GRID_INTERPOL 15
        1.E-11 1.E-10 1.E-8 1.E-6 1.E-4 1.E-2 1.  1.5 2.  5.  8.  11.  14.  17.  20.
```

Important:

- The time (or angle) specification grid is optional.
- Angular grid can only be used with surface-based estimators (see section 15.2).
- When a grid for the number of fission neutrons is used, it replaces the use of the energy grid (see Section 15.7), with the difference that it can encompass the zero value.

# Chapter 15

# DEFINING SCORES

Scores are defined between the keywords SCORE and END_SCORE. The syntax is as follows:

**SCORE**
**nb(scores)**
Descriptions of scores
**END_SCORE**

## 15.1 NOTES REGARDING SCORE DEFINITIONS

A score is defined using:

- the type of response function to be calculated, designated by an integer corresponding to the order of appearance of the response in the RESPONSES command or by its name if it was given in the RESPONSES command;
- an energy grid, designated by its name in the GRID_LIST command;
- an optional angle grid designated by its name in the GRID_LIST command;
- an optional time grid (only used in kinetic calculations) also designated by its name in the GRID_LIST command;
- an estimator defined in the SCORE command.

## 15.2 STANDARD MONTE CARLO ESTIMATORS

### 15.2.1 VOLUME-BASED ESTIMATORS

For volume tallies, the following estimators are available:

- the track length estimator `TRACK`

$$\chi = \omega \cdot L$$

where:

  - $\omega$ is the statistical weight of the particle;
  - $L$ is the length of the trajectory.

- the collision estimator `COLL`

$$\chi = \frac{\omega}{\Sigma(E)}$$

where:

  - $\omega$ is the weight of the particle;
  - $\Sigma(E)$ is the total cross-section of the composition.

### 15.2.2 SURFACE-BASED ESTIMATORS

The following estimator is available for a tally on a surface:

- the surface-based estimator `SURF`.

For a flux:

$$\chi = \frac{\omega}{\|\Omega \cdot \mathbf{n}\|}$$

For a current (outgoing current):

$$\chi = \omega$$

where:

- $\omega$ is the weight of the particle;
- $\Omega$ is the direction of the particle;
- $\mathbf{n}$ is the surface normal unit vector.

## 15.3 NEXT-EVENT ESTIMATORS

### 15.3.1 POINT-BASED ESTIMATORS

For tallying at a point (flux or reaction rate), the point-based estimator `FLUX_PT` is available with a point definition.

### 15.3.2 SURFACE-BASED ESTIMATORS

For tallying at a surface (flux or reaction rate), the estimator `FLUX_PT` must be used with a target surface definition.

### 15.3.3 REMARKS

At the present time, the next-event estimators are only unbiased for vacuum boundary conditions. They can be used with *anisotropic* sources.
Next-event estimators are performing a full virtual transport from each collision to one of the scoring points. If the number of collision is important (as in some neutron calculations), it can be very time consuming.

## 15.4 VARIANCE EVALUATION

For each score, a mean value is estimated from the simulation. A standard deviation of this mean is also given to estimate the fluctuations of the score. Since version 12, the Welford algorithm is used in place of the standard algorithm (see [46, 101]) to evaluate the variance (and then the standard deviation).

## 15.5 GEOMETRICAL ELEMENTS

Such elements can be volumes (volume-based estimators), surfaces (surface-based estimators), points (point-based estimators) or a regular spatial mesh tally.

- A volume is designated by its definition number.
- A surface is designated by the interface between two volumes. An escape surface is designated by a volume number and virtual number -1. In the case of a target surface with a next-event estimator, a special syntax will be given below.
- A point is designated by its coordinates $(X,Y,Z)$.
- A spatial mesh independent of the geometry is defined with a syntax detailed below.

Important:

- **a surface cannot be defined by a virtual volume** (the result would be a zero score; the test is made on the basis of the particle belonging to a volume and a particle cannot belong to a virtual volume);
- for the `CURRENT` option, it is **an exiting current from the first volume given towards the second volume** that is calculated. It is not a net value, and only the particles leaving the first volume and entering the second are counted.
- for the `FLUX` with a surface estimator, see remarks on the associated geometrical element types in subsection 15.7.

## 15.6 FILTERING

It is possible to tally only the results for particles that have passed through certain volumes (conditional tallying or filtering). In this case, it is necessary to use the `FILTER` option.
Note: This option is incompatible with the collision band command.

## 15.7 DEFINING THE SCORE

A score is defined using the following syntax:

```
        NAME <score_name>
*1      num(response)
*1      <response_name>
            <ESTIMATOR_NAME>
            FILTER  VOLU
                    LIST  nb(volu)
                          vol(1)... vol(n)
            FILTER_BY_COMPOSITION  nb(compo) compo(1) ... compo(nb)
            TIME  GRID  name(time grid)
#           ANGULAR  GRID  name(μ angle grid)
#           2D_ANGULAR  GRID  name(μ angle grid) name(ϕ angle grid)
*2          GRID  name(energy grid)
*2          WHOLE_GRID  name(grid encompassing 0)
*2          2D_GRID  min1 max1 min2 max2
            <RESULTS>
            STORAGE
                    filename
                    LIMIT_SIZE nb_Mo
            GAUSSIAN_BROADENING  a b c
            <GEOMETRICAL_ELEMENT_TYPE>
                    STORE_IN_FILE
                    <LIST_OR_SUM> nb(elements)
                          element(1)
                          ...
                          element(n)
```

where:

1. The score name is optional. In the case of a neutronic-shielding coupling calculation (see also chapter 22), this data must be given in order to determinate in which simulation mode this score is to be performed.

2. The associated response is given either:

   (a) by `num(response)`, the response number given in the `RESPONSES` command: the first response defined is number 1, the second number 2, etc.
   Important: Number n is not specifically indicated in the command; adding a response can offset the numbering.

   (b) by the name of the response given in the `RESPONSES` directive (by `NAME`)

3. $< ESTIMATOR_NAME >$ is one of the following keywords:

- TRACK for a track length estimator
- COLL for a collision estimator
- SURF for a surface estimator
- FLUX_PT <*score_choice*> for a point estimator
  <score_choice> is an optional parameter which specifies the scored data. It can be :
  - COLLIDED all contributions to flux point have at least one collision.
  - UNCOLLIDED all contributions to flux point come directly from source without collision.
  - TOTAL all contributions are take into account, this is the default.
- MACROSCOPIC_REACTION_RATE for macroscopic reaction rates. This mode of estimation is necessary for scores associated with MACROSCOPIC_REACTION_RATE response.
- LOCAL_ENERGY_DEPOSITION for energy deposition. This mode of estimation is necessary for scores associated with DEPOSITED_ENERGY response.
- KINEMATIC_LIMITS for kinematic limits. This mode of estimation is necessary for scores associated with KINEMATIC_LIMITS response.
- DPA for calculation of Displacements Per Atom (DPA). This mode of estimation is necessary for the scores associated with DPA response.
- PRODUCTION for production of particles. This mode is necessary for scores associated with PRODUCTION response functions (PHOTON_REPARTITION and ELECTRON_REPARTITION keywords from versions earlier than 7 are still allowed with the same keyword in response command block, and are actually particular cases for the calculation of a photon source or an electron source, see chapter 13).
- GAS_PRODUCTION for calculation of gas production. This mode of estimation is necessary for the scores associated with GAS_PRODUCTION response.
- FISSION_SPECTRUM for calculation of fission spectra. This mode of estimation is necessary for the scores associated with FISSION_SPECTRUM response.
- DEPOSITED_SPECTRUM for detector response calculation. This estimation mode is necessary for scores associated with the DEPOSITED_SPECTRUM_WEIGHTED_ENERGY or the DEPOSITED_SPECTRUM response functions.
- FIXED_SOURCE_EMISSION for fixed source emission calculations. This mode of estimation is necessary for scores associated with FIXED_SOURCE_EMISSION response. Note that this score does not need any geometrical element definition since it applies to the whole source. The request of this type of score ends up then with a grid name.
- DEPOSITED_CHARGE for the calculation of the charge-deposition response. This estimation mode is necessary for scores associated with the DEPOSITED_CHARGE response function.
- NU_FISSION for the calculation of a histogram of number of fission neutrons per fission. This mode of estimation is necessary for the scores associated with the NU_FISSION response. In addition, the WHOLE_GRID keyword must be used instead of the standard GRID energy-grid keyword, so as to encompass the 0 value.

- FISSION_N_SUM_SPECTRUM for the calculation of the sum of fission neutron energies per fission. This mode of estimation is necessary for the scores associated with the FISSION_N_SUM_SPECTRUM response. In addition, the WHOLE_GRID keyword must be used instead of the standard GRID energy-grid keyword, so as to encompass the 0 value.
- FISSION_ZA_DISTRIBUTION for the calculation of the (Z,A) distribution of the fission products (after emission of prompt neutrons). This mode of estimation is necessary for the scores associated with the FISSION_ZA_DISTRIBUTION response. In addition, the 2D_GRID keyword must be used for the definition of the (Z,A) 2D mesh with a cell size of 1 in Z and A, and with Zmin, Zmax, Amin and Amax bounds respectively defined with $\min_1$ $\max_1$ $\min_2$ $\max_2$ values. The end of the syntax is the standard one with volumes as geometrical elements (followed by <LIST_OR_SUM>) .

4. name(time grid) (or name($\mu$ angle grid) or name($\phi$ angle grid)) is the name of a time (or angle) grid given in GRID_LIST. The distinction between energy and time (or angle) grids is made using the keyword TIME (or ANGULAR or 2D_ANGULAR), preceding GRID. Note: The time (or angle) grid specification is optional.

   Important:

   - ANGULAR refers to $\mu$ angle, where $\mu$ is defined with repect to the normal to the surface used with the surface-based estimator SURF.
   - 2D_ANGULAR refers to both : $\mu$ and $\phi$, defined in the global frame of the geometry.
   - The name of the $\phi$ grid (e.g. : DEC_PHI_ANGLE) must then contain the string “PHI".

5. name(energy grid) is the name of an energy grid given in GRID_LIST.

6. name(grid encompassing 0) is the name of a grid, given in GRID_LIST, replacing the energy grid and encompassing the zero value. For example it can be used for a NU_FISSION or a FISSION_N_SUM_SPECTRUM response.

7. <RESULTS> is an optional keyword which can be BOOTSTRAP for confidence interval calculations or COMPLETE for complete storage of the results, or VOV for "on the fly" information on variance jumps. By default, the results stored are average values. The utilisation of BOOTSTRAP or COMPLETE options can take a lot of memory. Indeed, a floating point variable is allocated for each batch, each score and each energy group. These options may cause problems in parallel mode.

8. <FILTER> is an optional keyword indicating that a conditional tally is required. It can be one of the following keywords:

   - FILTER_OR: the particle must have previously passed through one of the specified volume list;
   - FILTER_AND: the particle must have previously passed through every volumes of the specified volume list;
   - FILTER_SEQ: the last volumes through which the particle has passed must be equal to the specified list in the specified order;
   - FILTER_TARGET: the direction of the particle must have crossed one of the specified volume list.

If the FLUX_PT estimator has been chosen, only FILTER_OR or FILTER_TARGET are available.

9. FILTER_BY_COMPOSITION is an optional keyword to be used with extended mesh geometrical elements. It allows to filter scoring events by the belonging to a list of compositions. All events occurring in a composition cited in this list will be used for scoring in the extended mesh.

10. STORAGE is an optional keyword only available with the FLUX_PT estimator and the TARGET_SURF geometrical element type (see below). Characteristics of the particles reaching the surface defined by TARGET_SURF are then stored. Similarly to the STORAGE command (see chapter 27), the user must provide a file name which will be suffixed by "_<i>_fxpt_target_stock". The variable $i$ is initialized to 1 and incremented each time the limit size nb_Mo (in Mega bytes) is reached. (The storage is then going on in a new file.)

11. GAUSSIAN_BROADENING is an optional keyword to be used together with DEPOSITED_SPECTRUM scores. It allows for Gaussian Energy Broadening (GEB), so as to approximate the behaviour of real-life radiation detectors. Indeed, the energy peaks in a physical radiation detector suffer a broadening effect which can be reproduced through Gaussian sampling.
Whenever a contribution of energy $E$ is to be deposited in the spectrum, it is replaced by an energy $E^*$ sampled from the Gaussian of mean $E$:

$$f(E^*|E,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(E^*-E)^2}{2\sigma^2}}$$

where the standard deviation $\sigma$ of the Gaussian is:

$$\sigma = \frac{\text{FWHM}}{2\sqrt{2\ln 2}}$$

The FWHM (Full Width at Half Maximum) is used to describe the physical detector behaviour. It is computed using three parameters $a$, $b$ and $c$ provided by the user, according to:

$$\text{FWHM} = a + b\sqrt{E + cE^2}$$

12. <GEOMETRICAL_ELEMENT_TYPE> is one of the following keywords:

    - VOLUME or CELL, to be used with the track length estimator TRACK and the collision estimator COLL;
    - FRONTIER or ORIENTED_FRONTIER, to be used with the SURF estimator.
      When ORIENTED_FRONTIER is used together with FLUX, only the exiting flux from the first volume towards the second volume is calculated;
      When FRONTIER is used together with FLUX, a scalar flux is calculated as the sum of fluxes in both directions of the frontier (provided that each requested frontier is not requested twice in a LIST, in one direction and then the opposite: in this case the user is invited to use ORIENTED_FRONTIER keyword instead);
    - POINT or TARGET_SURF if FLUX_PT has been chosen: next-event estimator toward one or several points or toward a single rectangular target surface;

- EXTENDED_MESH for extended functionalities of scores on meshes. See section 15.8 for details.
- MESH if the user requests a COLL tally on a regular mesh (defined independently of the geometry). This mesh of $N_x \times N_y \times N_z$ cells of sizes $d_x$, $d_y$, $d_z$ must be defined as follows:

**MESH** ***ENTROPY MESH_INFO***
**$N_x$ $N_y$ $N_z$**
**$d_x$ $d_y$ $d_z$**
**FRAME**
**$O_x$ $O_y$ $O_z$**

where $(O_x, O_y, O_z)$ is the center of the reference cell.
Important note: This option is unavailable for the current version. The user can use the extended mesh feature instead. See section 15.8 for details on the extended mesh.
The following example defines a mesh of $20 \times 20 \times 1$ cells each of sizes $10 * 10 * 300$, the reference cell being centered in $(5, 5, 150)$:

```
1 COLL
GRID DEC_ABSORPTION
MESH
20 20 1
10 10 300
FRAME 5 5 150
```

(subsection 15.9.2). The MESH_INFO option gives additional information about the mesh's specification displaying the volume and centers of all cells.

13. STORE_IN_FILE: when the score is required on a list of volumes using VOLUME and LIST keywords, the STORE_IN_FILE option stores the list of the requested volumes and associated results in a separate ASCII file.
The following example requests a score in volumes 5 and 7:

```
VOLUME
STORE_IN_FILE
LIST 2 5 7
```

14. <LIST_OR_SUM> is either LIST or SUM. This keyword is omitted in the cases of the mesh tally and the TARGET_SURF geometrical element type.

15. nb(elements) is the number of geometrical elements for which the results are requested. The result is given for each element on the list when the LIST option is selected, but it is the sum of the scores for the elements on the list when the SUM option is selected.
For choice VOLUME this number can be replaced by the keyword ALL. The results will then be given for each volume of the geometry.

The DEPOSITED_SPECTRUM and DEPOSITED_CHARGE scores can be used with LIST or SUM of nb(elements) volumes.

16. element(1)...element(n) is a list of elements, with an element(i) consisting of:

- a) a volume number for choice VOLUME.
  Examples:

```
2 TRACK
GRID GRID_ABSORPTION
VOLUME LIST 2 5 7
```

  Response 2 is requested with estimator TRACK for the grid indicated in two volumes: in volume 5 and in volume 7.

```
2 TRACK
GRID GRID_ABSORPTION
VOLUME SUM 2 5 7
```

  Here, the response is requested in volume 5+7 (consisting of two volumes).

```
2 TRACK
GRID GRID_ABSORPTION
VOLUME LIST ALL
```

  Here, the response is requested in all the volumes.

- b) a volume number, the depth and the three cell indices for each lattice (from the coarsest to the smallest) to the desired depth, for choice CELL. It is, therefore, possible to extract cells from cells in the case of a lattice of lattices. It should be noted that in version 4.1, it was the lattice number and not the depth that was requested. The presence of the additional keyword DEPTH ensures compatibility with older data sets.
  Examples:

```
2 COLL
GRID GRID_ABSORPTION
CELL LIST 2
5 50 1 2 3
4 40 5 2 3
```

  Response 2 is requested, with the COLL estimator for the grid indicated in two cells: in volume 5 (lattice 50), for cell (1,2,3) and in volume 4 (lattice 40), cell (5,2,3).

```
2 COLL
GRID GRID_ABSORPTION
CELL SUM 2
```

```
5 DEPTH 1 1 2 3
4 DEPTH 1 1 2 3
```

Here the response is requested in volume 5+4, only in cell (1,2,3).

```
2 COLL
GRID GRID_ABSORPTION
CELL LIST 1
5 DEPTH 2 1 2 3 1 1 1
```

Here the response is requested in volume 5, only in cell (1,1,1) of cell (1,2,3) of the largest lattice.

Notes: the sum of the results can be made in a cell and in a volume. In this case, for a volume sum, it is necessary to declare a depth of 0.

- c) two volume numbers, defining a surface, for the SURF option. It should be noted that the area outside the geometry is numbered -1 (a surface bounding the geometry therefore has a surface between a volume of a given number and -1). It must be borne in mind that the number of the volume is that of the volume to which the particle belongs, and not a construction volume. Consequently, the volume cannot be a lattice. This excludes the possibility of surface-based flux (or current) between a lattice and a volume.
  Example:

```
1 SURF
GRID GRID_FLUX
FRONTIER LIST 4
1 2
3 4
5 6
7 8
```

  Response number 1 is requested, with the SURF estimator for the grid indicated and for four surfaces: the surface between volumes 1 and 2, the surface between 3 and 4, the surface between 5 and 6, and the surface between 7 and 8. For calculation of flux or surface-based reaction rate (but not for a current), it is necessary to be able to calculate the normal unit vector.
  This is not currently possible in TRIPOLI-4® for a surface that results from the application of an operator (UNION, INTE, etc.). It is nonetheless sufficient to calculate the normal vector for one of the two volumes that define the surface.
  Note: each element is defined by two volume numbers. For the case of escape, the number of the volume and the number -1 are given. No verification is made and the surface that is defined may not exist.

- d) the coordinates $(X,Y,Z)$ of a point, for the FLUX_PT option. Warning: the position of this point is surrounded by an exclusion sphere that should not cut separating surfaces of different compositions volumes. By default, the radius of this exclusion sphere is set to 0.5 cm. The user can modify it

when using the keyword EXCLUSION followed by the radius, to be placed before the coordinates of the first point. In this case the choice of a suitable value is left to the user's responsability.
Examples:

```
1 FLUX_PT
GRID GRID_FLUX
POINT LIST 1 0 0 0
```

Response 1 is requested, with estimator FLUX_PT for the grid indicated at the point: (0,0,0). The radius of the exclusion sphere is 0.5 cm.

```
1 FLUX_PT
GRID GRID_FLUX
POINT LIST 1 EXCLUSION 0.2 0 0 0
```

In this case the radius of the exclusion sphere is 0.2 cm.

- e) data of the target surface with the FLUX_PT estimator and TARGET_SURF geometrical element.
  The syntax is as follows:

**TARGET_SURF**
**dimension_V dimension_W**
**POSITION Px Py Pz**
**ORIENTATION**
**Ux Uy Uz**
**Vx Vy Vz**
**Wx Wy Wz**

where:

  - dimension_V and dimension_W are the sizes or the rectangular surface;
  - Px Py Pz are the coordinates of the center of the target surface;
  - Ux Uy Uz , Vx Vy Vz , Wx Wy Wz are the coordinates of the orthogonal vectors. *U* must be orthogonal to the target surface.

## 15.8 EXTENDED MESHES

This provides extended functionalities of scores on meshes.
Both TRACK and COLLISION estimators can be used, and available frames are CARTESIAN, CYLINDER and SPHERE.

The syntax is as follows :

**EXTENDED_MESH**
*MESH_INFO*
*MERGE_ENERGIES*
*TRIANGLEZ*
*ENTROPY*
*NORMALIZE*
*STORE_IN_FILE*
* **WINDOW**
Uniform window data
* **WINDOW_VARIABLE**
Nonuniform window data
**FRAME**
<**FRAME_TYPE**>
$\mathbf{O'}_x$ $\mathbf{O'}_y$ $\mathbf{O'}_z$
$\mathbf{i}_x$ $\mathbf{i}_y$ $\mathbf{i}_z$
($\mathbf{j}_x$ $\mathbf{j}_y$ $\mathbf{j}_z$) if cartesian
$\mathbf{k}_x$ $\mathbf{k}_y$ $\mathbf{k}_z$

The MESH_INFO option gives additional information about the mesh's specification displaying the volume and centers of all cells.
The MERGE_ENERGIES option allows to store all the meshes for the different energy bins to be merged inside one unique file. The number of bins and the energy limits of the bins will be found in the grid and positions parameters of the .general header.
The TRIANGLEZ option allows the user to split each parallelepiped in two prisms in order to calculate scores on hexagonal lattices. It will thus double the number of cells in the $V$ direction. See figure 15.1 for numbering convention. Note that this option cannot be used on a nonuniform window.
The NORMALIZE option divides each output by the volume of each individual cell.
The use of the optional keyword ENTROPY is detailled below (subsection 15.9.2).
The STORE_IN_FILE option stores the result in a separate ASCII file. This file may be used by the T4G tool (see chapter 38). If MED keyword is specified after this option the result is stored in a separate MED format file including mean values and sigma values. The three usual coordinate systems are available with this feature: cartesian, cylindrical, and spherical.
As well as describing mesh for biasing, it can be useful to use uniform or nonuniform window depending on the problem. The window is defined after the keyword WINDOW for a uniform window, or WINDOW_VARIABLE for a nonuniform window.
The definition of a FRAME is explained in section 4.5.

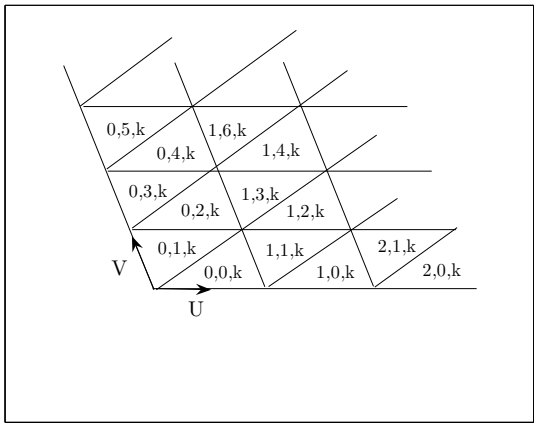



Figure 15.1: TRIANGLEZ numbering convention

### 15.8.1 UNIFORM WINDOW SYNTAX

The syntax for a window of $N_{u'} \times N_{v'} \times N_{w'}$ regular cells uses the same conventions as those used for the keyword CHECK_GEOM (see section 7.13.2), except for the additional arguments indicating the number of cells in each dimension:

**WINDOW**
**U**$_{min}$ **V**$_{min}$ **W**$_{min}$
**U**$_{max}$ **V**$_{max}$ **W**$_{max}$
**N**$_u$ **N**$_v$ **N**$_w$

### 15.8.2 NONUNIFORM WINDOW SYNTAX

The syntax for a variable window of $N_{u'} \times N_{v'} \times N_{w'}$ cells is:

**WINDOW_VARIABLE**
**N**$_u$ **N**$_v$ **N**$_w$
**u**$_0$ **N**$_1$ **u**$_1$ **N**$_2$ **u**$_2$ **... N**$_n$ **u**$_n$
**v**$_0$ **M**$_1$ **v**$_1$ **M**$_2$ **v**$_2$ **... M**$_m$ **v**$_m$
**w**$_0$ **P**$_1$ **w**$_1$ **P**$_2$ **w**$_2$ **... P**$_p$ **w**$_p$

where :

- u$_0$,...,u$_n$,v$_0$,...,v$_m$,w$_0$,...,w$_p$ are given in the mesh frame

- $N_i$ is the number of regular cells between coordinates $u_{i-1}$ and $u_i$
- $M_j$ is the number of regular cells between coordinates $v_{j-1}$ and $v_j$
- $P_k$ is the number of regular cells between coordinates $w_{k-1}$ and $w_k$
- moreover, these data must verify :
  $N_1 + N_2 + ... + N_n = N_u$
  $M_1 + M_2 + ... + M_m = N_v$
  $P_1 + P_2 + ... + P_p = N_w$

### 15.8.3 VARIANCE REDUCTION AND TRACK ESTIMATORS

Along with the TRACK estimator, the user wanting to use the standard variance reduction method of the code (based on the Exponential Transform) must place the statement "MAX_SPLITTING_PARTICLE_NUMBER 1" in the SIMULATION block (see Section 22.2). This option disables the quota sampling inside the simulation. The most important tool for variance reduction (the Exponential Transform) is still in function when using this option.

### 15.8.4 NOTE ON COLLISION ESTIMATOR

Using the COLLISION estimator with cells astride on compositions with both very low and high optical density is statistically ill conditioned: it may yield false results without warning (that is, with an underestimated standard deviation).
Other remarks:

- $U_{min}$ can be non-zero in any frame. For example, in CYLINDER mode, it allows to have scores on cylinders not including the origin.
- basis vectors need not to be orthogonal in the cartesian frame, thus enabling scores on one third of an hexagonal domain. You can use three meshes to have scores on a full hexagonal domain (figure 15.2). Using the TRIANGLEZ mode allows to get scores on all of the 6 triangles composing any hexagonal cell. Some post-processing by the user is necessary to get the value of scores on each hexagonal cell.

### 15.8.5 NOTE ON MESH EDITION RESULTS

The simulation keywords MESH_EDITION and SUPPRESS_MESH_EDITION_TEXT can be used to reduce the size of the results file when requesting results in meshes (see chapter 22).

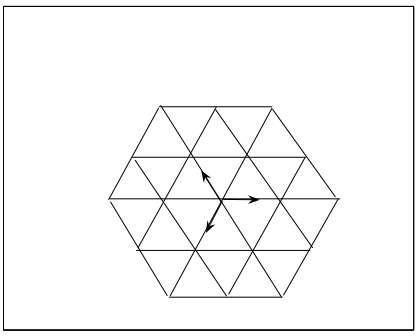


Figure 15.2: Hexagonal domain decomposition

## 15.9 EXAMPLES OF SCORE REQUESTS

### 15.9.1 EXAMPLE ON A SIMPLE GEOMETRY

In the case of a geometry consisting of a box with 10 cm sides along $x$ and 20 cm sides along $y$ and $z$, centered at the origin:

```
GEOMETRY
TITLE test
TYPE 1 BOX 10 20 20
VOLU 1 COMBI 1 0 0 0 ENDV
ENDGEOM
```

Assuming that it is empty:

```
COMPOSITION 1
POINT_WISE 300 VIDE 1
H1 1.E-20
END_COMPOSITION

GEOMCOMP
VIDE 1 1
END_GEOMCOMP
```

For a mono-kinetic source to 1 MeV centered on the origin (5 cm from the sides along $x$) and normalized to three particles, the particles being directed with an angle of 45 degrees with respect to the normal of one side:

```
SOURCES_LIST 1
NORM 3
SOURCE
NEUTRON POINT 0 0 0
ANGULAR_DISTRIBUTION MONO_DIR 1 0 1
ENERGETIC_DISTRIBUTION  SPECTRUM MONOKINETIC 1
TIME_DISTRIBUTION DIRAC 0
END_SOURCES
END_SOURCES_LIST
```

To simulate 50 batches of 100 particles each (see the command syntax, below in the manual, monitoring is disabled (it is pointless because no variance reduction technique is used), in order not to have 500 particles in the first batch):

```
SIMULATION
BATCH 50
SIZE 100
```

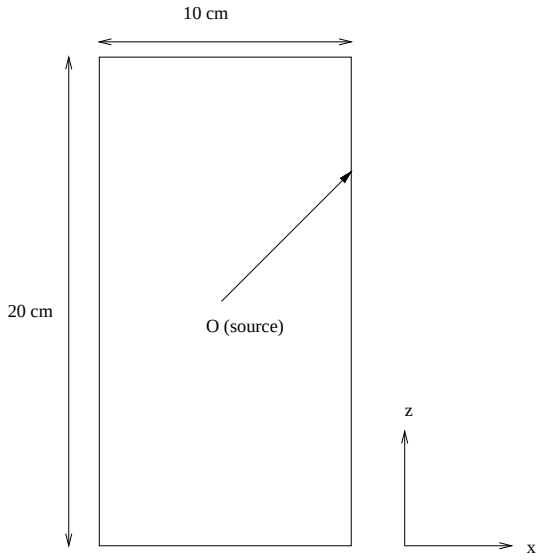



Figure 15.3: Example for score requests

```
PARTICLE 1 NEUTRON
MONITORING 0
END_SIMULATION
```

All the particles will therefore travel $5 \times \sqrt{2} \simeq 7.071$ cm in the box (volume 1) then escape without collision out of the geometry (volume number -1 by definition) through the surface (defined as the boundary between volumes 1 and -1), from 1 to -1.
Each source particle has an initial weight of 1. As they all escape without a change in their weight (no fractioning or roulette), the average weight of escape is 100 per batch.
For a one-group grid covering the domain (with a simulation default of 20 to $10^{-11}$ MeV):

```
GRID_LIST
1
GRID_INTEGRAL
2
20.  1.E-11
END_GRID_LIST
```

We seek the flux and a virtual microscopic reaction (fission) rate for uranium-235 (at 300 K):

```
RESPONSES
4
FLUX NEUTRON
CURRENT NEUTRON
REACTION NEUTRON
```

```
NUCLEUS U235 FICTIVE 300 INTERACTION 1 33
DOSE_H*(10) NEUTRON
END_RESPONSES
```

Let us request the volume-based flux (response number 1) using the volume-based track length and collision estimators in volume 1 and by the surface-based estimator between 1 and -1. Let us also request the current (response number 2) between 1 and -1. (Note that the code forbids requesting a current from -1 to 1, corresponding to an inward current.)
Finally, we request the reaction rate (response number 3) in volume 1 by collision and track length volume-based estimators:

```
SCORE
7
1 TRACK GRID GRID_INTEGRAL
VOLUME LIST 1 1
1 COLL GRID GRID_INTEGRAL
VOLUME LIST 1 1

1 SURF GRID GRID_INTEGRAL
FRONTIER LIST 1 1 -1

2 SURF GRID GRID_INTEGRAL
FRONTIER LIST 1 1 -1

3 TRACK GRID GRID_INTEGRAL
VOLUME LIST 1 1
3 COLL GRID GRID_INTEGRAL
VOLUME LIST 1 1

4 TRACK GRID GRID_INTEGRAL
VOLUME LIST 1 1
END_SCORE
```

The results are:

- The flux and reaction rates requested using the collision estimator give zero values. This is because there is no collision in the empty space and the estimator can therefore not operate.
- The track length estimator uses the fraction of the trajectories that pass through volume 1 for the energies in the grid interval. For the unique interval of this example, these fractions are $5 \times \sqrt{2} \simeq 7.071$ cm for each neutron. We must, therefore, have a flux (integrated in volume) of 7.071 neutrons·centimeters, given a source normalized to 1 neutron (the weight being 1 each time). Since the source is actually normalized to 3, the following volume integrated flux is obtained: $3 \times 7.071 \simeq 21.2$ neutrons·centimeters.

- The surface-based flux depends on the order in which the volumes are given for determining the area. The flux is calculated by dividing the weight (1 in this case) of the particle passing through by the cosine of the angle (45 degrees here) with the normal to the surface (a cosine of $1/\sqrt{2}$ here). There should, therefore, be a flux (integrated over a surface) of $\sqrt{2} \simeq 1.41$ neutrons for a source normalized to 1 neutron. As the source is actually normalized at 3; hence, the following flux is obtained (integrated over a surface) of $3 \times \sqrt{2} \simeq 4.24$ neutrons·centimeters.
- The current depends on the order in which the volumes are given to define the surface. It is zero from -1 to 1 since no particles come from outside the geometry. It is calculated by taking the weight (here 1) of the particle passing through the surface. There should, therefore, be a current (integrated over a surface) of 1 neutron for a source normalized to 1 neutron. As the source is in fact normalized to 3, a neutron current of 3 is obtained.
- Reaction rates are calculated by weighting the flux by a function. Here, the weighting is performed using the microscopic fission cross-section of uranium-235 (1.196 barns, i.e., $1.196 \times 10^{-24}$ cm$^2$ at 1 MeV with ENDF B-VI, see MF=3,MT=18 in the evaluation file). What is obtained is, therefore, a microscopic fission rate (integrated in volume) of $21.2 \times 1.196 \times 10^{-24} = 25.39 \times 10^{-24}$ particles created by fission in the volume. Usually, the particles do not have the same energy, and appropriate weighting is required.
- The dose equivalent rate is calculated by weighting the flux by a function. Here the function is 1.51 ($\mu$Sv/h)(n/cm$^2$-s) at 1 MeV. What is, therefore, obtained is a dose equivalent rate (integrated in volume) of $21.2 \times 1.51 = 32.01$ $\mu$Sv/h. Usually, the particles do not have the same energy, and appropriate weighting is required.

Note: if surface values had been given (100 cm$^2$) and volumes (1000 cm$^3$) by:

```
VOLSURF
SURFACE 1
1 -1 100
VOLUME 1
1 1000
END_VOLSURF
```

it would have been necessary to divide the obtained values by the volume or the surface data. Thus, the volume-based flux would have been 0.0212 neutrons per cm$^2$. The surface-based flux would have been 0.0424 neutrons per cm$^2$ and the current 0.03 neutrons per cm$^2$. The volume-based fission rate would have been $0.02539 \times 10^{-24}$ particles created by fission per unit of volume (for a 3-neutron source). The dose equivalent rate would then have been $3.2 \times 10^{-2}$ $\mu$Sv/(h $\cdot$ cm$^2$).

**Defining scores for the criticality calculation example of chapter 5**

```
SCORE 5
1 COLL
GRID GRID_FLUX
```

```
VOLUME LIST 2 4 7
1 SURF
GRID GRID_FLUX
FRONTIER LIST 1 1 2
2 TRACK
GRID GRID_ABSORPTION
VOLUME LIST 2 5 8
3 TRACK
GRID GRID_ABSORPTION
VOLUME LIST 3 1 4 7
4 TRACK
GRID GRID_ABSORPTION
VOLUME LIST 8 1 2 3 4 5 6 7 8
END_SCORE
```

Important: in the case of surface-based definition of volumes, the boundary plane between volumes 1 and 2 is, in fact, the boundary between volumes 12 (virtual and joined at 1, see section on geometry) and 2. Consequently, the code cannot calculate the normal vector for volume 1 on this boundary. But, it can calculate the normal vector to volume 2, TRIPOLI-4® can indeed compute the normal vector at the 1-2 interface.

### 15.9.2 USING THE KEYWORD ENTROPY

Using the keyword ENTROPY in the MESH command or in the EXTENDED_MESH command is optional. When using it, two values are printed in the output file: the Shannon and the Boltzmann entropies[28]. This values are not integrated over every batches but are those calculated at the current batch. Thus, if the user wants to proceed a detailed study of the evolution of entropy, he has to request a batch by batch edition. Moreover, the computation of entropy of sources has a physical meaning in criticality only if it is based on a fission rate (code 33, cf 13.4).

Warning: the Shannon entropy is not computed in parallel mode. Alternatively, the user can temporarily run in monoprocessor mode since the behaviour of the source convergence is similar in both modes: when studying the entropy in monoprocessor mode, it is possible to find out the number of batches necessary to reach the source convergence and discard then as many batches in the parallel run.

Example:

```
RESPONSES 1
REACTION NEUTRON
NUCLEUS U235 COMPOSITION FUEL
INTERACTION 1 33
END_RESPONSES
```

```
SCORE 1 COLL
GRID DEC_INTEGRAL
MESH  ENTROPY 20 20 1 10 10 300  FRAME 5 5 150 END_SCORE
```

This command block activates the computation of fission rates in each cell of the $20 \times 20 \times 1$ mesh of cell sizes $10 * 10 * 300$ with the reference cell being centered in $(5, 5, 150)$. The energy grid used is `DEC_INTEGRAL`. The entropy computation is usually based on one group.
Theses fission rates are computed only on the U235 nucleus of the `FUEL` composition.
Then, the code computes the entropy of the source distribution (i.e. $\nu$-fission rates) on the requested mesh. The result is thus a scalar value, its batch by batch evolution is used to guess the fission source stability. The code uses two different expressions of entropy: - the Shannon entropy - and a first order correction of the latter: the Boltzmann entropy. Important variations of entropy can reveal source convergence problems in criticality studies.

# Chapter 16

# DEFINING SCORES DECOMPOSED OVER REAL SPHERICAL HARMONICS

For some applications, it may be of interest to compute reaction rates decomposed over real spherical harmonics. When dealing with real spherical harmonics, there is many different conventions leading to potential confusion [76]. In order to be consistent with the conventions used in the APOLLO deterministic code, we chose the spherical harmonics convention named Schmidt semi-normalized in [76] with the z-axis coordinate considered as cosine θ.

The neutron flux can be expanded over real spherical harmonics basis as :

$$\psi^g(\Omega) = \int_{\Delta E_g} dE\psi(\Omega,E) \approx \sum_{\ell=1}^{\ell_{max}} \sum_{|m|<\ell} A_{\ell,m}(\Omega)\phi^g_{\ell,m} \tag{16.1}$$

$$\text{where } \phi^g_{\ell,m} = \frac{1}{4\pi}\int_\Omega d\Omega A_{\ell,m}(\Omega)\int_{\Delta E_g} dE\psi(\Omega,E) \tag{16.2}$$

where $A$ is the real spherical harmonics basis defined as

$$A_{\ell,m}(\Omega) = \begin{cases} \bar{P}_{\ell,m}(\cos\theta)\cos(m\phi) \text{ if } m \geq 0 \\ \bar{P}_{\ell,|m|}(\cos\theta)\sin(|m|\phi) \text{ if } m < 0 \end{cases} \tag{16.3}$$

$$\text{with } \bar{P}_{\ell,m}(cos\theta) = (-1)^m\sqrt{(2-\delta_{0m})\frac{(\ell-m)!}{(\ell+m)!}}P_{\ell,m}(\cos\theta) \tag{16.4}$$

With $\delta_{0m}$ being the Kronecker delta and $P_{\ell,m}$ the associated Legendre Polynomials derived from the following relations :

$$P_{\ell,m}(x) = (-1)^m(1-x^2)^{m/2}\frac{d^mP_\ell(x)}{dx^m} \tag{16.5}$$

where $P_\ell(x)$ is the Legendre Polynomial.
In an exact similar way to the flux in equation 16.2, reaction rate for a reaction $r$ and scattering matrix for reaction $s$ can decomposed over real spherical harmonics as :

$$\tau^g_{\ell,m}(r) = \frac{1}{4\pi}\int_\Omega d\Omega A_{\ell,m}(\Omega)\int_{\Delta E_g} dE\Sigma_r(E)\psi(\Omega,E) \tag{16.6}$$

$$\tau^{g'\to g}_{\ell,m}(s) = \frac{1}{4\pi}\int_{\Delta E'_g} dE'\int_{\Delta E_g} dE\int_\Omega d\Omega A_{\ell,m}(\Omega)\int_{\Omega'} d\Omega'\Sigma_s(\Omega'\cdot\Omega, E'\to E)\psi(\Omega',E') \tag{16.7}$$

When scores decomposed over real spherical harmonics are required, TRIPOLI-4 computes the quantities defined in 16.2, 16.6, 16.7 for the angular flux, the scattering matrix and the total, fission, scattering, absorption, and nu fission cross sections.
These reaction rates decomposed over real spherical harmonics calculations are defined between the keywords REACTION_RATES_SPHERICAL_HARMONICS and END_REACTION_RATES_SPHERICAL_HARMONICS. The syntax is as follows:

**REACTION_RATES_SPHERICAL_HARMONICS**
Description of an extended mesh
ENERGY GRID <energy_binning_name>
MOMENTS <number_of_moments>
**END_REACTION_RATES_SPHERICAL_HARMONICS**

The extended mesh definition is the same as defined in the section 15.8. However, there is few limitations for the moment: the NORMALIZE, ENTROPY and STORE_IN_FILE options cannot be used in the frame of the REACTION_RATES_SPHERICAL_HARMONICS block.

The <energy_binning_name> is the name of an energy grid defined in the GRID_LIST command. The <number_of_moments> is an integer corresponding to the maximum $\ell$-th order. This value must be positive.

Below is an example of requiring these reaction rates :

```
REACTION_RATES_SPHERICAL_HARMONICS
   EXTENDED_MESH
      WINDOW
       -0.7470 -0.7470 -5
        0.7470  0.7470  5
         2 2 1
      FRAME CARTESIAN
        0. 0. 0.
        1 0 0
        0 1 0
        0 0 1
```

```
   ENERGY GRID DEC_2G
   MOMENTS 5
END_REACTION_RATES_SPHERICAL_HARMONICS
```

# Chapter 17

# VARIANCE REDUCTION TECHNIQUES (BIASING)

Variance reduction techniques are defined between the keywords `VARIANCE_REDUCTION` and `END_VARIANCE_REDUCTION`. This command must be called once for a purely neutron (or purely gamma) problem with the keyword `NEUTRON` (or `PHOTON`). The command must be called twice for a combined neutron and gamma problem: once using the keyword `NEUTRON` and once with the keyword `PHOTON`. For other types of particles, it is not possible (with this version of the code) to define a variance reduction technique.

Variance reduction is defined by inputting:

- a particle type;
- a biasing energy grid;
- one or more targets determining the zone of maximum importance;
- a biasing mesh containing the geometry of the problem;
- a biasing type (user or automatic).

Visual output can be requested.

Notes:

- Variance reduction techniques are not implemented for homogenized multigroup cross-sections.
- Adaptive Multilevel Splitting (`AMS`) and weight windows can also be used as variance reduction technique and are detailed in Chapters 18 and 19.
- The CADIS variance reduction scheme is available through a coupling with the IDT deterministic solver, and is detailed in Chapter 20.

The syntax is as follows:

```
                VARIANCE_REDUCTION
                        NAME <biasing_name>
                        <PARTICLE_TYPE>
                        GRID
                                Grid data
*1                      DETECTOR
                                Discrete detector data
*1                      DETECT_RHO
                                Detection surface data (cylindrical)
*2                      MESH
                                Uniform mesh data
                        END_MESH
*2                      MESH_VARIABLE
                                Nonuniform mesh data
                        END_MESH_VARIABLE
                        INTERPOLATION flag
                        INIPOND
                                Biasing type data
                        END_INIPOND
                        ALFA
                                Energy biasing data
                        TIME
                                Time biasing data
#                       STORE_IN_FILE <format>
#                       USE_FILE <format> filename
                        GRAPH
                                Display data
                END_VARIANCE_REDUCTION
```

- The name of the biasing is optional. In the case of a coupled radiation shielding-criticality simulation (see chapter 22), this data is mandatory in order to specify in which type of simulation the biasing must be used.
- The possible values for <PARTICLE_TYPE> are NEUTRON or PHOTON.
- Grid data, detector data, mesh data, inipond data and optional ALFA, TIME, and GRAPH data are detailed in the following sections of this chapter.
- The optional keyword INTERPOLATION enables to deactivate (if flag equals 0) or to force (if flag equals 1) the spatial interpolation on the importance map (see section 17.2), for each biasing group (by default, the code tries, for each group, to verify a criterion on the variation of the importance of more than 20 between the centers of neighbor cells, and activates or not the interpolation on the whole mesh, for the current biasing group).

- STORE_IN_FILE allows for the spatial importance map to be stored in a file. Its format is given by the keyword <format> which can be either ASCII, ASCIIG or XDR.
  - In the ASCII case, an ASCII file filename.importanceMap.bcd is dumped, where filename is the name of the input deck data file. It contains keywords readable by the code, user's data on the biasing energy grid and mesh, and values produced by the code : the spatial importance values by group and cell, the k factors (see section 17.2) by material and group and the adjusting factors by group. This file can be used in subsequent TRIPOLI-4® simulations, e.g., in parallel mode (see chapter 33) in order to impose the same importance map to all simulators.
  - In the ASCIIG case, a file filename.importanceMap-group<igroup>.general is dumped for each group (as also obtained with the keyword STORE_IMPOR_IN_FILE of version 7 of the code), with a general ASCII format usable by tools such as OpenDX, where filename is the name of the input deck data file.
  - In the XDR case (XDR meaning processor independent binary format), the file is named filename.importances<pid> for neutrons and filename.importances_g<pid> for photons, where filename is the name of the input deck data file and the number <pid> is the process id that is found at the beginning of the results file. This case is equivalent to the keyword STORE_IMPOR_IN_FILE of TRIPOLI-4® versions up to 7.
- USE_FILE allows the user to load a file previously generated either by TRIPOLI-4® (with the STORE_IN_FILE option) or by another code and related to the same geometry and materials. The format of the file must be specified by the keyword <format> which can be either ASCII or XDR. The name of the file is filename.
  - The ASCII case should be used together with the keyword MONITORING 0 in the SIMULATION command block (see chapter 22) in order to avoid the calculation of new adjusting factors by TRIPOLI-4®. The structure of the file is that produced by the STORE_IN_FILE keyword, however, the k factors are optional and can be recomputed by the code on the given biasing mesh when missing.
  - The XDR case is equivalent to the keyword USE_IMPOR_FILE of TRIPOLI-4® versions up to 7. Geometry, materials, and biasing must be left unchanged between the storage and use modes. This old option enables only to save the cost of initializing biasing data in memory.

## 17.1 THE CONCEPT OF A WEIGHT

For radiation shielding problems with strong attenuation, it is possible to bias the transport of the particles emitted in order to mainly simulate trajectories whose contribution to the expected result is the greatest [9], [13], [14], [78], [79]. Each particle is assigned a **weight**, the inverse of the weight being **importance**.
Weight $\Pi$ of the particle is factorized as follows:

$$\Pi(\mathbf{r},\Omega,E,t) = \Pi_1(\mathbf{r}) \cdot \Pi_2(\Omega) \cdot \Pi_3(E) \cdot \Pi_4(t)$$

where:

- the spatial dependence $\Pi_1(\mathbf{r})$ is calculated by INIPOND, either automatically or with data supplied by the user for a selected mesh. In reality, an energy dependence is still present in the spatial biasing (see below) and $\Pi_1(\mathbf{r}) = \Pi_1(\mathbf{r}, E_g)$.
- the dependence on direction $\Pi_2(\Omega)$ is calculated by the code;
- the energy dependence $\Pi_3(E)$ is of the form $\frac{1}{E^\alpha}$;
- the time dependence $\Pi_4(t)$ is of the form $\frac{1}{(1+\lambda t)}$.

## 17.2 SPATIAL BIASING

The values of $\Pi_1(\mathbf{r})$ are calculated to establish importance maps. An importance map consists of the importance values (inverse of the particle weight) given at the center of each cell of a cylindrical or parallelepipedic **mesh**. Importances are calculated using **bias parameter** values $k$ (homogeneous to a cross-section) for each composition and either point **detector** positions or a cylindrical detector surface. The code calculates the importance maps for a number of energy ranges $E_g$. For each energy range or **biasing energy group** there is a corresponding value of $k$, hence a map.

### 17.2.1 BIASING ENERGY GROUPS

The user must indicate the grid of **biasing energy groups**: each group is associated with an importance map. There is, therefore, an energy dependence of the spatial biasing. It is necessary for the biasing grid to contain the simulation energy domain. The number of groups for neutrons is typically 3 to 5 in order to obtain satisfactory results, e.g.,

- one group for $E > 2$ MeV;
- one group for 5 eV $< E <$ 2 MeV;
- one group for $E <$ 5 eV.

With INIPOND, there must be at least two biasing groups.
The lowest bound of the given groups must be less than $10^{-10}$MeV.
The user enters the data as follows:

```
GRID
        nb(bounds)
                energy(1)...energy(n)
```

### 17.2.2 DETECTORS

At the present time, two types of biasing are available: either the results are optimized at discrete points of the geometry (detectors, specified in the DETECT command) or a specific analytical form is imposed for the equi-

importances (available forms are plane, cylindrical, and spherical), then complementary data for detection surfaces are input using the DETECT_RHO command (for the cylindrical and spherical cases) or the DETECT command (for the planar case).

- **The discrete detector case:**
  This corresponds to the type of biasing already used in code versions 3 and earlier. For each detector there is a corresponding coefficient β that specifies, when using **automatic biasing** with INIPOND, the degree of biasing (i.e., the value of bias parameter $k$). The coefficient is between 0 and 1. The higher the value of β is, the stronger the attraction of the particles towards that point is. The value of $k$ is calculated using the value of $\beta_1$ for the first detector. A value of $\beta_1 = 0.5$ for the first detector is sufficient for typical cases. The other values of β can give the value close to the detector (in reality, for each group $g$, the value of $I_g$ that plays a role in the importance formula). Similarly, **when user biasing** is utilized in INIPOND, the coefficient plays a role for the values close to the detector (factor $I_g$).
  The values of $k$ in the biasing groups corresponding to the user choice, as well as other values for different choices, are indicated in the TRIPOLI-4® output file.
  The syntax is as follows:

**DETECTOR**
**nb(detectors)**
$\mathbf{D}_{1x}$ $\mathbf{D}_{1y}$ $\mathbf{D}_{1z}$
$\beta_1$
...
$\mathbf{D}_{nx}$ $\mathbf{D}_{ny}$ $\mathbf{D}_{nz}$
$\beta_n$

  where (Dix, Diy, Diz) are the coordinates (in the reference frame of the geometry) of the $i$th detector and $\beta_i$ is the corresponding coefficient.

  These points are used to create the importance maps, and have no connection with physical detectors (for measuring flux for instance). They are used to attract particles to a specific point.

- **The splitting sphere case:**

  It may be helpful, in neutronic calculations, to increase the number of neutrons about an instrumentation point/detector, for instance.

  The syntax is as follows:

**CENTRAL_SPLITTING**
*weight_min* *radius*

  where **weight_min** is the minimum weight authorized for a neutron and **radius** is the radius of the (fictitious) splitting sphere about each of the detector points given in the DETECT command.

- **Case of a detection surface of an imposed analytical form:**
  This applies to **plane**, **cylindrical**, or **spherical** forms.
  For the cylindrical and spherical cases, the user indicates the cylindrical or spherical detection surface radius and the coefficient β, the degree of biasing, which is used (as in the case of discrete detectors) to calculate the values of $k$ for the different groups.
  The syntax is as follows:

```
DETECT_RHO
    1
            ρ(D)
            β_D
```

  where data item 1 must always be present: number of detection surfaces, where $\rho(D)$ designates the radius of the detection surface and $\beta_D$ designates the associated coefficient related to the degree of biasing.
  For the plane case, the syntax of the `DETECT` command is the same as in the case of discrete detectors, but, in this case, `nb(detectors)` is 1 and `(D1x, D1y, D1z)` are the coordinates of a point on the detection plane.
  For these three analytical cases, the other data concerning the surface (for example, in the cylindrical case: the axis of the cylinder for positioning it in the geometry) are included in the `INIPOND` command.

### 17.2.3 CALCULATION OF SPATIAL IMPORTANCE

#### 17.2.3.1 For biasing with discrete detectors

The spatial importance at point **r** for an energy group $g$ ($[E_g, E_{g+1}]$) is given by:

$$I_1(\mathbf{r}) = \frac{1}{\Pi_1(\mathbf{r})} = I_g e^{-\min_{(P_i)} \Sigma_{j=1}^{n_i} k[C(i,j)] \cdot d(i,j)}$$

where:

- $P_i$ is one of the possible trajectories between the current point **r** and the detector point $\mathbf{r}_0$ determining the zone of maximum importance;
- $n_i$ is the number of compositions passed through by trajectory $P_i$;
- $C(i,j)$ is the $j$th composition passed through by trajectory $P_i$;
- $d(i,j)$ is the distance travelled in composition $C(i,j)$ by trajectory $P_i$.

Close to the detector $I_1(r) = I_g$. The value is calculated as follows:

$$I_g = \frac{1}{\beta+1} \frac{E_{g+1}^{\beta+1} - E_g^{\beta+1}}{E_{g+1} - E_g}.$$

Thus, for a single composition, thus a single possible $k$, the importance at any point is proportional to $\exp(-kd)$, where $d$ is the distance between the point and the detector (minimum trajectory).

#### 17.2.3.2 For an analytical biasing (plane, cylindrical, or spherical)

The spatial importance at point **r** for energy group $g$ ($[E_g, E_{g+1}]$) is given by:

$$I_1(\mathbf{r}) = \frac{1}{\Pi_1(\mathbf{r})} = I_g e^{-\sum_{j=1}^{n} k[C(j)] \cdot d(j)}$$

with notations analogous to those used in usual discrete detector biasing, except that only one trajectory is possible (hence no index $i$). The trajectory is:

- orthogonal to the plane for a plane detection surface,
- radial with respect to the axis of the cylinder of biasing for a cylindrical detection surface,
- radial with respect to the center of the sphere of biasing for a spherical detection surface.

The trajectory goes from point **r** to the detection surface. As such, for a single composition, thus one $k$ possible, the importance at any point is proportional to $\exp(-kd)$, where $d$ is the distance between the point and the detection surface. This trajectory is along the normal unit vector of the plane (in plane biasing), the radius vector with respect to the axis of the cylinder in cylindrical coordinates, and the radius vector relative to the center of the detection sphere in spherical coordinates.

#### 17.2.3.3 For biasing with a splitting sphere

The spatial importance at point **r** is given by:

$$I_1(\mathbf{r}) = \frac{1}{\Pi_1(\mathbf{r})} = I_g e^{-\sum_{j=1}^{n} min(\frac{1}{weight_min}, \frac{radius^2}{d_j(r)^2})}$$

where:

- j is the index of the detector,
- n is the number of detectors,
- $d_j(r)$ is the distance between point r and detector j,
- radius is the radius of the splitting sphere.

## 17.2.4 MESH AND IMPORTANCE MAPS IN TRIPOLI-4®

A mesh is defined between the keywords `MESH` and `END_MESH` for a uniform mesh, or `MESH_VARIABLE` and `END_MESH_VARIABLE` for a nonuniform mesh.
For each case, two options are possible: a Cartesian mesh and a cylindrical mesh.

### 17.2.4.1 Cartesian mesh

The $N_{x'} \times N_{y'} \times N_{z'}$ parallelepipedic cells are directed along the axes of an **orthonormal** reference frame $(O'x', O'y', O'z') = (O', \mathbf{i}', \mathbf{j}', \mathbf{k}')$ that is distinct from that of the geometry and larger than it. The dimensions of each cell are $d_{x'}$, $d_{y'}$ and $d_{z'}$ along the three directions of the reference frame.
The origin point corresponds to the minimum coordinate point (*Xmin*, *Ymin*, *Zmin*) of the region in which the mesh extends, and the axes give the directions of the sides of the cells.

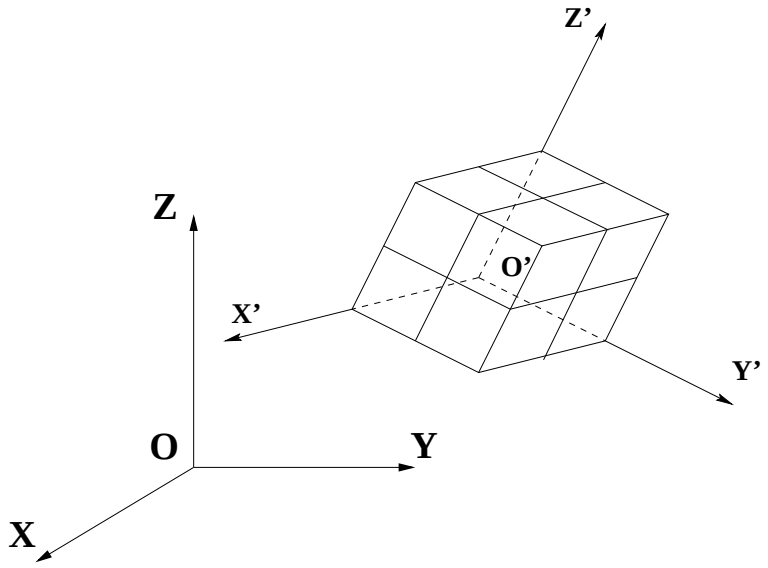



Figure 17.1: Example of a uniform Cartesian mesh of $2 \times 2 \times 2$ cells in a $(O'x', O'y', O'z')$ reference frame

### 17.2.4.2 Cylindrical mesh

The reference frame associated with the cylindrical mesh makes it possible to access the cylindrical coordinates $(\rho, \theta, z)$ of the points (and the centers of the cells). The **orthonormal** reference frame associated with the cylindrical mesh is of the same type as that used to define distributed sources: the coordinates of a vector $\mathbf{O'M}$ are:

$$\mathbf{O'M} = \rho \cos\theta \mathbf{i}' + \rho \sin\theta \mathbf{j}' + z\mathbf{k}'.$$

The user gives the origin $O'$ and the vectors $\mathbf{i}'$ and $\mathbf{k}'$, the second vector $\mathbf{j}'$ of the orthonormal reference frame being calculated automatically by vector product. The angles are defined counter-clockwise with respect to axis $(O',\mathbf{i}')$. The size of the $N_{\rho'} \times N_{\theta'} \times N_{z'}$ cells are: $d_{\rho'}$, $d_{\theta'}$, and $d_{z'}$ along the three coordinates, respectively (the angles are in radians).

**In both the Cartesian and cylindrical cases**, importances are calculated at the centers of the cells thus constructed. However, the center does not mean the center of gravity of the cell but the geometrical center, or the midpoint in each discretization, (in $\rho$, in $\theta$, and in $z$, for the cylindrical case).
A neighborhood degree (`neighborhood`) can be specified, a value of 3 must always be entered.
Important:

- The mesh must contain the geometry, and can, therefore, extend beyond it. However, the centers of the cells containing a discrete detector must belong to the geometry. In addition, the planes that define the cells must differ from the planes bounding the volumes, and the centers of the cells must not be on the boundaries of the geometry.

- Regarding the cylindrical detection surface, it must be included in the geometry over a height (coordinate $z$) that is significant (greater than $10^{-3}$ centimeters). If it extends beyond the geometry, the optical distances calculated between certain cell centers and the part of the detection surface outside the geometry will be set to infinity (zero importance at the centers of these cells). It should be noted that the code can stop (and display an error message) if it samples these points during the source normalization calculation or if, during the simulation, it evaluates the inverse of the importance at these points.

- Finally, again in the case of a cylindrical mesh, the mesh can extend beyond the geometry (centers of cells outside the geometry, for which the importance will be zero), provided that the geometrical boundaries are included in cells whose centers belong to the geometry.

Note:
The user can combine a Cartesian mesh with usual discrete detector biasing or with analytical biasing. Similarly, it is possible to combine the cylindrical mesh with any of the available biasings. The following cases are of particular interest: cylindrical biasing with a cylindrical mesh (making it possible to obtain cylindrical equi-importances highly smoothed over the mesh) and biasing with discrete detectors and a cylindrical mesh (cylindrical geometry but the points for which the results are to be optimized remain discrete). The case with a single cell in the $\theta$ variable can also be useful in attracting particles uniformly to rings, in each of which the user has placed a discrete detector.
In the other cases of analytical biasing, best results will be obtained using the most suitable mesh. Cartesian mesh will be well adapted to plane biasing, for example.

#### 17.2.4.3 Refining the mesh

In the case of a uniform mesh, it is possible to refine the mesh in certain particularly sensitive areas using the `DIVISION WINDOW` command.
The number of cells is then multiplied by 8 (a factor of 2 for each of the 3 coordinates) in the region involved,

limited by two points whose coordinates are given in the new reference frame: $(X'min, Y'min, Z'min)$ and $(X'max, Y'max, Z'max)$ if Cartesian, $(\rho_1, \theta_1, z_1)$ and $(\rho_2, \theta_2, z_2)$ if cylindrical, with the indices 1 (or 2) corresponding to the minimum (or maximum) coordinates for coordinates $\rho$ and $z$; however, for angular definition, $\theta_1$ and $\theta_2$ are given counter-clockwise (it may be that $\theta_1 > \theta_2$, if the initial mesh extends fully in $[0, 2\pi]$,).

#### 17.2.4.4 Uniform mesh syntax

The syntax for $N_{x'} \times N_{y'} \times N_{z'}$ regular cells of dimension $d_{x'}$, $d_{y'}$, $d_{z'}$ is :

```
MESH
        N_x' N_y' N_z'
        d_x' d_y' d_z'
        neighborhood
         FRAME
                <FRAME_TYPE>
                O'_x O'_y O'_z
                i'_x i'_y i'_z
                (j'_x j'_y j'_z)
                k'_x k'_y k'_z
        DIVISION   WINDOW
                X'_min Y'_min Z'_min
                X'_max Y'_max Z'_max
END_MESH
```

where $(x', y', z')$ are either Cartesian coordinates in the case of a Cartesian mesh, or cylindrical coordinates of the $(\rho, \theta, z)$ type in the case of a cylindrical mesh.
The definition of a frame is explained in section 4.5. Important From version 3, the reference frame type `<FRAME_TYPE>` **must be** specified. It may be either `CARTESIAN` or `CYLINDER`. Data files made for earlier versions than 3 must be modified accordingly.
From version 8 on, the cell size can be defined inferior to 0.1 cm.
It must also be borne in mind (note previous remark) that, in the block `DIVISION WINDOW`, $\theta_{min} < \theta_{max}$ is not necessarily true if the reference frame is cylindrical.
At the present time, `neighborhood` must equal 3.

#### 17.2.4.5 Nonuniform mesh syntax

For certain problems it can be useful to define fine cells in certain areas of the geometry (around the detector, on the contributing paths between the source and the detector ...). A uniform mesh even with a local refinement may not be sufficient then, or could lead to simulation time problems because of the too large number of cells. This can happen particularly :

- for large geometries, when tallying in a precise zone is requested,
- in configurations where there is locally a strong attenuation in certain compositions.

It can thus be judicious to use a nonuniform mesh.
The syntax for a nonuniform mesh of $N_{x'} \times N_{y'} \times N_{z'}$ cells is :

**MESH_VARIABLE**
**$N_{x'}$ $N_{y'}$ $N_{z'}$**
**0 $N_1$ x'$_1$ $N_2$ x'$_2$ ... $N_n$ x'$_n$**
**0 $M_1$ y'$_1$ $M_2$ y'$_2$ ... $M_m$ y'$_m$**
**0 $P_1$ z'$_1$ $P_2$ z'$_2$ ... $P_p$ z'$_p$**
**neighborhood**
**FRAME**
Mesh data
**END_MESH_VARIABLE**

where :

- $x'_1,...,x'_n,y'_1,...,y'_m,z'_1,...,z'_p$ are given in the mesh frame (Cartesian or cylindrical);
- $N_1$ is the number of regular cells between coordinates 0 and $x'_1$
- $N_i$ is the number of regular cells between coordinates $x'_{i-1}$ and $x'_i$
- $M_1$ is the number of regular cells between coordinates 0 and $y'_1$
- $M_j$ is the number of regular cells between coordinates $y'_{j-1}$ and $y'_j$
- $P_1$ is the number of regular cells between coordinates 0 and $z'_1$
- $P_k$ is the number of regular cells between coordinates $z'_{k-1}$ and $z'_k$
- moreover, these data must verify :
  $N_1+N_2+...+N_n=N_{x'}$
  $M_1+M_2+...+M_m=N_{y'}$
  $P_1+P_2+...+P_p=N_{z'}$
  From version 8 on, the resulting cell sizes can be inferior to 0.1 cm.
- the data on the neighborhood degree and the frame are the same as for the uniform mesh.

### 17.2.5 BIAS PARAMETER $k$

After establishing the mesh, it is necessary to give the parameters for the weight calculation. TRIPOLI-4® provides two options for bias parameter $k$, either:

- the values of $k$ are given by the user for each composition (from 1 to $n$) and each biasing group (from 1 to `nbgroups`) in order of **decreasing energy** (corresponding to the order in which the bounds have been entered). This is written as follows:

```
INIPOND
        DATA
                name(compo_1)
                k_1 ... k_nbgroups
                ...
                name(compo_n)
                k_1 ... k_nbgroups
END_INIPOND
```

- the values of $k$ are calculated by solving the Placzek equation, after collapsing the pointwise cross-sections in each composition and each biasing group. The syntax is:

```
INIPOND
        AUTO
END_INIPOND
```

If the name of the material is `VOID`, the value of $k$ is set by taking the lowest value among the other values of $k$ and dividing it by two.

### 17.2.6 SMOOTHING OPTION

`INIPOND` is designed to compute importance maps optimized for the use of the Exponential Transform variance reduction technique. Amongst other optimizations, `INIPOND` smoothes importance values to prevent weight discrepancies in the simulation. This behaviour is the default behaviour. However, it can be useful to deactivate this option, for example in case of using the `AMS` (see Chapter 18) variance reduction technique with an `INIPOND` importance map. This can be done by inserting the optional line `SMOOTHING 0` in the `INIPOND` command block:

```
INIPOND
        ...
        SMOOTHING 0
END_INIPOND
```

## 17.3 DIRECTIONS OF INTEREST

- In the case of biasing with discrete detectors (block `DETECT` already given), the directions of interest are constructed by calculating the gradients of the spatial importance.

- If the detection surface is of the analytical type (cylindrical or spherical, with a radius already given in block DETECT_RHO, or plane, with a point belonging to the plane already given in block DETECT), it is supplemented here with data making it possible to establish the detector surface (example: in cylindrical, we set the axis of the cylinder) and hence to establish the directions of interest, pointing from the point of calculation to the detection surface (in this case, the directions of interest are, therefore, independent of the energy group). When a mesh cell is crossed by the detection surface, the null vector is assigned to the direction of interest associated with that cell.

The syntax is as follows, again within the INIPOND block:

```
      INIPOND
*           DATA
*           AUTO
            OMEGA
#                 GRAD_I
#                 SET_OMEGA
                        <ANALYTICAL_BIASING_TYPE>
                        Analytical biasing type data
      END_INIPOND
```

The option GRAD_I in the OMEGA command is also the default option, taken into account if the two keywords OMEGA and GRAD_I are not given (for reasons of backward compatibility, for data files corresponding to versions preceding version 3).
The option SET_OMEGA indicates that a specific analytical form is to be set for the directions of interest (this option requires setting the analytical form for the directions of interest and for the spatial importances). Currently, the possible options for <ANALYTICAL_BIASING_TYPE> are CYLINDER, PLANE, or SPHERE. The data for this analytical biasing type are then specified:

- For cylindrical biasing:

```
SET_OMEGA
      CYLINDER
            u_x u_y u_z
            c_x c_y c_z
```

where $u_x$, $u_y$ and $u_z$ are the components of the unit vector for the axis of the biasing cylinder and $c_x$, $c_y$ and $c_z$ are the coordinates of a point on the axis. All the coordinates are given in the Cartesian reference frame of the geometry.
If the mesh is cylindrical, the two Z-axes of the cylinders (defining both mesh and biasing) must have parallel unit vectors.

```
SET_OMEGA
        PLANE
                ux uy uz
```

- For plane biasing:

  Coordinates of the vector normal to detection plane ($u_x$, $u_y$ and $u_z$) are given here in the geometry frame. This vector must not be zero, but need not be normalized.

- for spherical biasing:

```
SET_OMEGA
        SPHERE
                cx cy cz
```

  Coordinates ($c_x$, $c_y$ and $c_z$) of the point $C$, center of the biasing sphere, are given in the geometry frame. This point must be included in the biasing mesh defined earlier using the keyword `MESH`.

## 17.4 ENERGY BIASING

As indicated previously, the importance (inverse of weight), in each group, takes the form of $E^{\alpha}$ and is written as follows, in the case of n biasing groups:

```
ALFA
        α1 ... αn
```

The data for this biasing is optional and by default $\alpha_i = 0$.

## 17.5 TIME BIASING

As indicated previously, the importance (inverse of weight), in each group, takes the form of $1+\lambda t$ and a $\lambda$ parameter. It is written as follows:

```
TIME
        λ
```

The data for this biasing is optional and by default $\lambda = 0$.

## 17.6 SPECIAL BIASING

Use of these types of biasing is reserved to very special cases and to expert users only. We advise exhausting all of the other biasing settings options previously described in this chapter rather than using these "special case" options.

### 17.6.1 SPLITTING WHEN ENTERING SPECIFIC VOLUMES `VOLUME_SPLITTING`

This type of biasing splits the particles entering one or more particular volumes in cases where the user is aware that the passage of these particles in these volumes is of interest for the requested scores. As such, the statistics for the requested scores are improved.

This type of biasing has only been developped for neutrons and requires working in simulation mode `CROSS_SPLIT` specified in the directive `SIMULATION` (this option is used to treat the transport between two collisions with sampling performed volume by volume, and is more costly in terms of simulation time than the default option `QUOTA_SAMPLING`).

The syntax for the data of the `VOLUME_SPLITTING` biasing option is as follows:

```
VARIANCE_REDUCTION
        <PARTICLE_TYPE>
        VOLUME_SPLITTING  LIST  nb(volumes)
                volume(1) weight(1)
                volume(2) weight(2)
                ...
                volume(n) weight(n)
END_VARIANCE_REDUCTION
```

where:

- `<PARTICLE_TYPE>` equals `NEUTRON`
- volume(i) is a volume number
- weight(i) is the associated reference weight. The reference weight $\Pi_{ref}$ is used to set the number n of particles to be split.
  n = (int) $\Pi_{ref}$ / W with W being the simulation weight for the particle before the splitting is performed.

Volumes that do not figure on the list have a default reference weight of 1.

### 17.6.2 OPTICAL BIASING

This type of biasing modifies the total macroscopic cross-section for certain volumes by multiplying the total cross-section by a factor. This factor β is defined by the user for each volume such that:
$\Sigma_t^* = \Sigma_t \times (1 - \beta)$

- If $\beta < 0$, the cross-section is increased. For example, if β = -9, the cross section is multiplied by 10.
- If $\beta > 0$, the cross-section is decreased. For example, if β = 0.5, the cross-section is divided by 2.

The syntax is as follows:

```
VARIANCE_REDUCTION
        <PARTICLE_TYPE>
        OPTICAL_VARIANCE_REDUCTION LIST nb(volumes)
                volume(1) beta(1)
                volume(2) beta(2)
                ...
                volume(n) beta(n)
END_VARIANCE_REDUCTION
```

where:

- <PARTICLE_TYPE> equals NEUTRON or PHOTON
- volume(i) is a volume number
- beta(i) is the associated beta coefficient.

### 17.6.3 RUSSIAN ROULETTE CONTROL

When a INIPOND type biasing is defined, the splitting and Russian roulette are automatically managed by the code.
When no biasing is defined, a Russian roulette process is nevertheless used in order to manage small weights before the sampling of each collision (by default, it is used when the simulation weight for the particle is less than 0.8).
The user has now the option to control the Russian roulette process:

- the user may define the global threshold of the Russian roulette (the global value replaces the default threshold of 0.8)
- the user may define a threshold for each volume of a volume list
- the user may completely suppress the Russian roulette (global threshold is 0).

This mode of biasing has only been developped for neutrons. The user may also refer to the global RUSSIAN_ROULETTE option of the simulation command block, available for all particles (see chapter 22).
The syntax for the data is as follows:

```
        VARIANCE_REDUCTION
              <PARTICLE_TYPE>
              CONTROL_RUSSIAN_ROULETTE
*                   LIST  nb(volumes)
                          volume(1) threshold(1)
                          volume(2) threshold(2)
                          ...
                          volume(n) threshold(n)
*                   GLOBAL_VALUE  threshold
        END_VARIANCE_REDUCTION
```

where:

- <PARTICLE_TYPE> equals NEUTRON
- volume(i) is a volume number
- threshold(i) is the associated threshold.

All of the threshold values are real numbers.

### 17.6.4 SPECIAL SOURCE BIASING

The TRIPOLI-4® variance reduction scheme yields to a biased transport but also a biased source distribution. For special cases where no biased transport is needed but a biased source distribution is needed to enhance particular source energies or angles, it can be more convenient to directly adjust biasing parameters for the source. In such cases the user may refer to subsections 12.3.4 and 12.4.7. Note that the source energy and/or angle biasing defined in these subsections are not compatible with the variance reduction techniques defined in the present chapter.

## 17.7 VARIANCE REDUCTION TECHNIQUES FOR COUPLED NEUTRON PHOTON SIMULATIONS

The use of variance reduction techniques for neutrons only is not always sufficient when simulating a coupled neutron photon problem : biasing for both neutrons and photons may then be defined. In this case, the difficulty relies in defining a proper neutron biasing, so as to sample important photons in appropriate areas of the phase space, in terms of photon tallies.

A diagnosis tool can be used to ease the adjustment of this neutron biasing. It enables to find out the highest neutron importances (based on the photon production data given by the nuclear data and the user's defined photon biasing). The diagnosis is done at the end of the biasing initialization, during the initialization step of the run and before stopping: no simulation follows, so as to enable the user to modify the neutron biasing parameters, in the light of the diagnosis results, and rerun without this option.
To request this diagnosis, two VARIANCE_REDUCTION command blocks have to be defined, one for neutrons and one for photons, in the way described in this chapter. As a restriction, biasing energy grids must be the same for neutrons and photons. The special option AUTO_COUPLING has to be requested in the INIPOND part of the **neutron** command block with the following syntax:

```
      INIPOND
            AUTO_COUPLING
                  HEIGHT  height
*                 GRAPH_CONTOURS
*                 GRAPH_NO_CONTOUR
                        Px Py Pz
                        Lu Lv
                        <AXIS>
                        Flag
                         FILE filename
                         THRESHOLD_RATIO threshold
            END_AUTO_COUPLING
      END_INIPOND
```

A list of cells of highest neutron importances (ordered from 1 to a given threshold ratio of the maximal importance) is edited for each biasing group in the results file (the coordinates of their center and the ratio of their importance over the maximum neutron importance of the group are given) and a coupled neutron importance map file is produced (named "<inputdeck>.importanceCoupledMap.bcd" in monoprocessor mode and "<inputdeck>.importanceCoupledMap.<n>th_task.bcd" in parallel mode, where <inputdeck> stands for the name of the TRIPOLI-4® input deck file). The option THRESHOLD_RATIO enables to modify the threshold for the edition of cells of highest importance. The value of threshold must be between 0 and 1, the default value being 0.5.
The calculation is then stopped. Therefore, it should be noted that no result needing the simulation of batches (even the first ones) can be obtained when requesting this diagnosis (e.g., score editions or importance maps dumped after monitoring of the first 5 batches, such as requested by the STORE_IN_FILE option with the ASCII format). If needed, these results and dumps should be requested in the next run.
When the optional graphical output is requested in addition, the cell centers of highest neutron importance are displayed on 2D-views. height is the projection height on these 2D-views, GRAPH_CONTOURS enables to display also the geometry contours, whereas GRAPH_NO_CONTOUR displays none of them. The parameters of the window are input in the same way as for the display of importance maps (see section 17.8).
Several graphs may be requested with several GRAPH_CONTOURS or GRAPH_NO_CONTOUR commands. Flag is 0 if a Postscript file is not required. In this case, a window is displayed and if the user *a posteriori* decides

to dump the associated Postscript file, he can do it with the **Shift-P** command in the graphical window. Otherwise `Flag` is 1 and there is no display window but the associated Postscript file is dumped.
The option `FILE` allows for the user to specify the associated file name. By default, PostScript files are named `"<inputdeck>_highest_I_groupM_N.ps"`, otherwise they are named `"filename_groupM_N.ps"` where *M* is the number of the neutron biasing group and *N* is the number of the graph. They are numbered in the order in which the graphs are requested.
The colors of the displayed points are {red; orange; yellow; green; cyan; blue} for a ratio to the maximal importance of {1 to 0.99; 0.99 to 0.9; 0.9 to 0.8; 0.8 to 0.7; 0.7 to 0.6; 0.6 to 0.5} respectively.
The user can then perform a complete coupled neutron photon simulation by using the diagnosis results to adjust the neutron biasing parameters (position of the `INIPOND` detectors, β factors).

## 17.8 GRAPHICAL DISPLAY

It is possible to monitor the importance maps by a 2D-view of the geometry. The view is orthogonal to one of the axes ($O'x'$, $O'y'$ or $O'z'$) of the biasing reference frame. The display window is defined by a point (the point at the bottom left of the window) and by its dimensions along the two axes defining the 2D-view. The contours plotted correspond to lines of equal importance. The code plots one graph per biasing group. The syntax is as follows:

**GRAPH**
**$P_x$ $P_y$ $P_z$**
**Lu Lv**
**<AXIS>**
**Flag**
***FILE filename***

where `<AXIS>` = `X`, `Y` or `Z` and `(Px,Py,Pz)` is the lower left-hand point of the window, the coordinates being given in the geometry reference plane $OX, OY, OZ$. `Lu` and `Lv` are the dimensions of the display window.
In the `X` case: `Lu` is along $y'$ and `Lv` along $z'$. In the `Y` case: `Lu` is along $x'$ and `Lv` along $z'$. In the `Z` case: `Lu` is along $x'$ and `Lv` along $y'$.
If the mesh is cylindrical, the syntax is exactly the same, with `X`, `Y`, and `Z` respectively corresponding to the three axes of the orthonormal reference frame associated with the cylindrical coordinates.
Note: Once again, for a cylindrical mesh, if the user requests a 2D-view perpendicular to the `X` or `Y` axis, it is necessary for its angular discretization $N_{\theta'}$ to be adequate, which, in practice, means at least 5 for complete angular coverage of $[0, 2\pi]$; otherwise, there is a risk of too many cells being projected onto the 2D view and the map obtained being illegible (too dark).
Several graphs may be requested. `Flag` is 0 if a Postscript file is not required. In this case, a window is displayed and if the user *a posteriori* decides to dump the associated Postscript file, he can do it with the **Shift-P** command in the graphical window. Otherwise `Flag` is 1 and there is no display window but the associated Postscript file is dumped.
The option `FILE` allows for the user to specify the associated file name. By default, PostScript files are named

"importance_map_view.groupM_N.ps", otherwise they are named "filename.groupM_N.ps" where $M$ is the number of the energy group and $N$ is the number of the graph. They are numbered in the order in which the graphs are requested.
The importance maps are useful for checking the data. However, the 2D-views can give the impression that several surfaces of equal weight intersect (which is obviously impossible). This is merely a problem of graphical representation ("spider web" effects can occur).
It is possible to obtain importance maps for several 2D-views.

## 17.9 HINTS CONCERNING BIASING

### 17.9.1 HINTS CONCERNING INIPOND MODULE

Care must be taken not to place the detectors in regions diametrically opposite in the phase space (neutrons and photons cannot be efficiently driven in opposite directions). If necessary, it is perfectly possible to perform two different calculations for requested tallying in the neighboring regions with biasings corresponding to the tallies requested. Consideration should also be given to using symmetry of the problem and cylindrical meshes.

#### 17.9.1.1 The automatic case

In this case, the code collapses the cross-sections of different media in the multigroup grid of the biasing. Then, it deduces from these multigroup cross-sections an "opacity" (or attenuation coefficient) for the medium in each group as a function of the multigroup transport cross-sections and the slowing structure.

- The biasing grid must, therefore, be adapted to the case in point. In a conventional reactor physics problem covering energies from 20 MeV to 1.E-11 MeV, a basic biasing grid limited to 6 groups $[20.,4.5,2,0.5,1.E-3,5E-6,1.E-11]$ is recommended by default. The principle is to properly see the different energy components of the problem, i.e., the region above the fission spectrum, then the high, medium, and low parts of the fission spectrum, then an epithermal component and, finally, a thermal group. On the other hand, there should not be too many groups as the cost of biasing (initialization and storage in memory of the biasing maps) is proportional to the number of groups. However, for a fast problem, for example, with a lower bound energy of 1 MeV, case corresponding to a $n \rightarrow p$ reaction for sulphur), the proposed grid is inefficient. In such cases, it is better to adopt a grid of type $[20.,8,4.5,2,1]$. The grid has fewer groups and therefore is less costly, and in addition variance reduction will be more effective for fast neutrons.
- Media that are empty or of very low density must be handled carefully. The collapsing mechanism generates very low attenuation coefficients (smaller by several orders of magnitude) than those of other media. This results in importance functions that vary greatly from one medium to another, and hence in splitting which degrades calculation quality (sometimes up to a point where convergence is prevented). It is, therefore, recommended to give the compositions a name beginning with the string "VOID". The code then calculates the attenuations in these media as a function of the other media, in order to smooth the importance gradients.

#### 17.9.1.2 Mesh

- The cost in terms of memory and cpu time is proportional to the number of cells. It is, therefore, required to optimize the mesh size for the problem. With more than 10,000 cells, the cost of biasing can become prohibitive. However, there must be a sufficient number of cells for the paths linking the cell centers to see the paths taken by particles migrating in the geometry. Accordingly, the `DIVISION WINDOW` option makes it possible to locally refine the size of the mesh and thus achieve substantial savings. Or else, the `MESH_VARIABLE` command allows a better adjustment of the cells sizes with respect to the different compositions associated to the volumes of the geometry.
- The geometry must be convex. More precisely, if two cell centers are internal to the geometry, the edge linking them must be included in the geometry. (If the geometry of the problem is not naturally convex, it is then necessary to contain it in a box or cylinder, depending on whether a cartesian or cylindrical mesh is planned.) The mesh must cover all the geometry and should not be tangential to the boundary of the geometry. It must, therefore, extend beyond it, but by the least possible amount.
- For cylindrical meshes, the value of $\pi$ used for defining the angular component of the mesh must be precise to avoid the code stopping with messages indicating overlapping of meshes or, conversely, meshes indicating that the range $[0, 2\pi]$ is not covered by the mesh. It is, furthermore, strongly recommended to verify that in the part of the output file relating to the initialization phase, there is no message that diagnoses this case. This is because, in such a case, a calculation may end up with particles in the angular zone that is not covered by the mesh, causing the code to stop at that point. It is, therefore, advisable to have an angular mesh that extends very slightly beyond the zone to be covered (as for Cartesian meshes) but without causing overlapping of meshes at $2\pi$ (which cannot arise in the Cartesian case). The value used internally by the code is $\pi = 3.14159265$. Usually, a value of 3.141593 is adequate for defining the cells while ensuring that the angular mesh does not extend much beyond $[0, 2\pi]$.

## A biasing definition (different from the criticality case of the chapter 5)

```
VARIANCE_REDUCTION
NEUTRON
GRID
4
20.  3.  5.E-6 1.E-11
DETECT
1
0.  407.  0.
0.20
MESH
10 40 10
110.1 50.5 110.1
3
FRAME
CARTESIAN
-550.1 -1000.1 -458.
1.  0.  0.
0.  1.  0.
0.  0.  1.
DIVISION WINDOW
-300 -107 -200
300 107 200
END_MESH
INIPOND
AUTO
END_INIPOND
END_VARIANCE_REDUCTION
```

# Chapter 18

# THE ADAPTIVE MULTILEVEL SPLITTING VARIANCE REDUCTION TECHNIQUE

The Adaptive Multilevel Splitting (AMS) method [74] is a variance reduction method, based on splitting operations, which can be used as an alternative to the standard variance reduction techniques of TRIPOLI-4® presented in Chapter 17. It is based on an iterative process performed for each batch of the simulation. During each iteration, particle trajectories are simulated in an analog way (pure analog or possibly including the implicit capture default option of TRIPOLI-4®), then those particles are sorted with respect to the maximum of an importance function evaluated along the trajectories. For each iteration of the algorithm, the sorting process allows for the definition of a splitting level, which adaptively distinguishes the most interesting particles from the others. All trajectories below the splitting level are removed from the simulation, and resampled by duplicating some of the better ranking trajectories at their intersection with the current splitting level. Finally, a multiplicative correction is applied to the usual Monte Carlo tally, taking into account the number of completed AMS iterations.

Notes:

- The AMS can be used either for single-particle problems or for coupled problems including neutrons and photons, or photons, electrons and positrons.
- It can be used together with gamma spectrometry scores, which is even the only way with TRIPOLI-4® to perform variance reduction with the DEPOSITED_SPECTRUM response.

## 18.1 SYNTAX

```
AMS
        TARGET  <target_definition>
        SPLIT  proportion
        CROSS_SPLIT  <flag>
        RESAMPLE_COLLISIONS  <flag>
        IMPORTANCE
                Description of the importance function
        END_IMPORTANCE
END_AMS
```

Within the AMS block, the mandatory keywords are TARGET and IMPORTANCE. The default values for other parameters are specified in the following sections.

## 18.2 TARGET DEFINITON

The keyword TARGET is used to define the volume of interest for the simulation (typically a detector). It is the responsibility of the user to ensure that the defined target is in accordance with the importance function used during the simulation.
There are two ways of defining the target of the simulation: either with a single target or with multiple targets.

### 18.2.1 SINGLE TARGET

The syntax is as follows:

```
TARGET  volume_id
```

where *volume_id* is either the volume number or name.

### 18.2.2 MULTIPLE TARGETS

AMS can handle a list of volumes as target, enabling the user to divide the target in multiple volumes. It should be kept in mind that the smallest variance for a given volume will be achieved with an AMS simulation using this particular volume as single target.
The syntax is as follows:

```
TARGET  LIST  nb(vol)
        volume_id(1) ... volume_id(n))
```

where *volume_id*$(i)$ is either a volume number or name.

### 18.2.3 SCORING OUTSIDE TARGETS

AMS is capable of returning an **unbiased** estimator of any score within the geometry. However, a variance reduction can only be expected for scores which are estimated either around the target(s) or on the path of the particles between their emission point(s) and the target(s).

## 18.3 SPLITTING PROPORTION

The keyword SPLIT allows the user to change the proportion of particles to be split at each AMS iteration. According to experiment, this parameter has no noticeable influence on efficiency as long as it remains greater than 0.5%. On the other hand, it seems unreasonable in terms of correlations to split more than half of the particles per iteration, which is why it is recommended to keep this proportion below 50%.
If SPLIT is not specified, a value of 1% is assumed.
The syntax is as follows:

```
SPLIT  proportion
```

where *proportion* is expressed in percent.

## 18.4 AMS OPTIONS

### 18.4.1 CROSSING POINTS SPLITTING

In order to have a more precise estimation of the importance of a particle during its transport through the geometry, the AMS algorithm can evaluate its importance when it crosses frontiers between volumes. This allows the method to classify and split particles at crossing points, which are treated in the exact same way as collision points. This functionality is controlled by a flag which can take the value 1 (activated) or 0 (disabled):

```
CROSS_SPLIT  <flag>
```

The default value for this flag is 1, but it can be useful to disable crossing points if the geometry is composed of many volumes, which can significantly increase computation time.

### 18.4.2 COLLISION RESAMPLING

There are two ways of performing particle splitting with AMS.
Either the particles importances are evaluated just after collision points, in which case the particles are duplicated by copying the particles characteristics outgoing a collision.
Or, in some cases such as simulations with highly absorbing materials or importance functions with poor or no account of the particles direction, it may be interesting to evaluate importance **before** collisions, and to duplicate particles at collision sites by resampling collisions.
The syntax for enabling/disabling collision resampling is as follows:

**RESAMPLE_COLLISIONS** <**flag**>

The default value for this flag is 0.

## 18.5 IMPORTANCE FUNCTION

The AMS algorithm uses an importance function to determine which particle tracks are the most interesting. The keyword IMPORTANCE has to be followed by the particle type (NEUTRON, PHOTON, ELECTRON or POSITRON). The corresponding particles will be taken into account by the AMS algorithm with the importance defined inside the block.

Obviously, the AMS cannot perform variance reduction on particles that are not in the simulation. If a single particle type is present in the simulation, this type should be given after IMPORTANCE. If multiple particle types are simulated, an IMPORTANCE block has to be given for each particle type that the AMS should take into account. The particles that do not have a corresponding importance will be left out of the AMS variance reduction algorithm.

To apply the same importance function to all particle types present in the simulation, ALL_PARTICLE can be used as particle type.

### 18.5.1 IMPORTANCE MAPS

When parametrized with MAP, the AMS uses an importance map computed by the INIPOND module (see Section 17.2). Note that this map can also be an external map, possibly coming from another code.
Therefore, it **requires** the VARIANCE_REDUCTION command block in the same data file (and in the case of an external map, the option USE_FILE has to be used). The AMS takes into account every option available to other variance reduction techniques, such as the manual modification of the bias parameters $k$ (i.e. DATA option of INIPOND command) or the importance monitoring.
The keyword MAP has to be followed by the type of particle type (either NEUTRON or PHOTON) corresponding to one of the types specified in the VARIANCE_REDUCTION command block. This allows for example to use a

photon importance while performing AMS on neutrons. The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
       MAP <PARTICLE_TYPE>
END_IMPORTANCE
```

**NOTE**: There is no evidence that importance monitoring (see Section 22.2) improves the efficiency of the AMS. Its use forces the first batches of the simulation to run without AMS, which may lead the simulation to freeze the same way it would if another variance reduction technique was used. This is why it is recommended to disable importance monitoring by adding the line MONITORING 0 to the SIMULATION block.

### 18.5.2 SPATIAL IMPORTANCES

The keywords FROM, TOWARDS and PATH allow the user to set a spatial importance. The importance of a point is in this case computed regarding to a POINT, LINE, PLANE, SPHERE, CYLINDER or PATH.

#### 18.5.2.1 POINT

The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
       <MODE> POINT X Y Z
END_IMPORTANCE
```

The importance of a given point is defined according to the distance $d$ between this point and the point $(X,Y,Z)$. MODE is either FROM or TOWARDS. Using FROM, the importance is $d$, which causes the particles to be pushed away from the point $(X,Y,Z)$, regardless of the geometry. With TOWARDS, the importance is $\frac{1}{d}$, which causes the particles to be attracted to $(X,Y,Z)$.

#### 18.5.2.2 LINE

The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
       <MODE> LINE
              POINT X Y Z
              VECTOR V_X V_Y V_Z
END_IMPORTANCE
```

The importance of a given point is defined according to the shorter distance $d_L$ between this point and the line passing through point $(X,Y,Z)$ and parallel to direction vector $(V_X,V_Y,V_Z)$. `MODE` is either `FROM` or `TOWARDS`. Using `FROM`, the importance is $d_L$, which causes the particles to be pushed away from the line. With `TOWARDS`, the importance is $\frac{1}{d_L}$, which causes the particles to be attracted to the line.

#### 18.5.2.3 PLANE

The syntax is as follows:

**IMPORTANCE** <**PARTICLE_TYPE**>
<**MODE**> **PLANE**
**POINT** *X Y Z*
**VECTOR** $V_X$ $V_Y$ $V_Z$
**END_IMPORTANCE**

The importance of a given point is defined according to the shorter distance $d_P$ between this point and the plane passing through point $(X,Y,Z)$ with normal vector $(V_X,V_Y,V_Z)$. `MODE` is either `FROM` or `TOWARDS`. Using `FROM`, the importance is $d_P$, which causes the particles to be pushed away from the plane. With `TOWARDS`, the importance is $\frac{1}{d_P}$, which causes the particles to be attracted to the plane.

#### 18.5.2.4 SPHERE

The syntax is as follows:

**IMPORTANCE** <**PARTICLE_TYPE**>
<**MODE**> **SPHERE** *R*
**POINT** *X Y Z*
**END_IMPORTANCE**

The importance of a given point is defined according to the distance $d_S$ between this point and the sphere of radius $R$ and center $(X,Y,Z)$. `MODE` is either `FROM` or `TOWARDS`. Using `FROM`, the importance is $d_S$, which causes the particles to be pushed away from the sphere surface regardless of the geometry. With `TOWARDS`, the importance is $\frac{1}{d_S}$, which causes the particles to be attracted towards the sphere surface.

#### 18.5.2.5 CYLINDER

The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
      <MODE> CYLINDER R
             POINT X Y Z
             VECTOR VX VY VZ
END_IMPORTANCE
```

The importance of a given point is defined according to the distance $d_C$ between this point and the surface of the infinite cylinder with radius $R$ which axis passes through point $(X,Y,Z)$ and is directed by vector $(V_X,V_Y,V_Z)$. `MODE` is either `FROM` or `TOWARDS`. Using `FROM`, the importance is $d_C$, which causes the particles to be pushed away from the cylinder surface. With `TOWARDS`, the importance is $\frac{1}{d_C}$, which causes the particles to be attracted towards the cylinder.

#### 18.5.2.6 RING

The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
      <MODE> RING R
             POINT X Y Z
             VECTOR VX VY VZ
END_IMPORTANCE
```

The importance of a given point is defined according to the distance $d_R$ between this point and the ring of radius $R$ and center $(X,Y,Z)$, which axis is directed by vector $(V_X,V_Y,V_Z)$ and is directed by vector $(V_X,V_Y,V_Z)$. `MODE` is either `FROM` or `TOWARDS`. Using `FROM`, the importance is $d_R$, which causes the particles to be pushed away from the ring. With `TOWARDS`, the importance is $\frac{1}{d_C}$, which causes the particles to be attracted towards the ring.

#### 18.5.2.7 PATH

The syntax is as follows:

```
IMPORTANCE <PARTICLE_TYPE>
    PATH
        POINT LIST nb(points)
            X1 Y1 Z1
            ...
            Xn Yn Zn
        STRENGTH s
    END_PATH
END_IMPORTANCE
```

A path is defined as an ordered sequence of points. The importance of a point outside the path is defined using the minimal distance from the point to the path $d_{\perp}$, and the distance along the path from the first point of the path to the orthogonal projection of the point on the path $d_{\parallel}$. The importance is defined as

$$I = d_{\parallel} - s.d_{\perp}$$

so that it increases with the distance along the path and decreases with the distance to the path. The parameter $s$ represents the attraction strength of the path, which influences the slope of the importance function around the path. By default, this parameter is set to 1. If needed, the optional keyword `STRENGTH` must be placed after the list of points coordinates.

#### 18.5.2.8 VOLUME IMPORTANCE WEIGHTING

In addition to the importance function, each volume of the geometry carries an importance factor that weights the importance of any point within its boundaries. By default, this value is set to 1. The user can change the weight of any number of volumes using the following syntax:

```
IMPORTANCE <PARTICLE_TYPE>
    ...
    VOLUME num(vol1) ... num(voln) WEIGHT w
END_IMPORTANCE
```

## 18.6 RELATED OPTIONS IN OTHER COMMAND BLOCKS

Some existing options of the `SIMULATION` command block (see Section 22.2) and the `VARIANCE_REDUCTION` (see Chapter 17) command block can be used alongside the `AMS` and have an impact on the algorithm behaviour.

### 18.6.1 ANALOG

During the AMS iterations, particle transport is **unbiased**. However, the **implicit capture** may also be used without altering the unbiasedness of the AMS algorithm. It is recalled that the implicit capture is activated by default in TRIPOLI-4®. For a pure analog transport during the AMS iterations, the user has to use the ANALOG option of the SIMULATION command block, which deactivates the implicit capture.

### 18.6.2 MONITORING

When using the importance map from the VARIANCE_REDUCTION block, the AMS is **by default** deactivated during the 5 first batches, thus enabling TRIPOLI-4® to monitor importance and adjust energy-group biasing. The monitoring process may eventually be quite expensive, and there is no assurance that the resulting importance map is any better for the AMS use. Therefore, we recommend to deactivate this feature using the optional line MONITORING 0 in the SIMULATION command block.

### 18.6.3 SMOOTHING

The INIPOND (see Section 17.2) module is designed to compute importance maps optimized for the use of the Exponential Transform variance reduction technique. Amongst other optimizations, INIPOND smoothes importance values to prevent weight discrepancies in the simulation. However, this process has no benefits if the importance is used to perform AMS. In fact, importance map smoothing is inherently linked to a loss of information. When using the AMS with an importance map computed by INIPOND module, it is advised to disable importance smoothing using the optional line SMOOTHING 0 in the INIPOND command block.

## 18.7 RESTRICTIONS AND FORBIDDEN OPTIONS

When using the AMS, the simulation mode of the SIMULATION command block (see Section 22.2) must be SHIELDING (which is the default option) or FIXED_SOURCES_CRITICALITY.
Some options are forbidden when used together with the AMS: PACKET_LENGTH option (see Section 22.2) and score filtering (see Section 15.6).

# Chapter 19

# THE WEIGHT WINDOWS VARIANCE REDUCTION TECHNIQUE

The weight windows method is a variance reduction method, based on splitting and Russian roulette operations, which can be used as an alternative to the standard variance reduction techniques of TRIPOLI-4® presented in Chapter 17. This technique is commonly used by Monte Carlo transport codes, such as MCNP.

Notes:

- The implementation of the weight windows technique in TRIPOLI-4® allows for playing weight windows games at several possible event types of particle trajectories, such as collision and surface crossing.
- Weight windows are available in TRIPOLI-4® only for spatial meshes superimposed on the geometry, but are not available for volumes of the geometry.
- Weight windows games need the knowledge of so-called *lower weight bounds* (or *lower bounds*) data which can be provided either by an importance map produced by the INIPOND module of TRIPOLI-4® (see Chapter 17), or by adjoint fluxes values given on a spatial mesh and an energy grid (*e.g.* retrieved from the output of a deterministic code), or by the output of version 3.0 of ADVANTG deterministic code [90] (directly through a so-called *wwinp* file readable by version 5 of MCNP and also by TRIPOLI-4® within the weight windows feature).

  According to the common weight windows technique (see Reference [7]), each weight window is defined by a lower weight bound $W_l$ (threshold for the Russian roulette game, played for particles whose statistical weight is below $W_l$), a survival weight $W_s$ (outgoing weight of particles surviving the Russian roulette game) and an upper weight $W_u$ (threshold for splitting operations, played for particles whose statistical weight is above $W_u$). No action is taken for particles within the window (statistical weights between $W_l$ and $W_u$). Only lower weight bounds data have to be given in each spatial cell and energy group. Two constants, $C_u$ and $C_s$ (ratios also given as input data), allow to get $W_s$ and $W_u$ in all cells and groups with the following relations:

$$W_s(P) = C_s\, W_l(P) \tag{19.1}$$

$$W_u(P) = C_u\, W_l(P) \tag{19.2}$$

where $P$ is the current point in phase space (actually only space and energy variables are considered here).
The following formula gives the relation between lower bounds and adjoint flux values:

$$W_l(P) = \frac{2R}{(1+C_u)\,\phi^+(P)} \tag{19.3}$$

where $R$ is a total response renormalization factor and $\phi^+(P)$ is the adjoint flux at $P$.

- When using weight windows, TRIPOLI-4® automatically applies importance values on the source (*i.e.* source biasing), so as to simulate source particles with emission statistical weights already in agreement with the weight windows data applying to the related source area. These importance values are retrieved from the adjoint fluxes or derived from the lower bounds weight data, as given by the previous formula.

## 19.1 SYNTAX

As for the standard variance reduction technique of TRIPOLI-4® presented in Chapter 17, the syntax for the weight windows technique is defined between the keywords `VARIANCE_REDUCTION` and `END_VARIANCE_REDUCTION` with the specific following syntax:

```
VARIANCE_REDUCTION
        <PARTICLE_TYPE>
         WEIGHT_WINDOWS filename
END_VARIANCE_REDUCTION
```

- `<PARTICLE_TYPE>` is the particle type related to the weight windows data of the current `VARIANCE_REDUCTION` command block and can be either `NEUTRON` or `PHOTON`;
- `filename` is the name of the file containing the weight windows data (with its absolute path) according to the format detailed in the next section.

## 19.2 WEIGHT WINDOWS FILE SYNTAX

The syntax of the weight windows data file is as follows:

**WEIGHT_WINDOWS title**
**FORMAT format_number**
**GRID energy grid values**
**MESH or MESH_VARIABLE mesh values ...**
**... END_MESH or END_MESH_VARIABLE**
**ADJOINT adjoint flux values ... END_ADJOINT**
***MODE*** *<ww_mode>*
**CU_RATIO Cu**
**CS_RATIO Cs**

- `title` is any chain of characters (a maximum of 256);
- `format_number` is an integer standing for the format number with the following meaning:
  - with format number 1, adjoint flux values are given as such;
  - with format number 2, adjoint flux values are given as $-ln(\phi^+)$, which is consistent with the way of storing importance values in TRIPOLI-4® and can be retrieved when using the keyword `STORE_IN_FILE` with the standard variance reduction technique of the code (see Chapter 17) (*e.g.* for an importance map generated by the `INIPOND` module);
  - with format number 3, lower bounds data of weight windows are given, such as provided by version 3.0 of `ADVANTG` code, with a specific syntax and additional optional keywords detailed in the next section.
- `GRID`, `MESH` or `MESH_VARIABLE` are required only with format numbers 1 and 2 and the energy grid values and mesh values follow the same ordering convention as for data dumped when using the keyword `STORE_IN_FILE` of Chapter 17;
- For format numbers 1 and 2, adjoint flux values $\phi^+$ or $-ln(\phi^+)$ are given after the keyword `ADJOINT`, with the following order of loops (from the most exterior one): the energy group of the grid, then $k$, then $j$, then $i$ ($i$,$j$,$k$ being the cell indices);
- `MODE` is optional and allows the user to choose the event types in trajectories where weight windows are played during the simulation: with `AFTER_COLLISIONS` (which is the default value for *<ww_mode>)* weight windows are played before each flight between two collisions; with `BEFORE_AND_AFTER_COLLISIONS` they are played before each flight and before each collision; with `AFTER_COLLISIONS_AND_AT_SURFACES` they are played before each flight and at each surface crossing between volumes.
- `CU_RATIO` is the ratio of upper and lower weight bounds and `CS_RATIO` is the ratio of survival weights and lower weight bounds. These ratios are constant for all weight windows and can be chosen as follows: $C_u = 5$ and $C_s = 3$, according to the recommendations in Reference [7].

## 19.3 SPECIFIC SYNTAX WHEN DATA ARE PROVIDED BY ADVANTG CODE

When the format number is 3, then data are directly provided by the output of version 3.0 of ADVANTG code and the syntax of the weight windows data file is as follows:

```
        WEIGHT_WINDOWS  title
        FORMAT  3
        wwinp_filename
        MODE <ww_mode>
#       NOT_FOR_ZERO_VALUES
#       SWITCH_ZERO_TO_MAX_VALUES
        CU_RATIO  Cu
        CS_RATIO  Cs
```

The name of the *wwinp* file is given just after the format number (3), with its absolute path. The format of the *wwinp* file is such as produced by version 3.0 of ADVANTG code.
Cartesian and cylindrical windows are available.
TRIPOLI-4® assumes the following rules if zero values of lower weight bounds are present in the *wwinp* file:

- In case of a zero lower weight bound for a given cell and a given group, the adjoint flux used by TRIPOLI-4® in this cell (for the calculation of the biased source and the birth weight of source particles), is the minimal adjoint flux among the other cells for the same group. Moreover, by default (which is equivalent to the presence of the optional keyword NOT_FOR_ZERO_VALUES), the weight windows are then not played in this cell during particle transport simulation (following the same convention as for MCNP).
- Otherwise, if the optional keyword SWITCH_ZERO_TO_MAX_VALUES is present, the same rule applies for the source, but weight windows are played in this cell during transport simulation with the maximal lower weight bound (which is equivalent to the minimal adjoint flux value) for this group.
- The relation between lower weight bounds and adjoint fluxes is used to retrieve importances and to calculate the biased source of the simulation.

## 19.4 OTHER RELATED SIMULATION OPTIONS

The optional keyword MAX_SPLITTING_PARTICLE_NUMBER can be used in the SIMULATION command block (see Section 22.2), so as to limit the number of particles possibly split whenever weight windows are played (the default being 1000).

# Chapter 20

# THE CADIS VARIANCE REDUCTION TECHNIQUE

Many variance-reduction techniques, including TRIPOLI-4's exponential transform (Chapter 17), require the use of an importance map. For several techniques, it can be proved that the optimal importance map is given by the adjoint flux, i.e. the solution to the adjoint Boltzmann equation.
This chapter describes the CADIS technique, its implementation in TRIPOLI-4, and the input syntax that is required to use it.

## 20.1 PRINCIPLE OF THE CADIS METHOD

The CADIS (Consistent Adjoint-Driven Importance Sampling) technique consists in using a deterministic solver to solve the adjoint Boltzmann equation for the adjoint flux, and using it as an importance map in a direct Monte Carlo calculation. The detector acts as a source for the adjoint equation; thus, different detectors (and different detector response functions) result in different importance maps. On the other hand, the particle source does not appear in the adjoint Boltzmann equation; therefore, CADIS importance maps for a given detector can be used for any particle source.
In TRIPOLI-4, the CADIS technique is implemented via a coupling with the IDT deterministic solver [104]. IDT is developed at CEA/SERMA. It is a multi-group, $S_N$ solver that uses the method of short characteristics. Starting with version 12, it is shipped together with TRIPOLI-4. IDT can be used for photon calculations and neutron calculations; coupled neutron-photon calculations are not supported yet. The code is shipped with a photon file and a neutron file in the HDF5 format containing microscopic multi-group cross sections. The photon cross sections use a 94-group energy grid, while the neutron cross sections use a 281-group grid. The HDF5 schemes used by these files are documented in technical notes [82, 83].
In the current implementation of the CADIS scheme, TRIPOLI-4 uses the exponential transform as a variance-reduction method by default (see Chapter 17). Other variance-reduction methods may be used by writing the generated importance map to a file and using it as an external map.

## 20.2 MODELLING AND APPROXIMATIONS

In the IDT solver, mesh cells are assumed to be homogeneous. In practice, a mesh cell can overlap with more than one TRIPOLI-4 volume, with possibly different compositions. It is important to stress that IDT will assume that cells are filled with the material that is found at the center of the cell. This admittedly crude approximation will be improved upon in future versions of the code. For the moment we recommend users to adapt the coarseness of the space mesh to the complexity of the geometry.

## 20.3 SYNTAX

```
VARIANCE_REDUCTION
        NAME <biasing_name>
        <PARTICLE_TYPE>
        GRID
                Grid data
        MINIMAL_IMPORTANCE <importance>
*       MESH
                Uniform mesh data
        END_MESH
*       MESH_VARIABLE
                Nonuniform mesh data
        END_MESH_VARIABLE
        IDT
                IDT data
        END_IDT
        STORE_IN_FILE <format>
END_VARIANCE_REDUCTION
```

The CADIS scheme uses a slightly modified version of the standard `VARIANCE_REDUCTION` block. The syntax and the semantics of the `GRID`, `MESH` and `MESH_VARIABLE` keywords are the same as in the `VARIANCE_REDUCTION` block, described in Chapter 17, with the following caveats:

- the recommended way to set the energy grid with the `GRID` keyword is to read it from the default IDT microscopic cross section file (see below). The syntax is the following:

```
GRID FROM_IDT_XS_T4PATH
```

  where the string T4PATH refers to the path file using the new path file format (see Section 3.1.2) with the `idt_photon_xs_file` and `idt_neutron_xs_file` keywords.

  Alternatively, the cross section grid can be read from another HDF5 microscopic cross section file with the following syntax:

**GRID FROM_IDT_XS_FILE** <*filename*>

Finally, the grid can be explicitly specified, following the standard syntax of Chapter 17.

Note that, in all cases, the given energy grid must match the grid stored in the microscopic cross section file that is actually used by the solver (see below). No cross section condensation takes place at the moment, and **an error is emitted if the given energy grid does not match the cross section grid**.

Note also that the lower simulation energy bound must be greater than or equal to the lower energy bound of the energy grid used by IDT. Likewise, the upper simulation energy bound must be less than or equal to the upper energy bound of the IDT energy grid. These rules are not specific to CADIS simulations; they also apply to other variance-reduction techniques (see Sec. 17.2.1). However, in the case of CADIS, the energy grid is essentially imposed by the IDT cross-section file. Therefore, the user may need to adjust the energy simulation domain to match the cross-section grid (see Sec. 22.2 for the syntax).

- the optional `MINIMAL_IMPORTANCE` keyword sets the smallest value that is accepted in the generated importance map; values below this threshold are truncated to the threshold value. In problems with strong attenuation, the use of `MINIMAL_IMPORTANCE` is recommended with a very small value (e.g. `1e-30`).
- the `MESH`/`MESH_VARIABLE` keywords follow the syntax of Section 17.2.4 and define the space mesh for the importance map as seen by TRIPOLI-4. For use with IDT, the frame type of the space mesh must be `CARTESIAN`. The actual space mesh used internally by the IDT solver may be finer (see `CONVERGENCE` keyword below). Although this possibility exists, it is generally recommended that IDT and TRIPOLI-4 use the same space mesh: the TRIPOLI-4 importance map is anyway smaller than the memory required for the IDT calculation, so there is little gain in homogenizing it on a coarser mesh.
- as with `INIPOND`, the IDT importance map can be stored in a file for further analysis or reused with the `STORE_IN_FILE` keyword (see Chapter 17).

### 20.3.1 IDT PARAMETERS

The `IDT` block has the following syntax:

```
IDT
        TARGET <target data>
        SPECTRUM <spectrum data>
        CONVERGENCE <convergence parameters>
        MICROSCOPIC_CROSS_SECTIONS
            <microscopic cross section data>
        END_MICROSCOPIC_CROSS_SECTIONS
        MAX_INNERS  <max inner iterations>
        MAX_THERMALS  <max thermal iterations>
*       CHARACTERISTICS  CONSTANT
*       CHARACTERISTICS  LINEAR
        ALLOW_IDT_IN_PARALLEL
        IDT_SOLVER_OPTIONS
            <IDT solver options>
        END_IDT_SOLVER_OPTIONS
END_IDT
```

The TARGET, SPECTRUM, and CONVERGENCE keywords are required.

#### 20.3.1.1 TARGET keyword

TARGET describes the spatial location of the adjoint source, i.e. of the detector. The arguments of the TARGET keyword are names of scores, as defined in the SCORE block. The syntax is

**TARGET** <*number of scores*> <*score name 1*> ... <*score name n*>

The scores must be defined over a list of volumes (possibly consisting on only one volume), a sum of volumes or an extended mesh. The location of all the scores is jointly taken as the location of the adjoint source.

#### 20.3.1.2 SPECTRUM keyword

SPECTRUM describes the energy response of the detector, i.e. the energy spectrum of the adjoint source. The spectrum can be given as a constant (energy-independent) response, it can be given as a tabulated response or it can be automatically extracted from one of the TRIPOLI-4 responses, defined in the RESPONSES block.
In the first case the spectrum of the adjoint source is given as

**SPECTRUM  CONSTANT** <*response value*>

For example, SPECTRUM CONSTANT 1.0 can be used to represent a flux detector (which is characterized by a response function that is equal to 1 for all energies).

For the second case (tabulated response), the syntax is

**SPECTRUM LIST** <*response value*> <*response value*> ...

Here the response values must be given on the energy grid specified in the VARIANCE_REDUCTION block. If the grid has length $N$, exactly $N-1$ values are expected here.
Finally, for the case of an existing response, the SPECTRUM syntax is

**SPECTRUM RESPONSE NAME** <*response name*>

where <*response name*> must be the name of one of the responses defined in the RESPONSES block (see Chapter 13).

#### 20.3.1.3 CONVERGENCE keyword

The CONVERGENCE keyword describes the space and angular discretization of the mesh used by IDT for solving the adjoint Boltzmann equation. The syntax is

**CONVERGENCE FIXED** <*max optical thickness*> <$S_N$ *order*>

The *max optical thickness* parameter is used to control the coarseness of the IDT internal calculation mesh: IDT starts by considering the mesh provided by the user with one of the MESH/MESH_VARIABLE keywords. It computes the optical thickness of each mesh cell, defined as the product of the total cross section for the (homogenized) cell and the cell size. If the optical thickness is larger than the *max optical thickness* value given by the user, then IDT will automatically refine the internal calculation mesh to meet the optical thickness criterion. See Section 20.4 below for a discussion of the optimal value for this parameter.
The $S_N$ *order* parameter is the order of the $S_N$ angular quadrature to be used by IDT. Larger values of this parameter improve the accuracy of the angular quadrature formulas and help with mitigating ray effects in the importance map; however, note that the memory footprint of IDT is mildly dependent on the value of this parameter. The CPU time also increases when this parameter is increased. A value of 8 is a good starting point.

#### 20.3.1.4 MICROSCOPIC_CROSS_SECTIONS block

The optional MICROSCOPIC_CROSS_SECTIONS block provides the parameters for the computation of the macroscopic cross sections from the microscopic multi-group photon/neutron cross sections. The syntax is the following:

```
MICROSCOPIC_CROSS_SECTIONS
    ORDER <order>

*   PARTICLE <particle type> FILE <HDF5 file name> ALL
*   PARTICLE <particle type> FILE <HDF5 file name> NUCLEI
        <number of nuclei> <nucleus> <nucleus>...

END_MICROSCOPIC_CROSS_SECTIONS
```

The ORDER keyword represents the maximum order of the development of the angular distribution of microscopic cross sections over Legendre polynomials. For the default microscopic cross-section files provided with the installation, this parameter can range from 0 to 6 (inclusive); the default value is 1. It is recommended to use larger values for high-energy neutron problems and for photon problems. Note also that the memory footprint of the IDT solver and the required CPU time scale as the square of this parameter.
The PARTICLE and FILE keywords can be used to customize the microscopic cross section files that should be used by IDT. By default, all nuclei in the geometry will use the files defined by the idt_photon_xs_file and idt_neutron_xs_file keywords in the dictionary file (see Chapter 3). This choice can be overridden on a nucleus-by-nucleus basis with the NUCLEI syntax, or globally with the ALL keyword. Multiple FILE keywords may be given for the same PARTICLE; if the ALL syntax is used together with one or more NUCLEI lines, then the filename specified by the ALL keyword only applies to the nuclei that were not explicitly specified.

### 20.3.2 OTHER KEYWORDS

There are a few other optional keywords that can be used in the IDT block:

- MAX_INNERS can be used to set the maximum number of inner iterations performed by IDT. By default this number is 1000. Reducing this number to a smaller value may speed up the IDT calculation but it might also prevent the code from converging.
- MAX_THERMALS sets the maximum number of thermal iterations in neutron problems. The default value is 50. Reducing this value might prevent IDT from converging.
- CHARACTERISTICS sets the type of characteristics to use. Possible values are CONSTANT and LINEAR, with CONSTANT being the default. Linear characteristics are more precise than constant characteristics for a given mesh size, but their memory footprint is four times larger and they may lead to negative importances in unconverged meshes.
- the IDT_SOLVER_OPTIONS block can be used to pass any option understood by IDT to the solver. The content of this block is copied verbatim into the IDT input file (filename <input_file>_idt.d). A partial list of useful IDT options follows:
  - FissionIntegralPrecision <float>: convergence criterion for the fission integral during outer

iterations. Default: `1e-4`. This option is only relevant for `FIXED_SOURCES_CRITICALITY` calculations.

- `InnersPrecision <float>`: convergence criterion for inner iterations (self-scattering). Default: `1e-5`.
- `ThermalsPrecision <float>`: convergence criterion for thermal iterations. Default: `1e-5`.
- `AccEqPrecision <float>`: convergence criterion for acceleration (which drives the convergence in each group, in the thermals and in the outers). Default: `1e-4`. The value of this parameter should be at most, roughly, of the order of the smallest of `ThermalsPrecision`, `FissionIntegralPrecision` and `InnersPrecision`.
- `MaxOuters`: maximum number of outer iterations. This parameter is only relevant in `FIXED_SOURCES_CRITICALITY` mode. Default: `50`.
- `MaxThermalsBeforeConvergence <integer>`: maximum number of thermal iterations before convergence of outer iterations; after the outer iterations have converged the value of `MAX_THERMALS` is used. Default: `1`. Good values are in the 1–5 range.
- `MaxInnersBeforeConvergence <integer>`: maximum number of inner iterations before convergence of thermal iterations; after the thermal iterations have converged the value of `MAX_INNERS` is used. Default: `1`. Good values are in the 1–5 range.

- see the next section for the use of the `ALLOW_IDT_IN_PARALLEL` keyword.

## 20.4 CAVEATS

A few important points about the usage of IDT must be brought to the attention of the reader.

- The mesh cells containing the adjoint source are determined by checking if the center of the cell falls within the `TARGET` score region (volume or extended mesh). In the case of a small detector volume, it is possible that no cell center falls into the detector region. In this case, TRIPOLI-4 will emit an error. Refining the mesh around the detector region should fix the error. It is often also possible to slightly displace the whole mesh in such a way that the center of at least one cell falls into the detector.

- If more than one `TARGET` score is requested, TRIPOLI-4 will not emit an error as long as at least one detector region contains one cell center. In this case, detectors that do not contain any cell center may be silently ignored from the adjoint source. If you want to make sure that all your detectors are correctly picked up by IDT, you can check the generated IDT input file. Alternatively, you can run replace the list of `TARGET` scores with one score at a time and check that TRIPOLI-4 is picking it up correctly (i.e. no error message is emitted).

- **It is of the utmost importance to make sure that the space discretization used by IDT is converged**. A coarse calculation mesh will result in inaccurate importance maps at best, and unphysical maps at worst. The user can ensure that the internal calculation mesh is sufficiently fine either by setting the *max optical thickness* parameter to a moderately small value (of the order of 1) or by refining the mesh by hand. However, in case the *max optical thickness* parameter is used, the importance map produced by IDT will be homogenized onto the user-provided TRIPOLI-4 mesh for the transport phase. This

throws away most of the advantage of using a finely discretized importance map. Since the memory requirements are anyway dominated by the IDT resolution phase, it is recommended to set the *max optical thickness* to a large value, such as 1000. This will practically prevent IDT from automatically refining the internal calculation mesh and force it to use the user-defined mesh. It is also recommended **to study the convergence of the space discretization** by manually refining the mesh and verifying the stability of the importance map with respect to small changes in the discretization coarseness. Note that convergence is expected to require cell optical thicknesses of the order of 1. This typically means using much finer meshes than those required by `INIPOND` (Chapter 17).

- The **energy** convergence of the IDT calculation can be studied by inspecting its output file (filename `<input_file>_idt.listing`). It is recommended to check the following points:
  - The maximum number of inner iterations (`MAX_INNERS` keyword) should not be reached in any group, at least during the last thermal iteration, or the last outer iteration for fixed-source problems with fission (`FIXED_SOURCES_CRITICALITY`, see Chapter 22). The maximum number is also echoed at the beginning of the output file.
  - The maximum number of thermal iterations (`MAX_THERMALS` keyword) should not be reached. The maximum number is also echoed at the beginning of the output file.
  - The maximum number of outer iterations (for fixed-source problems with fission) should not be reached. The maximum number is given at the beginning of the output file.

  If any of these numbers are actually reached, it is likely that the IDT calculation has not converged. A warning message of the form "`THERE ARE UNSATISFIED CONVERGENCE CRITERIA`". Increasing the maximum number of iterations may help.
- In case the *max optical thickness* parameter is used, note that internal mesh refinement performed by IDT does not resolve any heterogeneity in the TRIPOLI-4 cell. That is, if a user-provided mesh cell contains several materials, IDT will still consider the cell as homogeneous (using the material found at the center), even if the internal calculation mesh is refined.
- In adjoint calculations, the thermal adjoint flux influences the fast adjoint flux. This is the opposite of what happens in direct calculations. If your detector is sensitive to thermal neutrons, you should pay careful attention to the convergence of thermal iterations (see `MAX_THERMALS` and `ThermalsPrecision` above).
- By default, IDT uses internal shared-memory parallelism to accelerate the resolution of the adjoint problem. The number of threads used by IDT is typically equal to 8.
- The maximum number of threads used by IDT can be set with the `OMP_NUM_THREADS` environment variable. IDT will never use more threads than the value given in `OMP_NUM_THREADS` if it is defined, and it may use fewer. As mentioned above, there is little gain in setting `OMP_NUM_THREADS` to a value larger than 8.
- Because of IDT's shared-memory parallelization, it is forbidden by default to run IDT in a parallel TRIPOLI-4 calculation (see Chapter 33). The recommended way to run a parallel TRIPOLI-4 calculation

with an IDT importance map is to generate the map in a mono-processor calculation, write it to a file using the STORE_IN_FILE keyword, and reuse it with USE_FILE (Chapter 17). However, if you know what you are doing, you can use the ALLOW_IDT_IN_PARALLEL keyword to run IDT in a parallel calculation.

- If the FILE keyword of the MICROSCOPIC_CROSS_SECTIONS block is used, then care must be taken to ensure that all nuclides use the same microscopic cross-section energy grid. The correct grid must also be declared in the GRID keyword of the VARIANCE_REDUCTION block; consider using the FROM_IDT_XS_FILE keyword.
- The IDT input file is always written to a file (filename <input_file>_idt.d), so that it can be inspected. By default, the importance map is not written to a file, unless the user requests it with STORE_IN_FILE.
- When using the IDT block, TRIPOLI-4 always uses the exponential transform as a variance-reduction technique. However, importance maps written to external files (STORE_IN_FILE) may be suitable for the use with other variance-reduction techniques (Adaptive Multilevel Splitting, see Chapter 18; weight windows, see Chapter 19).
- Weight adjustment of the importance map (MONITORING keyword) is automatically disabled when using IDT.

## 20.5 EXAMPLE

Here is a full example of the VARIANCE_REDUCTION block for a neutron CADIS calculation:

```
VARIANCE_REDUCTION
  NEUTRON
  GRID FROM_IDT_XS_T4PATH
  MINIMAL_IMPORTANCE 1e-35
  MESH_VARIABLE
    60 60 20
    0 4 80.001 52 240.001 4 320.002
    0 4 80.001 52 240.001 4 320.002
    0 20 100.002
    3
    FRAME CARTESIAN
    -160.001 -160.001 -50.001
    1 0 0
    0 1 0
    0 0 1
  END_MESH_VARIABLE
  IDT
    TARGET 1 DOSE_N_SPHERE
    CONVERGENCE FIXED 100.0 8
```

```
    MAX_INNERS 200
    SPECTRUM RESPONSE NAME DOSE_N
  END_IDT
  STORE_IN_FILE ASCII
END_VARIANCE_REDUCTION
```

# Chapter 21

# THE HOMOGENIZATION COMMAND

With the TRIPOLI-4® code, it is also possible to homogenize a composition. This option is useful for calculating cross-sections with a motif of one or more fuel assemblies in an infinite medium. It is helpful for making comparisons with lattice codes. It is necessary to request the cross-sections in a grid with a small number of groups to obtain adequate statistical accuracy for the cross-sections (typically 0.5%). The command used is `HOMOGENIZE`.

When this option is requested, the code generates homogenized multigroup cross-sections in a file. A second TRIPOLI-4® calculation can then be made with these cross-sections.
The syntax is as follows:

```
HOMOGENIZE
        DIFFUSION 1
        NBMAT nb(compo)
                name(compo-1)
                 VOLUME LIST nb(volu)
                        num(vol_1) ... num(vol_N)
                 OUTPUT ALL
                name(grid)
                ...
                name(compo-n)
                 VOLUME LIST nb(volu)
                        num(vol_1) ... num(vol_N)
                 OUTPUT ALL
                name(grid)
        FILE filename
END_HOMOGENIZE
```

where:

- `nb(compo)` is the number of compositions to be homogenized;

- `name(compo-i)` is the name of the `i`-th composition to be homogenized. The user can give any name for this compostion;
- `num(vol`$_1$`)` ...`num(vol`$_N$`)` is the list of the `nb(volu)` volumes in the composition;
- `name(grid)` is the energy grid on which the composition is homogenized. The grid must have previously been defined in the `GRID_LIST` command;
- `filename` is the name of the file in which the homogenized multigroup cross-sections are stored, plus the extension `-homog`. By default, the filename is `-homog` alone.

A subsequent TRIPOLI-4® calculation can then be made using these cross-sections. All that is required is to declare that the compositions are made using cross-sections from the file generated using the command (see `COMPOSITION` cross-section).
Currently, the total cross-sections $\Sigma_t$, the absorption cross-section $\Sigma_a$, and the product of the fission cross-section and the number of fission neutrons $\nu\Sigma_f$ are calculated using this command. These outputs are consistent with the APOTRIM format of APOLLO2.
Keyword `NBMAT` is required (although not present in version 4.1).
By indicating `OUTPUT ALL`, the code supplies the self-shielded cross-sections for the compositions to be homogenized. Output is directed to the conventional output file, but is not taken up into the `-homog` extension cross-section file.
When `DIFFUSION 1` is indicated, the code outputs scattering coefficients. There must, however, be only one medium to be homogenized, and it must be **infinite** (consequently, reflection from all the boundary surfaces of the geometry is required). *Important: for the time being the results are not guaranteed and should not be used in project studies*.

*Important: The homogenization command is not allowed in parallel mode*.

# Chapter 22

# DEFINING NUMERICAL SIMULATION PARAMETERS

Simulation parameters are defined between the keywords SIMULATION and END_SIMULATION.
This section of the TRIPOLI-4® data relates to the simulation phase itself and allows for the simulation parameters to be defined.

## 22.1 SIMULATION PARAMETERS

Some parameters must be specified (see syntax in section 22.2):

- the number of batches ;
- the number of particles per batch ;
- the type of particles to be tracked (NEUTRON, PHOTON, ELECTRON and POSITRON).

Other parameters are optional:

- the simulation mode: radiation shielding, criticality, fixed-source sub-criticality, multiple, critical parameter search or adjoint criticality (**default: SHIELDING**). It should be noted that prompt fission gamma are simulated for all problem types, including radiation shielding, whenever photons are tracked. For further details on the multiple simulation mode, see Section 22.19.
- DISCARD <n>, used in criticality mode (and for the critical parameter search), allows the code to discard n batches in the calculation of scores (only for the scores associated with responses like flux, reaction rates,..., but not for keff). For the keff, an automatic convergence is used. The optimal number of batches to discard in order to satisfy the convergence criteria is calculated, even if the keyword DISCARD is present (see complements in section 33.4).
- it is possible to request convergence to a given precision, over a part of the score. In this case, CONVERGENCE 0.05 0.80 means that convergence is required for the scores at 5%, over 80% of the score integrated over the entire energy domain. For instance, in the case of a grid with three groups,

two fast and one thermal, if the fast groups have results with standard deviations of 2% and 4% but the thermal group has results with a standard deviation of 20%, it may be due to the fact that the fast groups are substantially more important for that particular score. In this case, if they represent more than 80% of the total score, the code will stop as it considers that convergence has been achieved (each fast group must have converged). This presentation applies to as many groups as there are in the grid. It must be noted that the code does not check that the groups are contiguous. Furthermore, a score of zero with a standard deviation of zero is considered not to have converged: it is, therefore, important not to use a convergence request if the score or standard deviation is expected to be zero (which can occur with void media). The code stops when convergence is achieved, or when the total number of batches is reached (a large batch number must, therefore, be given).

- `PACKET_LENGTH` <n> allows the code to average n batches in a packet before averaging again or sending the value to the collector. This leads to several advantages.
  - Used in parallelism, it considerably reduces the amount of communications and can therefore improve the performance of parallelism runs.
  - For performance purpose, the user should request as big as possible batches. Nevertheless, large batches in criticality calculations lead to a longer source convergence phase. The `PACKET_LENGTH` option allows then both constraints to be satisfied.
  - In some cases of criticality calculations, the estimation of the variance is improved. This is especially the case if there are strong inter-cycle correlations. [39, 43, 93]
- The `EDITION` keyword defines the frequency of output results. The default is every 100 batches.
- The `MESH_EDITION` keyword defines the frequency of mesh output results. This option is useful to prevent too large mesh outputs. The default is the `EDITION` value, otherwise `MESH_EDITION` must be a multiple of `EDITION`.
- The `SUPPRESS_MESH_EDITION_TEXT` keyword allows the user to suppress the edition of mesh results in TRIPOLI-4® listing results file only. The VTK, MED and ASCII files are normally filled. This option is useful to prevent too large TRIPOLI-4® listing results files while keeping other outputs.
- lower energy and upper energy for the simulation of each particle type (**default**: `Einf(neutron)` = $10^{-11}$ MeV, `Esup(neutron)` = 20 MeV, `Einf(photon)` = $10^{-3}$ MeV, `Esup(photon)` = 20 MeV. For electrons and positrons: `Einf` = $10^{-3}$ MeV, `Esup` = 20 MeV). **Warning**: for neutrons, electrons and positrons the kinetic energy must be given;
- for a time dependent problem only : lower time and upper time for the simulation of each particle type (**default**: `Tinf` = 0 s, `Tsup` = $10^{35}$ s for neutrons, photons, electrons and positrons).
- storing of the results of each batch (**default: not stored**);
- biasing of collision if a variance reduction technique is specified (default: no biasing);
- integration of the path (**default: no integration**);

- monitoring: in shielding mode, 500 particles are simulated in the first batch (instead of a number of particles set by SIZE, kept for the following batches) and, in case of variance reduction usage, a fine improvement of the importance map is performed by using statistical information from the simulation of the first 5 batches (**default: monitoring**, see also section 22.2 to deactivate this option);
- for a criticality problem (or a critical parameter search), the number of batches not to be retained (**default: automatic**).
- it is possible to obtain the $K_{ij}$ fission matrix using the KIJ_MATRIX option.
- it is possible to activate photonuclear reactions using the PHOTO_NUCLEAR option. In this case the PHOTONUCLEAR_ROULETTE option of the present command block should also be used in conjunction, so as to lower the default threshold of the Russian roulette for neutrons produced by photonuclear reactions only (in order to deal with very low simulation weights of photoneutrons). The RUSSIAN_ROULETTE keyword can be used together for all other neutrons.
- VOLUMIC_SCORES can be used when all scores are volumic (based on TRACK or COLL estimator) to speed-up the calculation of scores. It may be very efficient when using more than 1000 scores and/or responses. The potential gain can simply be estimated by running two short simulations: one without any scores, and another one with all the needed scores. If they yield very different simulation times, then the keyword VOLUMIC_SCORES may be of some use. This option is now enabled by default. If needed it can be disabled by using the keyword DISABLE_VOLUMIC_SCORES.
- GLOBAL_SCORES can be used to calculate the following responses: Production, Absorption, Leakage, NXN excess, Total flux. Such scores are the sum of these responses on the whole geometry.
- with EPITHERMAL_BROADENING it is possible to activate user-defined models for the Doppler broadening of the elastic scattering kernel in the epithermal range (1 eV - 1 keV): see section 22.13
- the RUSSIAN_ROULETTE option enables to change the default simulation weight threshold of 0.8 (under which the Russian roulette is played). This option is available when no variance reduction command block is present in the input deck. The user may also refer to the Russian roulette control option defined for neutrons only in subsection 17.6.3.
- the PHOTONUCLEAR_ROULETTE option enables to change the default simulation weight threshold of 0.8 (under which the Russian roulette is played) only for neutrons produced by photonuclear reactions.
- the MAX_SPLITTING_PARTICLE_NUMBER option enables to limit the number of split particles to an integer N during each splitting operation. When a TRACK estimator is requested on an extended mesh together with the standard Exponential Transform variance reduction method of TRIPOLI-4®, N has to be set to 1 (see Section 15.8.3). When used together with weight windows, N can be changed by the user, the default value being 1000 (see Section 19.4).
- the ABORT_ON_LEAKAGE option enables to stop the calculation if too many neutrons leak out of the geometry. Once the limit is reached, the calculation is stopped (in parallel mode the limit is to be reached for each simulator).

- the FORCED_UNIT_BASE_INTERPOL option enables to force the unit-base interpolation type on nuclear data for outgoing particle energy sampling. This is equivalent to the choice made by default by the MCNP code.
- the INELASTIC_THRESHOLD_CORRECTION option enables to correct the threshold of neutron discrete inelastic scattering reactions: with this keyword, the threshold taken into account by the code is -QI of the considered inelastic level, instead of the reaction threshold taken from the evaluated data (but kept as default option).
- the ANALOG option enables to perform a fully analog simulation (regarding collisions and transport between collisions). It cannot be used with criticality calculations and only NEUTRON or PHOTON are allowed as particle type for this option (but both cannot be used together since analog photon production induced by neutron interactions is not available yet). Note that in case of electromagnetic shower simulations, photon transport is already analog, hence the ANALOG option does not need to be added.
- the FISSION_MODEL option enables to simulate analog fission by sampling a full fission neutron multiplicity distribution. It allows addressing time nuclear instrumentation applications needing detailed correlations between neutrons of fission chains. Since useful data for this fission sampling are not provided by the standard nuclear data librairies (which provide averaged neutron multiplicities only), this option activates a coupling between TRIPOLI-4® and an external fission model code. Two fission models are available as fission model type: LLNL_FISSION and FIFRELIN.
  The LLNL_FISSION option gives access to the event-by-event FREYA fission model which is directly called by TRIPOLI-4®. Downloading of the LLNL Fission Library package version 2.0.5 (http://nuclear.llnl.gov/simulation/) is required to perform such simulations. Note that for spontaneous fission an external source can be used, that directly calls FREYA (see Section 12.8).
  The FIFRELIN option has to be used with specific event files produced by FIFRELIN code [72] for a given nuclide concerned with induced fission. The FIFRELIN keyword has then to be followed by:
  - the *za* number of this isotope (where za $= 1000 * Z + A$)
  - the USE_FILE keyword, the format number (1 for the n-$\gamma$ uncoupled mode, 2 for the n-$\gamma$ coupled mode without angular data and 3 for the n-$\gamma$ coupled mode with angular data)
  - the number of files (integer *n*)
  - the *n* absolute paths to these files.

  It should be noted that a FIFRELIN event file is valid for a given energy domain only. For spontaneous fission, a specific keyword SPONTANEOUS_FISSION has to be used in the SOURCE command block (see Section 12.12).
  Whatever the fission model is, detailed information on simulated fission neutrons is then available through the ROOT TRACKS options of TRIPOLI-4®: see section 36.3. *It should be noted that for the time being, this coupling has not been fully validated, results are thus not guaranteed and should be handled for R&D purposes only.*
  It can be useful, when using a fission model, to change the upper bound of energy accepted in the fission energy sampling with the keyword FISSION_ENERGY_SUP followed by the appropriate energy value (by default, this upper bound is the upper energy of the simulation).

- Un-weighted and adjoint-weighted kinetics parameters can be computed in criticality calculations: see Sec. 22.6.
- The neutron importance function (the adjoint fundamental mode) can be computed by using the keyword ADJOINT_CRITICALITY: see Sec. 22.7.
- Special simulation modes are available so as to determine some adjustable parameters (albedo boundary conditions and Boron concentration) that make the system exactly critical: see Sec. 22.8.
- The users interested in computing only the prompt fission energy contribution to deposited energy (which is for instance the choice in the HEATR module of the NJOY code [36, 35]) can resort to the following additional optional keywords. With ONLY_PROMPT_ENERGY, delayed beta and delayed gamma contributions are subtracted from the fission Q-value; with ONLY_PROMPT_GAMMA_ENERGY, delayed beta energy is deposited locally, whereas delayed gamma contribution is subtracted from the fission Q-value (i.e. it is neglected). These keywords are mutually exclusive. An alternative treatment of delayed fission photons is possible with the the RESCALE_DELAYED_FISSION_PHOTONS keyword, which activates transport of delayed fission photons under the assumption that their spectrum is the same as the spectrum of prompt photons. The RESCALE_DELAYED_FISSION_PHOTONS and ONLY_PROMPT_GAMMA_ENERGY keywords are mutually exclusive.

  An important remark is due: when the user demands only prompt neutrons to be transported, via the PROMPT_FISSION_ONLY keyword in the SIMULATION block, all delayed contributions to the deposited energy are ignored to preserve coherence. See Sec. 13.7.2 for further details.
- If needed, it is possible to normalise the average number of fission neutrons ($\bar{\nu}$) by a coefficient (typically the $k_{\text{eff}}$ coming from a preliminary criticality calculation). This is achieved by using the keyword RESCALE_NU_BY_KEFF followed by COEFF and the value of the coefficient by which we want to divide $\bar{\nu}$.
- Calculation of the global slowing down current integrated over the geometry is available via the SLOWING_DOWN_CURRENT keyword: see Sec. 22.9 for further details.
- The reactor physics parameters appearing in the six factor formula can be computed in criticality calculation via the SIX_FACTOR_FORMULA keyword: see Sec. 22.10.
- The migration area can be computed in criticality calculations: see Sec. 22.6.
- The Diven dispersion factor can be computed in criticality calculations: see Sec. 22.11.
- If the simulation involves nuclide temperatures that are not available in the neutron nuclear data library, the user has the possibility to request a temperature interpolation between the two nearest temperatures available in the library. A stochastic interpolation of the neutron cross-sections is then performed during the simulation. The related keyword is TEMPERATURE_INTERPOL, followed by the type of interpolation, which can be either LINEAR (for linear interpolation) or SQRT (for square root interpolation). For version 12 of the code, the current usage restrictions are the following: stochastic temperature interpolation cannot be used together with analog transport, perturbations, microscopic multigroup transport (GENDF format), special outputs of cross-sections or nuclei information, SPEED_UP simulation option and adjoint

IFP. It is recalled that integer values must be given by the user when defining the material temperatures (see Chapter 10) and the temperatures for reaction rates on fictive nuclides (see Section 13.2). It should also be noted that the speed up option (see Section 2.2) is deactivated when using the temperature interpolation option (no unionized energy-grid is used).

- The `SUPER_HISTORY` keyword allows to trigger the Super-History method when performing a `CRITICALITY` calculation (see Section 22.17).
- The `WIELANDT_METHOD` keyword allows to trigger the Wielandt simulation method when performing a `CRITICALITY` calculation. This keyword is followed by one of the options `CONSTANT`, `DEFAULT`, `LINEAR` or `USER` (see Section 22.18).
- The particle histories can be stopped at the $k$-th collision by using the keyword `KTH_COLLISION_SOURCE`, with the number of collisions as a free parameter. The syntax is as follows: `KTH_COLLISION_SOURCE NB_COLLISION` <Collision_order>. This option allows computing the partial contribution to the scores up to the $k$-th collision source. Setting <collision_order> can be helpful to compute un-collided, first-collision, second-collision, $\cdots$, scores (flux, reaction rates, etc.) for neutrons and photons.

All the calculation parameter keywords are read by the code in a non-specific order, except for the command indicating the number of batches to be discarded, which, since it is valid only for a criticality calculation (or critical parameter searches), requires prior definition of the type of problem.

## 22.2 SYNTAX

The simulation data syntax is as follows, from keyword `SIMULATION` to keyword `END_SIMULATION` :

```
SIMULATION
        /// Statistics
         BATCH  nb(batches)
         EDITION  frequency
         SIZE  nb(particles per batch)
         CONVERGENCE  precision factor
         PACKET_LENGTH  nb(batches per packet)
         KEEP_RESULT
         RANDOM  Additional data

        /// Particle energy domains
         PARTICLE  nb(types of particles) <TYPE_OF_PARTICLES>
         ENERGY_INF  <TYPE_OF_PARTICLES 1> E_inf(1)
         ENERGY_SUP  <TYPE_OF_PARTICLES 1> E_sup(1)
         TIME_INF  <TYPE_OF_PARTICLES 1> T_inf(1)
         TIME_SUP  <TYPE_OF_PARTICLES 1> T_sup(1)
         ...
         ENERGY_INF  <TYPE_OF_PARTICLES n> E_inf(n)
         ENERGY_SUP  <TYPE_OF_PARTICLES n> E_sup(n)
         TIME_INF  <TYPE_OF_PARTICLES n> T_inf(n)
         TIME_SUP  <TYPE_OF_PARTICLES n> T_sup(n)

        /// Mesh related keyword
         MESH_EDITION  mesh_frequency
         SUPPRESS_MESH_EDITION_TEXT

        /// XML related keyword
         XML_EXPORT
                 XML_NOCOMPLETE
         SERMA_XML_EXPORT

        /// Transport modes and options
#        SHIELDING
#        CRITICALITY
                 DISCARD  nb(batches to be eliminated)
#        FIXED_SOURCES_CRITICALITY
#        MULTIPLE
#        ADJOINT_CRITICALITY  Additional data
#        ALBEDO_SEARCH  Additional data
#        CRITICAL_BORON_SEARCH  Additional data
#        DYNAMIC  Additional data
```

```
                    MONITORING  0
                    TRANSPORT_BIASING
*                           CROSS_SPLIT
*                           QUOTA_SAMPLING
                    COLLISION  1 E_inf(coll)
                    RUSSIAN_ROULETTE  threshold
                    PHOTONUCLEAR_ROULETTE  threshold'
                    MAX_SPLITTING_PARTICLE_NUMBER <N>
                    ABORT_ON_LEAKAGE  limit
                    FORCED_UNIT_BASE_INTERPOL
                    INELASTIC_THRESHOLD_CORRECTION
                    ANALOG  <TYPE_OF_PARTICLES>
                    FISSION_MODEL  <TYPE>
                    FISSION_ENERGY_SUP  E_sup(fiss)
                   WIELANDT_METHOD Additional data
                  SUPER_HISTORY Additional data

                  /// Special scores or transport options
                    DISABLE_VOLUMIC_SCORES
                    GLOBAL_SCORES
                    KIJ_MATRIX
                    KINETIC_PARAMETERS
                    IFP_KINETIC_PARAMETERS  Additional data
                    NAUCHI_KINETIC_PARAMETERS
                    DIVEN_DISPERSION_FACTOR  Additional data
                    PROMPT_FISSION_ONLY
                    ONLY_PROMPT_GAMMA_ENERGY
                    ONLY_PROMPT_ENERGY
                    RESCALE_DELAYED_FISSION_PHOTONS
                    RESCALE_NU_BY_KEFF  Additional data
                    SLOWING_DOWN_CURRENT  Additional data
                    EPITHERMAL_BROADENING  Additional data
                    ALL_COLLIDED_FLUX_PT_POINTS
                    SIX_FACTOR_FORMULA   Additional data
                    TEMPERATURE_INTERPOL  <TYPE>
                    KTH_COLLISION_SOURCE   NB_COLLISION  Collision_order
```

```
                    /// EM Cascade
                    PHOTO_NUCLEAR
                    PHOTON_NEUTRON_RATIO  Additional data
                    ELECTRON_INELASTIC_CUTOFF  Ec
                    ELECTRON_BREMS_CUTOFF  Ec
                    THICK_TARGET_BREMSSTRAHLUNG
                    NO_ELECTRON_MULTIPLE_SCATTERING
                    ELECTRON_TOTAL_STOPPING_POWER
                    ELECTRON_PHOTON_BALANCE
                    NO_ANOMALOUS_SCATTERING_FACTOR
                    ELECTRON_MIN_STEP_LENGTH  step

                    /// particle event files
                    STORE_SOURCES_IN_FILE
                    STORE_COLLISIONS_IN_FILE
                    STORE_LEAKAGE_IN_FILE

                    /// Debug features
#                   HOLE_COUNT
#                   HOLE_MAP
#                   HOLE_VOLU_MAP
                          LAMBDA_HOLE  lambda_hole
          END_SIMULATION
```

Monitoring is enabled by default in shielding mode, which automatically results in simulating 500 particles in the first batch of a shielding calculation. In this case, the total number of simulated particles is `nb(batches)-1` multiplied by `nb(particles per batch)` plus 500. To deactivate the monitoring option, see at the end of this section.
`<TYPE_OF_PARTICLES>` may have the following values:

- `NEUTRON` and/or `PHOTON` and/or `ELECTRON` and `POSITRON`, depending on the value of `nb(types of particles)`, which can range from 1 to 4.
  To activate the `PHOTO_NUCLEAR` option, at least `PHOTON` and `NEUTRON` types must be present.
  Photons, electrons and positrons have to be simulated together.

If a criticality calculation is chosen (introduced using the keyword `CRITICALITY`):

- the entire neutron energy domain is necessary (from 1.E-11 to 20. MeV);
- it is possible to eliminate a certain number of batches, the `nb` batches to be discarded, to achieve equilibrium and eliminate the transient regime. This is introduced using the keyword `DISCARD`. By default, the code automatically determines the number of batches to discard.

If $k_{eff}$ is close to 1, the FIXED_SOURCES_CRITICALITY mode is not appropriate. In this case all single source neutron progeny is simulated in each batch, therefore the calculation time may be very long, or even possibly infinite if the $k_{eff}$ is greater than or equal to 1. Good practices with $k_{eff}$ is close to criticality:

- make a test of simulation within a CRITICALITY mode to assess the value of $k_{eff}$;
- if $k_{eff}$ is confirmed close to 1 and still a FIXED_SOURCES_CRITICALITY run is needed, considerably decrease the size of batch (divide by $\frac{k}{1-k}$ the size of batch for example);

Note: In FIXED_SOURCES_CRITICALITY mode the $k_{eff}$ result in TRIPOLI-4® output is NOT a real $k_{eff}$ but rather a multiplication factor: it is the average number of the overall generation of any fission neutron ($k_{eff} = \frac{\rho}{1-\rho}$).

The keyword DYNAMIC is used to activate time-dependent (kinetic) simulations, including neutrons and delayed neutron precursors. See Sec. 22.12 for more details and a thorough description of the associated syntax and options.

The keyword SHIELDING is optional in a radiation shielding problem (as it is the default option).
The monitoring option can be deactivated with MONITORING 0. In this case, the number of particles of the first batch is set by SIZE and no improvement of the importance map is sought by the code, particularly the downscattering of particles towards lower energy groups is not favoured. Deactivating this option can be useful when using the AMS variance reduction technique (see Chapter 18). Otherwise, it is intended for skilled users only (it is adviced to keep the default monitoring option).
COLLISION 1 Einf(coll) means that the collision is biased at energies higher than Einf(coll). Currently, collision biasing is only possible in neutron problems. It is not possible for combined neutron-gamma problems. This option is also intended for skilled users (the default, no collision biasing, is typically used).
The keyword TRANSPORT_BIASING is optional and allows the user to choose between two options for the transport when a variance reduction technique is used. When biasing is used, the default option for the transport is QUOTA_SAMPLING. The keyword CROSS_SPLIT is used to treat the transport between two collisions with sampling performed volume by volume (i.e. the particle is stopped each time it crosses a surface, and the distance to next collision sampled again). This option is implemented for neutrons only, and is more costly in terms of simulation time than the default option QUOTA_SAMPLING.

## 22.3 OPTION KEEP_RESULT

Long calculations can be broken up into smaller pieces. First $N_1$ batches are calculated with the option KEEP_RESULT.
In this case, if the data file is **tripoli4.data**, a file named **tripoli4.data.RESU** is created containing the results of each batch.
If the calculation is resumed (once again using the option KEEP_RESULT) with $N_2$ batches, without deleting the file named **tripoli4.data.RESU**, the calculation continues from the last batch. The results are obtained by taking into account the $N_1$ previous batches; therefore, for $N_2$ ($> N_1$) batches. (Warning: this is a change

from versions of the code earlier to tripoli4-version5.)

Important: in this case, the simulation results file must be given a name other than **tripoli4.data.RESU** (this name is given at the execution call).

This option is available in parallel mode starting with version 4.5 of the code. For parallel runs, dump files will be generated for the collector process as well as for each simulator process using a name in the following form: **tripoli4.data.**< $n$ >**th_task.RESU**.
These dump files are not compatible with versions of the code before version 4.5.

## 22.4 OPTION XML_EXPORT

The keyword XML_EXPORT is used to request that the code generates a XML results file during the simulation. TRIPOLI-4® produces then a file named <data_file>.xml. This file can then be processed by any XML parser (rather SAX than DOM). It provides a structured and standard access to the values computed by the code. Results are recorded every batch by default, but if additional keyword XML_NOCOMPLETE is given, only mean results are recorded at editions.
SERMA_XML_EXPORT produces another type of XML output. It provides only mean results.

## 22.5 OPTION RANDOM

By default, the TRIPOLI-4® code handles management of the first random number. The user can choose another random number using RANDOM. The additional data have a syntax that depends on the implementation of the random number generator. This syntax is used when printing out the properties of the random number generator for each batch (it is the last line of the results file).
The three available random generators are the drand48, the GFSR, and the Mersenne-Twister 19937.
The drand48 is the unix standard random generator (see manpage). This generator is only used in monoprocessor mode. Its period is $2^{48}$. The seed can be changed using the following statement:

```
RANDOM DRAND48_RANDOM 4125 45310 47934
```

The GFSR random generator is the historical random generator for the parallel mode in TRIPOLI-4® [94]. Its seed can not be changed. Its period is about $2^{607}$. The generator can be called in parallel mode using the syntax:

```
RANDOM GFSR_RANDOM
```

The Mersenne-Twister 19937 is a well-known random generator [75]. It can be used both in parallel and in monoprocessor mode. Its period is $2^{19937}-1$. Since version 10, it becomes the default generator in parallel. It can be seeded with the following statement:

```
RANDOM MT19937_RANDOM SEED 8191
```

## 22.6 COMPUTING THE KINETICS PARAMETERS

Knowledge of kinetics parameters is key to the study of nuclear reactor dynamics, in connection with transient behavior due to normal operation or accidents. Usually, kinetics parameters stem from integrating the full

time-dependent Boltzmann and precursors equations over the space, energy and angle variables: after various algebraic manipulations, they take the general form of a ratio of dot products, namely, $\langle \psi, F\varphi \rangle / \langle \psi, G\varphi \rangle$, where $F$ and $G$ are operators acting on the neutron flux $\varphi$, and $\psi$ is a weighting function depending on specific choices associated to the system under analysis [98]. Essential to reactor control are the mean generation time $\Lambda$ of a neutron within a reactor, which is expressed as

$$\Lambda = \frac{\langle \psi, \frac{1}{v}\varphi \rangle}{\langle \psi, P\varphi \rangle}, \tag{22.1}$$

where $P$ is the total fission operator (including prompt and delayed productions) and $v$ is the neutron speed; and the delayed neutron fraction $\beta$, which is expressed as

$$\beta = \frac{\langle \psi, B\varphi \rangle}{\langle \psi, P\varphi \rangle}, \tag{22.2}$$

where $B$ is the fission operator corresponding to delayed neutrons alone. Most often, the weighting function $\psi$ is either taken to be $\psi = 1$, in which case the kinetics parameters are said to be un-weighted, i.e.,

$$\Lambda_0 = \frac{\langle 1, \frac{1}{v}\varphi \rangle}{\langle 1, P\varphi \rangle}, \quad \beta_0 = \frac{\langle 1, B\varphi \rangle}{\langle 1, P\varphi \rangle} \tag{22.3}$$

or $\psi = \varphi^\dagger$ ($\varphi^\dagger$ denoting the adjoint neutron flux), in which case the kinetics parameters are said to be adjoint-weighted, or effective, namely,

$$\Lambda_{eff} = \frac{\langle \varphi^\dagger, \frac{1}{v}\varphi \rangle}{\langle \varphi^\dagger, P\varphi \rangle}, \quad \beta_{eff} = \frac{\langle \varphi^\dagger, B\varphi \rangle}{\langle \varphi^\dagger, P\varphi \rangle}. \tag{22.4}$$

The un-weighted neutron removal time $T_0$ is related to the un-weighted generation time by $\Lambda_0 = T_0/k_{eff}$. The units of kinetics parameters are coherent with those used in the code. The effective mean generation time $\Lambda_{eff}$ is expressed in [s], and these units apply to all kinetics parameters homogeneous to a time (for instance the removal time, the leakage time, the absorption time, etc.). The effective delayed neutron fraction $\beta_{eff}$ is given in (dimensionless) units of [$10^5$ pcm], and these dimensionless units apply also to the un-weighted delayed neutron fraction $\beta_0$. In criticality calculations, it is possible to compute adjoint-weighted and un-weighted kinetics parameters. The specific syntax is the following.

```
SIMULATION
        ...
        KINETIC_PARAMETERS
        ...
        NAUCHI_KINETIC_PARAMETERS
        ...
        IFP_KINETIC_PARAMETERS
        IFP_CYCLE_LENGTH
                <NB_CYCLES>
        <CONVERGENCE_STATISTICS>
        <NUCLEI_CONTRIBUTIONS>
                LIST
                <NB_NUCLEI> NUC_1 ··· NUC_NB_NUCLEI
        or
        <NUCLEI_CONTRIBUTIONS>
                ALL
        ...
END_SIMULATION
```

Un-weighted kinetics parameters are activated by resorting to the keyword `KINETIC_PARAMETERS`. This option yields the average delayed neutron fraction $\beta_0$, the generation time $\Lambda_0$, the removal time $T_0$, the average time to leakage and the average time to absorption.

Adjoint-weighted kinetics parameters can be computed by resorting to either exact or approximated methods. The IFP method allows estimating the adjoint neutron flux in criticality simulations by resorting to the importance approximation [98]: this yields an exact weighting factor for kinetics parameters, provided that the neutron importance is measured after a sufficiently large number of cycles. The IFP ponderation is activated by the keyword `IFP_KINETIC_PARAMETERS`, followed by `IFP_CYCLE_LENGTH` and the number of cycles for the estimation of the asymptotic neutron importance. A suggested value for `IFP_CYCLE_LENGTH` is 20. Adjoint-weighted parameters include the effective delayed neutron fraction $\beta_{eff}$, the effective generation time $\Lambda_{eff}$, and their ratio $\alpha_{Rossi} = -\beta_{eff}/\Lambda_{eff}$, which is called Rossi alpha or alpha value at delayed criticality. The Rossi alpha is expressed in units of the inverse of time, i.e., $[s^{-1}]$.

As an option, the keyword `IFP_CONVERGENCE_STATISTICS` displays the convergence of the IFP kinetics parameters as a function of the number of IFP cycles, from 1 to the maximum value defined by the keyword `IFP_CYCLE_LENGTH`. See [98] for more details.

Until very recently, the exact weighting method by the adjoint neutron flux was not available in Monte Carlo codes, and approximated methods were available to estimate the adjoint-weighted kinetics parameters. Among these techniques, one of the the most successful is the so-called Nauchi's method, which can now

be seen as a first-order approximation of the IFP ponderation [87]. The Nauchi's kinetics parameters can be computed by resorting to the keyword NAUCHI_KINETIC_PARAMETERS (the keyword BETA_EFFECTIVE works also, for retro-compatibility with previous versions of the code). The option NAUCHI_KINETIC_PARAMETERS will yield the effective delayed neutron fraction $\beta_{eff}$ and the effective generation time $\Lambda_{eff}$ weighted by the first-order importance function (this corresponds by definition to the use of the IFP method with IFP_CYCLE_LENGTH equal to 1). Moreover, the option NAUCHI_KINETIC_PARAMETERS will also provide estimates of the kinetics parameters by the so-called prompt method in a single criticality simulation.

The three keywords for kinetics parameters, namely, KINETIC_PARAMETERS, IFP_KINETIC_PARAMETERS and NAUCHI_KINETIC_PARAMETERS, are compatible with each other and can be used together in a criticality simulation.
Observe that in principle it would be also possible to use the PROMPT_FISSION_ONLY keyword to run a criticality simulation without delayed neutrons. In this case, the delayed neutron fraction could be roughly estimated by simply taking the difference between the $\beta_{eff}$ value with and without delayed neutrons. Observe however that, when the PROMPT_FISSION_ONLY keyword is activated, the behaviour of the Kstep and Kcoll estimators for Keff is as expected, i.e., delayed neutron contributions are ignored. As for the Ktrack estimator, we use the total number of neutrons ($\nu_t$) and we attribute to those neutrons the prompt energy spectrum.
See also section 13.11 for more details about the relation between $\beta_0$ calculations and prompt/delayed fission spectra.

In addition to global parameters such as $\beta_{eff}$ and $\Lambda_{eff}$, the kinetics parameters pertaining to each precursor family $j$ are also provided. The number of families depends on the specific nuclear data library. When computing un-weighted kinetics parameters with the keyword KINETIC_PARAMETERS, these include the precursor decay constants $\lambda_j$ (in units of $[s^{-1}]$) and the natural proportions of delayed neutrons $\beta_j$ for each family (in units of $[10^5$ pcm$]$). The quantities $\beta_j$ satisfy the relation

$$\sum_j \beta_j = \beta_0. \tag{22.5}$$

Moreover, the $\beta_j$-weighted average decay constant

$$\bar{\lambda} = \frac{\beta_0}{\sum_j \frac{\beta_j}{\lambda_j}} \tag{22.6}$$

is also computed. The quantity $\bar{\lambda}$ is also given in units of $[s^{-1}]$. When computing adjoint-weighted kinetics parameters with the keyword IFP_KINETIC_PARAMETERS, the code provides the effective delayed neutron fractions $\beta_{eff,j}$ for each precursor family (in units of $[10^5$ pcm$]$). The quantities $\beta_{eff,j}$ satisfy the relation

$$\sum_j \beta_{eff,j} = \beta_{eff}. \tag{22.7}$$

Finally, for some applications it is interesting to further decompose the kinetics parameters over the contributions of each fissile isotope. This can be achieved either by using the optional keyword NUCLEI_CONTRIBUTIONS followed by LIST, plus the number of isotopes for which the decomposition

is sought, and the list of the names of the chosen isotopes; or by using NUCLEI_CONTRIBUTIONS followed by ALL in the case where the user wants to analyze the contributions of all fissile isotopes that are present in the simulation. The optional keyword NUCLEI_CONTRIBUTIONS applies to both adjoint-weighted and un-weighted kinetics parameters, if requested.

The keyword KINETIC_PARAMETERS activates also the calculation of the migration area, which is a useful parameter in reactor physics. Actually, the code computes the average square distance $\langle r^2 \rangle$ that a neutron travels within a single fission generation, from birth to disappearance, weighted by the forward critical flux. The displayed quantity is thus the average square distance, in units of [cm$^2$]. The migration area can be then estimated by dividing the obtained score by a factor of 6, as suggested in the framework of the diffusion theory. When using the IFP_KINETIC_PARAMETERS keyword, the computed average square distance is also weighted by the adjoint critical flux.

Note: the migration area takes into account the *true* boundary conditions chosen by the user. For instance, if the user specifies reflecting boundary conditions on a given surface, the particles will be reflected by this boundary and the migration area will correspondingly take into account the presence of a reflection. Therefore, the use of reflection to mimic an infinite medium cannot be properly taken into account in the migration area calculation.

## 22.7 NEUTRON IMPORTANCE (CRITICAL ADJOINT FLUX)

The critical adjoint (angular) neutron flux $\varphi^\dagger(r_0, \Omega_0, E_0)$, as a function of position $r_0$, energy $E_0$ and direction $\Omega_0$, physically represents the asymptotic importance of a neutron injected into a multiplying system with coordinates $r_0$, $E_0$ and $\Omega_0$. The formal identification between the neutron importance and the adjoint flux lies at the basis of the so-called Iterated Fission Probability method [85]. In order to explicitly compute the importance function, in the ADJOINT_CRITICALITY simulation mode the code runs an ensemble of $M$ fixed-source batches over $L$ fission generations. For each batch, neutrons start with coordinates $r_0$, $E_0$ and $\Omega_0$. The quantity $L$ defines the IFP cycle length.

More generally, one is possibly led to computing scalar products of the kind $\langle \varphi^\dagger, S \rangle$, where $S$ is an arbitrary user-defined source. The scalar products are then decomposed on a spatial, energetic and angular user-defined mesh with respect to the initial coordinates of the neutrons. In particular, by taking a delta-like source $S = \delta(r - r_0)\delta(\Omega - \Omega_0)\delta(E - E_0)$ at a given point of the phase space, the scalar product precisely defines the adjoint flux $\varphi^\dagger(r_0, \Omega_0, E_0)$.

The critical adjoint flux calculation is activated by the keyword ADJOINT_CRITICALITY, with the following syntax:

```
SIMULATION
    ...
    ADJOINT_CRITICALITY
    DISCARD 0
    ...
END_SIMULATION
```

Since the adjoint flux calculation is formally of the fixed source type, DISCARD must be set to zero. The only allowed particle type for the simulation is NEUTRON. The scalar product $\langle \varphi^{\dagger}, S \rangle$ corresponding to the source $S$ defined by the user (by using the customary source block; see Chapter 12) is then projected over user-defined meshes by using special scores designed for the adjoint calculations. An arbitrary number of such ad-hoc scores are introduced by the block:

```
IFP_SCORES
    <IFP_SCORE_TYPE_1>
    NAME <SCORE_NAME_1>
    IFP_CYCLE_LENGTH <CYCLE_LENGTH_1>
    <SCORING_DATA_1>
    ENERGY DECOUPAGE <ENERGY_DECOUPAGE_NAME_1>
    <OPTIONAL_KEYWORDS_1>
    ...
    <IFP_SCORE_TYPE_M>
    NAME <SCORE_NAME_M>
    IFP_CYCLE_LENGTH <CYCLE_LENGTH_M>
    <SCORING_DATA_M>
    ENERGY DECOUPAGE <ENERGY_DECOUPAGE_NAME_M>
    <OPTIONAL_KEYWORDS_M>
END_IFP_SCORES
```

Two kinds of adjoint scores can be computed, either adjoint fluxes or adjoint reaction rates: correspondingly, the keyword IFP_SCORE_TYPE can be set to be either IFP_ADJOINT_FLUX for the adjoint flux $\langle \varphi^{\dagger}, S \rangle$, or IFP_REACTION_RATE for $\langle \varphi^{\dagger}, \Sigma_t S \rangle$, where $\Sigma_t$ is the total cross section having the coordinates of the neutron at emission. This latter quantity sometimes appears in reactor physics calculations.

The user must then define the name of the score via the keyword NAME and the length of the IFP cycle for each score via the keyword IFP_CYCLE_LENGTH. The recommended value is 20.

In order to properly decompose the scalar product between the source and the adjojnt flux, the user must ensure that the score mesh covers the emission space of the source, . For instance, in order to compute $\varphi^{\dagger}(r_0, \Omega_0, E_0)$ over a given spatial range $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$ and $[z_{\min}, z_{\max}]$, an angular range $[\theta_{\min}, \theta_{\max}]$ and $[\phi_{\min}, \phi_{\max}]$ and an energy range $[E_{\min}, E_{\max}]$, the user must take a spatially uniform source,

isotropic in angle and uniform in energy over the same domain. In this case, the adjoint flux decomposed over the grid will precisely define $\varphi^{\dagger}(r_0, \Omega_0, E_0)$ for each phase space point $r_0$, $E_0$ and $\Omega_0$, integrated over each cell of the mesh. For very fine meshes, we can thus recover $\varphi^{\dagger}(r_0, \Omega_0, E_0)$.

The block SCORING_DATA is used in order to define the spatial and angular meshes over which the adjoint scores are to be projected. This can be defined by using:

- CARTESIAN, in which case the user provides a Cartesian mesh in space. The keyword is followed by the coordinates and the number of meshes for each coordinate, in the form:
  $X_{\min}$ $X_{\max}$ nb_mesh_X $Y_{\min}$ $Y_{\max}$ nb_mesh_Y $Z_{\min}$ $Z_{\max}$ nb_mesh_Z
  It is implicitly assumed that the scores are integrated with respect to the emission angles, i.e., we score $\varphi^{\dagger}(r_0, E_0)$;

- CARTESIAN_ANGULAR, in which case the user provides a Cartesian mesh in space and an angular grid for $\phi$ and $\theta$. The keyword is followed by the coordinates and the number of meshes for each coordinate, in the form:
  $X_{\min}$ $X_{\max}$ nb_mesh_X $Y_{\min}$ $Y_{\max}$ nb_mesh_Y $Z_{\min}$ $Z_{\max}$ nb_mesh_Z
  $\phi_{\min}$ $\phi_{\max}$ nb_mesh_$\phi$ $\theta_{\min}$ $\theta_{\max}$ nb_mesh_$\theta$

- VOLUMES, in which case the user provides the list of volumes over which the adjoint flux is decomposed. It is implicitly assumed that the scores are integrated with respect to the emission angles and over each volume of the list.

Finally, the user must provide the energy mesh over which the adjoint score is computed, by using the keywords ENERGY GRID, followed by the name of the energy mesh (which must be declared in the usual mesh block).

IMPORTANT. In the ADJOINT_CRITICALITY simulation mode, all other kinds of scores are forbidden.

Similarly as in the case of the forward critical flux that is computed in regular criticality calculations, the adjoint critical flux is defined up to a normalization constant. The optional keyword NORMALIZE can be used in order to normalize each adjoint score to unit.

In addition, the user can define the optional keyword ENTROPY to analyze the convergence of the IFP scores by computing the relative entropy with respect to the score that has the highest IFP cycle length [85].

An example of adjoint scores is provided here:

```
IFP_SCORES
IFP_ADJOINT_FLUX
NAME FluxAdj
IFP_CYCLE_LENGTH 20
CARTESIAN -6 +6 10 -3 +3 10 -5 +5 10
```

```
ENERGY GRID DEC_281G
NORMALIZE

IFP_REACTION_RATE
NAME ReactionAdj
IFP_CYCLE_LENGTH 20
CARTESIAN -6 +6 10 -3 +3 10 -5 +5 10
ENERGY GRID DEC_33G
END_IFP_SCORES
```

## 22.8 CRITICAL PARAMETER SEARCH

When the fundamental $k$-eigenvalue depends on a single free control parameter $p$, the $k$-eigenvalue equation can be solved for the fundamental eigenvalue $k = k(p)$ by power iteration for a given $p$. We can then search the value $p = p_c$ that makes the system exactly critical, with $k(p) = k(p_c) = 1$ [26].

### 22.8.1 ALBEDO SEARCH

The critical albedo search consists in imposing albedo conditions on the boundaries (or part of the boundaries) of the system, and determining the albedo value $\beta$ (if any) that makes the reactor critical (i.e., $k_{\text{eff}} = 1$). Such problems emerge for instance in connection with Monte Carlo depletion calculations at the lattice scale, as a way of reproducing whole-core effects at the boundaries of the assembly. The albedo search simulation mode is activated with the keyword `ALBEDO_SEARCH`.

For each of the boundary surfaces where albedo is to be adjusted, the user must apply an `ALBEDO` boundary condition (see Sec. 9.1.3) and set the initial albedo value $\beta_0^j$ for the search ($j$ being the index of the surface), with a minus sign. Surfaces having an albedo boundary condition with a positive albedo value will not be adjusted during the critical albedo search: this allows providing adjustable and non-adjustable albedo conditions in the same geometry, for the sake of generality. During the albedo search procedure, the albedo $\beta_j$ of each adjustable surface (i.e., those whose initial value was set with a minus sign) will be updated as

$$\beta_j = \beta \times |\beta_0^j|, \tag{22.8}$$

where $\beta$ is the global multiplier that is sought by the code by using the update law

$$\beta^{(g+1)} = \frac{\beta^{(g)}}{\left(k^{(g)}\right)^{\gamma}}, \tag{22.9}$$

where $g$ is the index of the current batch (generation) of the power iteration, and $k^{(g)}$ is the dominant $k$-eigenvalue determined at the current batch based on the global albedo value $\beta^{(g)}$. The exponent $0 < \gamma \leq 1$ is used in order to adjust the speed of convergence of the search. The suggested value is $\gamma = 0.5$.

The code will provide the global albedo value $\beta$ and the resulting $k$-eigenvalue, which should be very close to 1. The actual albedo value of each surface $j$ is then determined by multiplying the obtained global value $\beta$

by the absolute value of the initial $\beta_0^j$ that he user has assigned to each adjustable boundary condition.

The syntax for the albedo search is given as follows:

```
SIMULATION
        ...
        ALBEDO_SEARCH
        INITIAL_GUESS
                <VAL>
        POWER
                <VAL>
        <LOWER_LIMIT>
                <VAL>
        <UPPER_LIMIT>
                <VAL>
        ...
END_SIMULATION
```

The initial value for the global albedo multiplier $\beta^{(0)}$ that is used for the iterative search is introduced by the keyword `INITIAL_GUESS` and must be positive. The `POWER` keyword is used to set the exponent $\gamma$. Two optional keywords `UPPER_LIMIT` and `LOWER_LIMIT` can be used in order to restrict the search within a given interval: if the value $\beta^{(g+1)}$ at a given generation lies beyond the limits, the search is aborted. In this case the initial value for the search must be also chosen within the bounds. By default, the lower limit is set to zero and the upper limit is set to infinity.

Similarly as in criticality calculations, a number of batches to be discarded must be assigned by using the keyword `DISCARD`.

### 22.8.2 BORON SEARCH

The critical Boron search simulation mode aims at finding the Boron concentrations in one or more user-defined material compositions such that the reactor is critical (i.e., $k_{\text{eff}} = 1$). The Boron concentration search is activated with the keyword `CRITICAL_BORON_SEARCH`.

For each of the material compositions where the Boron concentration is to be adjusted, the user must introduce the additional keyword `ADJUSTABLE_BORON` before the temperature value. This keyword is used in order to specify that the concentration of the $^{10}$B and/or $^{11}$B nuclides contained in the material composition must be adjusted. An example is provide here:

```
COMPOSITION <NB_COMPOSITIONS>
    ...
```

```
POINTWISE ADJUSTABLE_BORON <TEMP> COMPOSITION_NAME <NB_NUCLEI>
NUC_1     <NUC_1_CONC>
...            ...
NUC_M    <NUC_M_CONC>
B10         <INITIAL_B10_CONC>
B11         <INITIAL_B11_CONC>
...
END_COMPOSITION
```

Material compositions containing $^{10}$B and/or $^{11}$B nuclides and not displaying the keyword ADJUSTABLE_BORON will not be adjusted during the critical Boron search: this allows providing adjustable and non-adjustable material compositions in the same geometry, for the sake of generality. The initial (guess) concentrations $c_0^j > 0$ for $^{10}$B and/or $^{11}$B nuclides must be provided by the user. During the Boron search procedure, the $^{10}$B and/or $^{11}$B nuclide concentration $c_j$ of each adjustable material composition $j$ will be updated as

$$c_j = c \times c_0^j, \tag{22.10}$$

where $c$ is the (dimensionless) global multiplier that is sought by the code by using the update law

$$c^{(g+1)} = c^{(g)} \times k^{(g)}, \tag{22.11}$$

where $g$ is the index of the current batch (generation) of the power iteration, and $k^{(g)}$ is the dominant $k$-eigenvalue determined at the current batch based on the global Boron concentration $c^{(g)}$. Observe that the initial ratio between $^{10}$B and $^{11}$B nuclide concentrations will be exactly preserved during the iterations.

The code will provide the global Boron concentration multiplier $c$ and the resulting $k$-eigenvalue, which should be very close to 1. The actual concentration of the $^{10}$B and/or $^{11}$B nuclides in material composition $j$ must be then determined by multiplying the obtained global value $c$ by the initial concentrations $c_0^j$ that the user has assigned.

The syntax for the Boron search is given as follows:

**SIMULATION**
...
**CRITICAL_BORON_SEARCH**
**INITIAL_GUESS**
<VAL>
<**LOWER_LIMIT**>
<VAL>
<**UPPER_LIMIT**>
<VAL>
...
**END_SIMULATION**

The initial value for the global concentration multiplier $c^{(0)}$ that is used for the iterative search is introduced by the keyword `INITIAL_GUESS` and must be positive. Two optional keywords `UPPER_LIMIT` and `LOWER_LIMIT` can be used in order to restrict the search within a given interval: if the value $c^{(g+1)}$ at a given generation lies beyond the limits, the search is aborted. In this case the initial value for the search must be also chosen within the bounds. By default, the lower limit is set to zero and the upper limit is set to infinity.

Similarly as in criticality calculations, a number of batches to be discarded must be assigned by using the keyword `DISCARD`.

## 22.9 SLOWING DOWN CURRENT

The slowing down current is an important parameter in neutron transport problems, yielding a global estimate of the 'hardness' of the energy spectrum. The slowing down current exceeding a given energy $E_t$ can be computed by resorting to the keyword `SLOWING_DOWN_CURRENT`, followed by `ENERGY` and the threshold energy value $E_t$, expressed in units of MeV. This reponse will automatically provide as an output three components, i.e., the slowing down current (taking into account only neutrons traversing the threshold energy from higher to lower energies), the upscattering current (taking into account only neutrons traversing the threshold energy from lower to higher energies), and finally the net slowing down current (the net balance of the two previous currents, with the positive sign for slowing down and the negative sign for upscattering).

## 22.10 SIX FACTOR FORMULA

In criticality calculations, it is possible to decompose the $k_{eff}$ into six factors accounting for separate effects analysis:

- $\chi_{n,xn}$ for the excess weight due to $(n,xn)$ reactions;
- $\varepsilon_{even}$ for fast fission on even actinides ($^{236}U$, $^{238}U$, $^{238}Pu$, $^{240}Pu$, etc.);
- $\varepsilon_{odd}$ for fast fission on odd actinides ($^{235}U$, $^{239}Pu$, $^{241}Pu$, $^{241}Am$, etc.);
- $p$ for fhe resonance escape probability;
- $f$ for the thermal utilization factor;
- $\eta$ for the thermal fission factor.

In order to account for possible leakage, we have introduced a seventh factor $P_{NL}$ expressing the probability that a neutron will not leak from the multiplying system. The multiplication factor is then expressed as the product

$$k_{eff} = \chi_{n,xn}\varepsilon_{even}\varepsilon_{odd}pf\eta P_{NL}. \tag{22.12}$$

The terms introduced above are defined by setting two energy groups, say 2 for thermal and 1 for fast, with a boundary at $E_0$. Typically, $E_0 = 0.625 \times 10^{-6}$ [MeV]. Let us set $S$ for the fission source, $P_i$ for the production

terms in group $i$, with $P = P_1 + P_2$, $A_i$ for the absorption terms in group $i$, with $A = A_1 + A_2$, and $F_i$ for the fission terms in group $i$, with $F = F_1 + F_2$. Furthermore, denote by $L$ the leakage term. Then we have

$$\varepsilon_{even} = \frac{P}{A}\frac{A - F_{1,even}}{P_{1,odd} + P2}, \tag{22.13}$$

$$\varepsilon_{odd} = \frac{P_{1,odd} + P2}{A - F_{1,even}}\frac{A - F_1}{P_2}, \tag{22.14}$$

$$p = \frac{A_2}{A - F_1}, \tag{22.15}$$

$$f = \frac{A_{2,fuel}}{A_2}, \tag{22.16}$$

$$\eta = \frac{P_2}{A_{2,fuel}}, \tag{22.17}$$

$$P_{NL} = \frac{A}{A + L}, \tag{22.18}$$

$$\chi_{n,xn} = \frac{A + L}{S}, \tag{22.19}$$

whence

$$k_{eff} = \chi_{n,xn}\varepsilon_{even}\varepsilon_{odd}pf\eta P_{NL} = \frac{P}{S}. \tag{22.20}$$

The six factor formula can be computed by resorting to the keyword `SIX_FACTOR_FORMULA` , followed by `ENERGY` and the threshold energy value $E_0$ between thermal and fast groups, expressed in units of MeV.

## 22.11 DIVEN DISPERSION FACTOR

In reactor physics, the Diven dispersion factor $D$ expresses the dispersion of the number of prompt neutrons at each fission event and is typically used in order to calibrate the experimental measurements of the kinetics parameters based on the reactor noise theory. It is usually defined as

$$D = \frac{\langle \nu_p(\nu_p - 1)\rangle}{\langle \nu_p\rangle^2} = D_\nu D_s, \tag{22.21}$$

where $D_\nu$ expresses the contribution purely due to fluctuations in the prompt neutron number (which would correspond to the point-kinetics approximation) and $D_s$ is a spatial correction factor that takes into account the shape of the fundamental and adjoint mode within the core. Both coefficients are computed by the code. The adjoint-weighting procedure is based on the IFP method and the Diven factor calculation is available in `CRITICALITY` simulation mode only. The specific syntax is the following.

**SIMULATION**
...
**DIVEN_DISPERSION_FACTOR**
**IFP_CYCLE_LENGTH**
<NB_CYCLES>
**FILE**
Filename
<**CONVERGENCE_STATISTICS**>
...
**END_SIMULATION**

The keyword IFP_CYCLE_LENGTH is used to provide the number of cycles for the estimation of the asymptotic neutron importance. A suggested value for IFP_CYCLE_LENGTH is 20.

The definition of the Diven factor implies the use of the probability $P(\nu_p)$ of obtaining $\nu_p$ prompt neutrons at fission events, for each fissile nucleus. Since nuclear data libraries typically provide only the average value $\langle\nu_p\rangle$, the calculation of the Diven factor requires additional information from the user. The distribution $P(\nu_p)$ can be provided by the user by using the keyword FILE, followed by the file name. (See also Section 3.1.2 for the AdditionalData/P_NU_DATA directory provided with the code). Probability data in the file must be organized by isotope and energy intervals. Values of $P(\nu_p)$ are given from $P(\nu_p = 0)$ to $P(\nu_p = \nu_{p,\max})$. The structure of the distribution for each fissile nuclide must be the following:

```
NUCLEUS <NUC_NAME_1>

ENERGY <E0> <E1>
0  P(0)
...
nu_max_nuc_1_E_1  P(nu_max_nuc_1_E_1)

ENERGY <E1> <E2>
0  P(0)
...
nu_max_nuc_1_E_2  P(nu_max_nuc_1_E_2)

...

NUCLEUS <NUC_NAME_M>

ENERGY <E0> <E1>
0  P(0)
...
```

```
nu_max_nuc_M_E_1  P(nu_max_nuc_M_E_1)

ENERGY <E1> <E2>
0  P(0)
...
nu_max_nuc_M_E_2  P(nu_max_nuc_M_E_2)

...
```

where the distribution values are given for each energy interval defined by the user. The distribution must be provided for each fissile nucleus that is present in the simulation by writing the corresponding sections consecutively in the data file.
As an option, the keyword `IFP_CONVERGENCE_STATISTICS` displays the convergence of the Diven factor as a function of the number of IFP cycles, from 1 to the maximum value defined by the keyword `IFP_CYCLE_LENGTH`.

## 22.12 `DYNAMIC` (TIME-DEPENDENT SIMULATION) OPTIONS

Time-dependent neutron transport, taking into account the presence of the delayed neutron precursors, is activated using the keyword `DYNAMIC` in the simulation block. The goal of kinetic Monte Carlo methods is to address the solution of the full time-dependent Boltzmann equation, coupled to the $M$ precursor equations, one for each delayed neutron precursor family. The number of families depends on the nuclear data library: for instance, in ENDF/B-VIII (and previous versions) there exist 6 families, and each fissile isotope has its own set of decay constants $\lambda_j$, $j = 1,\ldots,6$; in JEFF3.1.1, on the contrary, the number of families has been increased to 8 and all fissile isotopes have the same set of decay constants $\lambda_j$, $j = 1,\ldots,8$. When running a time-dependent simulation with neutrons and precursors, a few issues must be taken into account:

- The characteristic time scale of the precursor population (i.e., the average decay time $\sim 10$ s) is much longer than that of neutrons (i.e., the mean generation time, $10^{-5}$ s in a typical PWR), and the delayed neutron fraction is very small ($\sim 0.7\%$). This implies that at the critical regime, when the neutron and precursor populations are in equilibrium, precursors are considerably more abundant than neutrons within the core, by a factor $\sim 10^4$. This makes the use of special variance-reduction techniques mandatory to deal with the presence of two very different time scales in the simulation;
- Population control should be ensured to prevent the neutron and precursor populations from shrinking to zero or growing unbounded. This is true even for the stationary regime, where the variance of the population grows linearly in time if population control is not enforced (the 'critical catastrophe');
- If the initial conditions correspond to a critical (i.e., equilibrium) regime, the particle source must be prepared in the equilibrium state before running the kinetic Monte Carlo simulation for the time-dependent transport.

The precursor particles do not directly contribute to Monte Carlo scores during the simulation: their role is to provide a 'buffer' for delayed neutrons and regulate the statistical weight of the neutron population that is physically transported and thus directly contributes to Monte Carlo scores. Precursors can be created at fission events and can possibly also be present at the beginning of the simulation in the case of a critical source. All possible precursors associated to a given fission event are combined into a single representative precursor particle carrying the total statistical weight, for the purpose of reducing the variance due to the dispersion of the decay constants. This method ensures unbiased estimates for the average scores. For more details, we refer the reader to Ref. [37].
The syntax for the `DYNAMIC` simulation mode is as follows:

**SIMULATION**
...
**DYNAMIC**
**GRID**
time_mesh_name
**POPULATION_CONTROL**
population_control_option
***POPULATION_IMPORTANCE_RATIO***
*population_importance_options*
***CRITICALITY_SOURCE***
*inactive_cycles decorrelation_cycles*
***CRITICALITY_GEOM***
*geom_name*
***TIME_GEOMETRY LIST*** *n_geometries*
*time_1 geom_name_1 ... time_n geom_name_n*
***BRANCHLESS***
...
**PARTICLES**
2 **NEUTRON PRECURSOR**
**TIME_INF**
**NEUTRON** time_inf
**TIME_SUP**
**NEUTRON** time_sup
**END_SIMULATION**

### 22.12.1 TIME MESH FOR FORCED PRECURSOR DECAY

In order to cope with the separation of the neutron and precursor time scales, the code uses a variance reduction method called 'forced decay': the idea is to modify the decay probability of the precursor particles, and yet preserve an unbiased Monte Carlo game by correspondingly modifying the statistical weight of the delayed neutrons emitted upon decay. This ensures a larger fraction of neutrons in the fission chains and thus

a reduced variance in the total population per unit time. Several choices are possible, the simplest being to introduce an arbitrary time mesh $t_0 < t_1 < \cdots < t_N$: the code will force precursors to decay uniformly in each bin of the mesh.
The time mesh must be defined in the GRID_LIST block and its name must be given as an argument to the GRID keyword, which must immediately follow the DYNAMIC keyword.

### 22.12.2 POPULATION CONTROL

Depending on the reactivity of the system, the population size (number of neutrons and precursors) may shrink to zero (subcritical case) or grow without bounds (supercritical case). Both scenarios are problematic and thus require some control mechanisms to the population size during the calculation (detailed below). For the sake of simplicity, population control is enforced at the end of the time intervals selected for forced decay. This has two advantages: the user only needs to input one time grid, and it does not introduce additional stopping times for the precursors. While the time steps are in principle arbitrary, in practice population control restricts the choice of the interval boundaries: if the system is super-critical, the population might diverge within a step; conversely, if the system is sub-critical the population might disappear within a step. Two population control methods are available in the code: Russian roulette & splitting, and combing.

#### 22.12.2.1 Russian roulette & splitting

The Russian roulette and splitting can be separately applied to the neutron and precursor populations banked at the end of each time step. The roulette is straightforward for neutrons. Concerning precursors, the roulette is applied on the basis of the expected delayed neutron weight that can be emitted during the following time step.
This choice corresponds to the use of POPULATION_CONTROL RUSSIAN_ROULETTE.

#### 22.12.2.2 Combing

The aim of combing is to normalize the number of transported particles while preserving their total weight and possibly the proportion of respective weights in the case of multiple particle species. For kinetic simulations, combing can achieve variance reduction and save computer time by keeping the population size approximately constant over time steps.
This choice corresponds to the use of POPULATION_CONTROL COMBING.

### 22.12.3 TIME MESH FOR SCORES

In addition to the simulation time grid for forced decay, a scoring time grid must also be defined. By construction, the contents of each bin $[\tau_q, \tau_{q+1}]$ will yield the ensemble-averaged time integral of the corresponding Monte Carlo score, between time $\tau_q$ and $\tau_{q+1}$. Unlike the time grid for precursor decay (and for population control), the time mesh for scoring is arbitrary, and the bins can be irregularly spaced and have any size.
Scoring time grids are specified in the SCORE block with the TIME GRID keywords (see Chapter 15).

### 22.12.4 RELATIVE PARTICLE IMPORTANCE

The kinetic simulation of fast transients requires a finely discretized time mesh in order to capture the fast time variations of the population. However, if the simulation time steps are too small, ordinary roulette will kill most of the delayed neutrons, even when forced decay is applied. Indeed, for small time steps, the statistical weight of delayed neutrons is proportional to the time interval $\tau_{q+1} - \tau_q$. For small time intervals, the weight of the delayed neutrons generated by forced decay is also small and particles will not survive the roulette. With no delayed neutrons to initiate new fission chains, the simulation will then suffer from severe under-sampling issues. In order to increase the number of neutrons present in the simulation and therefore improve the efficiency of simulations over short time scales, a time-dependent importance scheme can be used in the dynamic simulation. For each time step $i$, the user can provide an importance ratio $R_i = I_{N,i}/I_{C,i}$, where $I_{N,i}$ is the importance of neutrons and $I_{C,i}$ is the importance of precursors for tha step. At the beginning of a time step, including the first one, the code uses the current particle weights to estimate $I_{N,i}$ and $I_{C,i}$, in such a way that the 'physical' weights are preserved.
Note that the population weights are only adjusted if the importance ratio changes (nothing needs to be done if $R_{i+1} = R_i$).
The choice of this ratio should depend on the time step and the reactivity. In order to observe both fast and slow transients, one needs a non-uniform simulation time grid and importance factors. On one hand, fast transients can exclusively be observed over very short times (time steps should be of the order of the mean generation time); on the other hand, delayed neutrons need several seconds to be emitted. Hence, short time grids should be associated with the application of an importance factor to favour the forced decay of delayed neutrons. On the contrary, importance is not necessary for wide time grids since delayed neutrons are emitted with a large statistical weight.
The relative particle importance is defined on the time mesh for forced precursor decay (`DYNAMIC GRID`). If the user wants to specify a single importance value for the whole simulation, the syntax is as follows:

**POPULATION_IMPORTANCE_RATIO** <value>

If a time-dependent importance ratio needs to be specified, the syntax is the following:

**POPULATION_IMPORTANCE_RATIO**
**LIST** <n_values> <value_1> ... <value_n>

Here it is understood that the first value of the importance ratio refers to the first time step of the time grid for forced precursor decay, the second value refers to the second time step, and so on. If the number of values given here is smaller than the number of time steps in the time grid for forced precursor decay, then the importance ratio is assumed to take the last value, value_n, until the last time step.

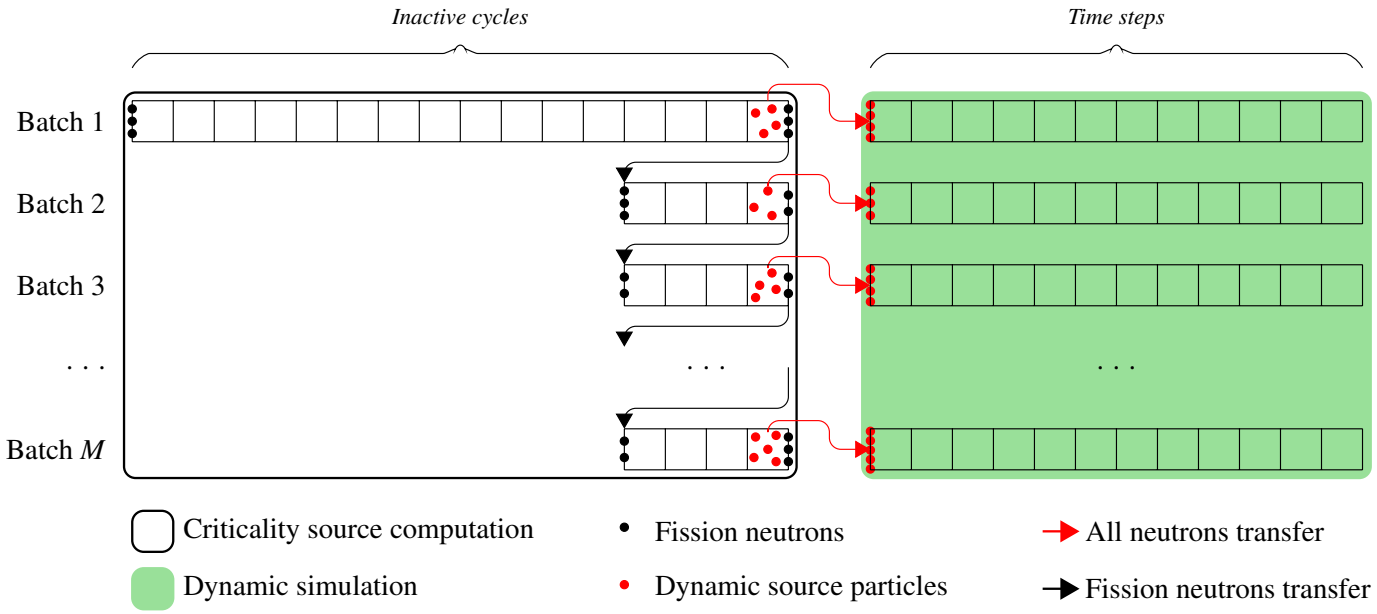



Figure 22.1: Kinetic simulation process with $M$ batches. A complete power iteration is performed once by the first batch. During the final cycle, precursors are sampled so that the dynamic simulation can begin while the generated fission neutrons are transferred to the second batch which runs a few additional power iterations in order to ensure reasonable decorrelation. The third batch gets the fission neutrons generated by the second one, and so on. In a parallel run, each processor would apply this scheme.

### 22.12.5 CRITICAL SOURCE

The typical configuration at the initial time for a kinetic Monte Carlo simulation is either a system containing only a neutron source (e.g., for reactor start-up calculations), or a system in a critical state (e.g., for the departure from a stationary regime). For the latter, the code applies the power iteration algorithm (i.e., a $k$-eigenvalue calculation) over a large number of inactive cycles, with neutrons alone, in order to obtain an estimate of the fundamental mode. During the last inactive cycle, neutron and precursor sources for the dynamic phase are sampled at each collision, based on the fundamental mode: the critical neutron flux is converted into a neutron and precursor population, whose time is set to $t = 0$. These 'critical' sources are then used for the dynamic simulation.
The code averages the dynamic calculation results over consecutive batches. Ideally, each batch should represent a genuine independent replica of the full calculation, namely a source-convergence phase followed by a kinetic phase. In order to reduce the number of inactive cycles, the fission neutrons generated in the last inactive cycle of the current batch are injected in the inactive cycles of the following one. This reduces the overhead of inactive cycles, at the cost of introducing correlations among batches (the estimation of the statistical uncertainty over the calculation results can be thus affected). Note, however, that this is essentially the same difficulty that one faces in conventional criticality calculations. A scheme of this process is illustrated in Fig. 22.1.
The number of inactive cycles and the number of additional decorrelation cycles between batches are specified with the `CRITICALITY_SOURCE` keyword:

**CRITICALITY_SOURCE** inactive_cycles decorrelation_cycles

The example of Fig. 22.1 corresponds to inactive_cycles=16 and decorrelation_cycles=4.

### 22.12.6 ADJUSTING THE REACTIVITY

The simulation of a nuclear reactor is necessarily affected by uncertainties on geometry details and on nuclear data. For this reason, the effective multiplication factor for nominally critical conditions may fail to be equal to 1. Even small deviations from 1 may be important in kinetic Monte Carlo simulations, because the system will drift away from the stationary state during the dynamic simulation. For this reason, in order to minimize these deviations, we propose a simulation option to rescale the average number of neutrons produced by fission by the multiplication factor (obtained by a preliminary criticality calculation). To use this option, use the optional `RESCALE_NU_BY_KEFF` keyword, which is detailed in Sec. 22.1.

### 22.12.7 CHANGING GEOMETRIES

In order to simulate transients involving moving reactor parts, the code allows modifying the geometry and material compositions at each time step. As long as all the geometries have the same bounding box, there is no constraint on their internal structure (number of volumes, composition of volumes, etc.). This new option, which for the moment is only available for ROOT geometries, paves the way for all types of time-dependent simulations: different geometries can be defined for the criticality step and for each time step of the dynamic simulation.
The use of multiple ROOT geometries requires the `MULTI_GEOM` block, which replaces the usual `ROOT_GEOMETRY` block at the beginning of the input file. The syntax of `MULTI_GEOM` is the following:

**MULTI_GEOM**
  **NAME** geom_name_1
    ROOT_geometry_specification_1
  **NAME** geom_name_2
    ROOT_geometry_specification_2
  ...
**END_MULTI_GEOM**

Any number of geometries can be defined. Each geometry must be given a unique name with the `NAME` keyword. The specification of the ROOT geometry may take any of the forms allowed in the `ROOT_GEOMETRY` block, which are described in Chapter 36. In practice, the geometry specification may be any of the following: the name of a ROOT file containing a geometry, a `CPP_ROOT_GEOM` block, or a `PYTHON_ROOT_GEOM` block.
The geometries defined in the `MULTI_GEOM` block can be used in dynamic simulations with the `CRITICALITY_GEOM` and `TIME_GEOMETRY` keywords. For the former, the syntax is:

**CRITICALITY_GEOM** geometry_name

Here geometry_name must be the name of one of the geometries in MULTI_GEOM. It identifies the geometry that will be used for the criticality part of the dynamic simulation, if CRITICALITY_SOURCE is specified. The syntax of the TIME_GEOMETRY keyword is:

**TIME_GEOMETRY LIST** n_geometries
time_1 geom_name_1 ... time_n geom_name_n

The time_i values must be part of the time grid for forced precursor decay; they indicate the times at which the geometry changes. The geom_name_i strings must be geometry names, as declared in MULTI_GEOM, and they indicate the geometry that will be used for time_i $< t <$ time_i+1. The last geometry is assumed to last until the end of the last time step.

### 22.12.8 BRANCHLESS COLLISIONS

The fission chains in Monte Carlo transport are responsible for increased variance and spatial correlations in multiplying media with respect to the case of purely diffusing and absorbing configurations. These fluctuations, whose origin lies in the fission-induced spatial correlations between successive generations, might globally hinder the convergence of the kinetic Monte Carlo calculations. In TRIPOLI-4 we have chosen to cope with the variance associated to fission chains by implementing the branchless collisions method. The idea behind this algorithm is to suppress the variability due to the simultaneous propagation of the several branches associated to a fission event (i.e., the histories of all the neutrons descending from a common ancestor) and to collapse all the contributions into a single history carrying the average weight of all the descendants. In principle, the branchless transport has been shown to be quite effective in reducing the variance in multiplying systems [97, 38].
Branchless collisions can be activated with the optional BRANCHLESS keyword. This mode is only available in DYNAMIC mode.

## 22.13 EPITHERMAL BROADENING OPTIONS

In the context of neutron interactions with matter, it may become necessary to take into account relative velocity effects between the incident neutron and the collided nucleus at thermal equilibrium. The thermal agitation of collided nuclei is indeed responsible for Doppler broadened cross sections as well as scattering kernels. Doppler broadening of cross sections at a given medium temperature is typically taken into account by the external preprocessing of nuclear data, whereas Doppler broadening of the scattering kernels must be taken into account by the Monte Carlo code itself. This is usually required for elastic collisions in the epithermal region, say in the range from 1 eV to 1 keV, where the two energies become comparable.
Below the epithermal region, interactions are either dominated by $1/v$ radiative capture (so that there is no re-emitted neutron), or assumed to be described by more sophisticated models, such as elastic or inelastic

thermal kernels (of the $S(\alpha,\beta)$ kind, for instance, which take into account lattice structures and/or molecular bonds).
Above the epithermal region, on the contrary, the neutron energy is larger than the typical nucleus (Maxwell-Boltzmann) energy, so that the nucleus can be considered at rest as seen from the incoming neutron: in this case, standard two-body kinematics can be applied for the elastic collisions, which results in the so-called asymptotic elastic scattering kernel.
The intermediate epithermal region needs to be described by a specific model that takes into account relative velocity effects on the behavior of elastic cross sections and elastic scattering kernel (this latter being the double differential component of the former). The Doppler broadening of elastic scattering kernels must be consistent with that of the associated cross sections.
The default model for the Doppler broadening of elastic scattering kernels in TRIPOLI-4® is the so-called SVT (Sampling of the Velocity of the Target nucleus), which assumes that nuclei are at thermal equilibrium with a Mawxell-Boltzmann velocity distribution and derives the scattered neutron parameters (direction and speed) from two-body kinematics [4].
The SVT model is based on the hypothesis that the elastic cross sections in the epithermal region are approximately flat, which is true for light nuclei, but breaks down for heavy resonant nuclei such as $^{238}U$. If no optional keyword for Doppler broadening is specified in the `SIMULATION` block, then TRIPOLI-4® implicitly adopts the SVT model in the region from $10^{-5}$ eV to an upper threshold energy $E_{threshold}$ for the incident neutron, with $E_{threshold} = 400 k_B T$, $k_B = 8.617 \cdot 10^{-5}$ [eV/K] being the Boltzmann constant, and $T$ the medium temperature in Kelvin. Above $E_{threshold}$ the asymptotic scattering kernel is used instead.
If in the thermal region (say between $10^{-5}$ eV and $E_{therm}$, with $E_{therm}$ typically of the order of 5 eV) thermal elastic cross sections are available, TRIPOLI-4® uses these thermal sections up to $E_{therm}$, and above $E_{therm}$ chooses the SVT or the asymptotic elastic kernel according to whether the incident neutron energy is below or above $E_{threshold}$, respectively.
If the user needs to specify a different Doppler broadening treatment, then the optional keyword `EPITHERMAL_BROADENING` must be declared in the `SIMULATION` block. The syntax then depends on which model the user wants to adopt, and what are the parameters to be defined for the chosen model. The possible (mutually exclusive) options are the following:

### 22.13.1 SVT MODEL WITH A DIFFERENT THRESHOLD ENERGY $E_{threshold}$

The syntax for using the SVT model with a different threshold energy $E_{threshold}$ is

```
SIMULATION
    ...
    EPITHERMAL_BROADENING
        SVT UPPER_ENERGY E_threshold
        ONLY_SVT
    ...
END_SIMULATION
```

where $E_{threshold}$ is the user-defined energy threshold that applies to all the isotopes, expressed in units of MeV.

### 22.13.2 DOPPLER BROADENING REJECTION CORRECTION (DBRC)

The Doppler Broadening Rejection Correction (DBRC) has been especially conceived for improving the SVT model in presence of heavy nuclei, whose elastic cross sections typically display strong resonances in the epithermal region [5]. The DBRC model is implemented as follows: up to 0.1 eV, standard SVT is applied, since it is assumed that in this region there are no resonances in the elastic cross section. Above 0.1 eV, and up to a threshold energy $E_{dbrc}$ defined by the user (usually at some hundreds of eV), the DBRC is applied. Above $E_{threshold}$, the asymptotic elastic kernel is applied.
The user can further specify a list of nuclei for which the DBRC correction is to be used. For the isotopes that are not in the list, standard SVT is applied instead, up to an energy $E_{threshold}$. The syntax for DBRC depends on whether the user wants to give an exhaustive list of isotopes or prefers to give a window of atomic weights (this latter option is preferable when writing down the entire list might be too cumbersome).
In the former case, the syntax is

**SIMULATION**
...
**EPITHERMAL_BROADENING**
**SVT UPPER_ENERGY** $E_{threshold}$
**DBRC UPPER_ENERGY** $E_{dbrc}$
**NUCLEI** $NB_{nuclei}$ $NUC_1 \cdots NUC_n$
...
**END_SIMULATION**

The values $E_{threshold}$ and $E_{dbrc}$ are to be expressed in MeV.
In the latter case, the DBRC model is applied to all nuclei whose atomic weight $A$ lies between $A_{min}$ and $A_{max}$, and the syntax is

**SIMULATION**
...
**EPITHERMAL_BROADENING**
**SVT UPPER_ENERGY** $E_{threshold}$
**DBRC UPPER_ENERGY** $E_{dbrc}$
**ATOMIC_WEIGHT_WINDOW** $A_{min}$ $A_{max}$
...
**END_SIMULATION**

### 22.13.3 WEIGHT CORRECTION METHOD (WCM)

The Weight Correction Method (WCM) model has been conceived as an alternative to DBRC for improving the SVT model in presence of heavy resonant nuclei [25]. The WCM model is implemented as follows: up to 0.1 eV, standard SVT is applied, since it is assumed that in this region there are no resonances in the elastic cross section. Above 0.1 eV, and up to a threshold energy $E_{wcm}$ defined by the user (usually at some hundreds of eV), the WCM is applied. Above $E_{threshold}$, the asymptotic elastic kernel is applied. The user can further specify a list of nuclei for which the WCM correction is to be used.
The user can further specify a list of nuclei for which the WCM correction is to be used. For the isotopes that are not in the list, standard SVT is applied instead, up to an energy $E_{threshold}$. The syntax for WCM depends on whether the user wants to give an exhaustive list of isotopes or prefers to give a window of atomic weights (this latter option is preferable when writing down the entire list might be too cumbersome).
In the former case, the syntax is

```
SIMULATION
    ...
    EPITHERMAL_BROADENING
        SVT UPPER_ENERGY E_threshold
        WCM UPPER_ENERGY E_wcm
        NUCLEI NB_nuclei NUC_1 ··· NUC_n
    ...
END_SIMULATION
```

The values $E_{threshold}$ and $E_{dbrc}$ are to be expressed in MeV.
In the latter case, the WCM model is applied to all nuclei whose atomic weight $A$ lies between $A_{min}$ and $A_{max}$, and the syntax is

```
SIMULATION
    ...
    EPITHERMAL_BROADENING
        SVT UPPER_ENERGY E_threshold
        WCM UPPER_ENERGY E_wcm
        ATOMIC_WEIGHT_WINDOW A_min A_max
    ...
END_SIMULATION
```

**Remark**: setting for instance an atomic weight window with $A_{min} = 0.5$ and $A_{max} = 300$ would mean that all nuclei are to be corrected with either DBRC or WCM models.

## 22.14 ELECTROMAGNETIC SHOWER OPTIONS

The keywords related to the electromagnetic shower are:

- ELECTRON_INELASTIC_CUTOFF defines the inelastic cut-off. Only electrons with energies higher than this threshold are produced. By default, the code sets the threshold at 0.001 MeV.
- THIN_LAYER_SUBSTEP $< n >$ allows changing the number by which the transport step is divided when a particle approaches a frontier. Making substeps allows a more rigorous processing of frontier approaches. The default value is 8.
- ENERGY_LOSS_BY_STEP $< p >$ allows changing the parameter used to estimate step size in lepton transport. This step corresponds to the length when the electrons or positrons have lost a proportion $< p >$ of their energy by collisions. This length is calculated using CSDA models (Continuous Slowing Down Approximation). The default value is currently 0.02.
- ELECTRON_BREMS_CUTOFF defines the bremsstrahlung cut-off. Only photons with energies higher than the threshold are produced. By default, the code sets the threshold at 0.001 MeV.
- THICK_TARGET_BREMSSTRAHLUNG activates a simplified simulation mode for the electromagnetic shower. When this option is used, secondary electrons and positrons produced by photon collisions are not transported, but part of their energy is converted into new bremsstrahlung photons. This is much faster (but slightly less accurate) than the full simulation of the electromagnetic shower. This option requires electron and positron transport to be activated in the SIMULATION block. Note that it can be used even with electron/positron sources.
- BREMSSTRAHLUNG_MODEL selects the model to be used for the bremsstrahlung cross section. Until version 10.2, TRIPOLI-4® used Berger and Seltzer's 1970 bremsstrahlung model [6]. This model is still available and can be selected with the BREMSSTRAHLUNG_MODEL 0 keyword.

  Starting with TRIPOLI-4® version 11, the code uses Seltzer and Berger's 1986 model by default [96] (keyword: BREMSSTRAHLUNG_MODEL 1). This model requires external cross-section files, which are shipped with the TRIPOLI-4® installation. TRIPOLI-4® will first search for the cross-section files in the electron directory specified in the path file (see Section 3.1). If they are not found there, if the path file is specified in the new format (Section 3.1.2), and if it contains the electron_bremsstrahlung_xs key, TRIPOLI-4® will look in that directory, too. An error will be raised if the cross-section files cannot be found. The new path file format makes it possible to use the new bremsstrahlung model even if the cross-section files are not yet included in the nuclear data library.
- NO_ELECTRON_MULTIPLE_SCATTERING disables the multiple scattering simulation operation. This function is not available in this version of the code.
- ELECTRON_TOTAL_STOPPING_POWER makes energy losses due to inelastic collisions continuous throughout the energy domain. Secondary particles are produced independently. By default, this option is not enabled.

- Combined with the previous option, NO_ELECTRON_STRAGGLING allows disabling the fluctuations on energy losses of the particles. The default CSDA mode (Continuous Slowing Down Approximation) of TRIPOLI-4® includes these fluctuations. The default mode includes more physics, but CSDA without straggling may be better suited for comparisons with other codes.
- ELECTRON_PHOTON_BALANCE is an option that activates the calculation of the balance (in terms of number of particles, weight and energy) of the simulation. The balance is produced each time results are produced. By default, this option is not activated in the code.
- NO_ANOMALOUS_SCATTERING_FACTOR set all photon anomalous scattering factors to 0. This option can be useful to carry out comparisons to other calculation codes.
- ELECTRON_MIN_STEP_LENGTH sets the value of the minimum length for electron/positron steps. If the stopping power or the cross sections for hard collisions suggest a step length shorter than this value, then the step length is artificially extended, and the energy loss along the step is adjusted accordingly. This option affects the collision density and the energy deposition in thin layers. The default value is 1e-4 (cm), but smaller values may yield slightly better predictions, especially for low-energy electrons.

**Important notes concerning electromagnetic showers :**

1. Due to interaction relationships, it is not possible to simulate electrons or positrons alone.
2. The minimal photon, electron, positron kinetic energy is about 1 keV, below this energy atomic shell models currently implemented become inadequate.
3. In the current version of TRIPOLI-4®, if NJOY cross section option is used (see chapter 1), neutron data are always loaded even if no neutron is transported.

   In order to avoid this, use ENDL cross section option (and then ENDL photon data).
4. Fluorescence photons are emitted after photoelectron emission, then, currently, there is fluorescence only if electrons are tracked.
5. If no DEPOSITED_SPECTRUM score is present, then photoelectric interaction is modelized as non analog by weight reduction.
6. Be careful when comparing TRIPOLI-4® results in photon-only transport mode with other codes. Other codes can include bremsstrahlung photons even if electrons are not transported (thick-target bremsstrahlung). This feature is present in TRIPOLI-4®, but needs to be activated with the THICK_TARGET_BREMSSTRAHLUNG option.
7. Due to the minimal photon/electron/positron energy of 1 keV, the minimal group energy width (in DEPOSITED_SPECTRUM spectrum for example) is also 1 keV.
8. Currently, DEPOSITED_SPECTRUM scores are incompatible with biasing techniques. In case of necessity, in first approximation, one can couple by hand a photon flux calculation (FLUX_PT for example) at the entrance of the detector area and a DEPOSITED_SPECTRUM calculation.

9. Electron and positron transport is very time consuming. To speed up the calculation, taking into account the low electron/positron range, one can set high energy cut on electron/positron/photon in low interest volumes (far from the detector for example) using VOLENERCUT, see chapter 37.

## 22.15 OPTION PHOTON_NEUTRON_RATIO

If the keyword PHOTON_NEUTRON_RATIO is used in a $n-\gamma$ calculation, the population of photons can be biased (as soon as they are generated), using splitting or russian roulette. The ratio stands here for the mean number of photons generated by neutron reactions, over the number of simulated neutrons in a batch. The user has the choice either to set a global ratio value for all the geometry or to define ratio values for particular volumes of the geometry. The syntax is as follows:

```
PHOTON_NEUTRON_RATIO
*          LIST  nb(volumes)
                 volume(1) ratio(1)
                 volume(2) ratio(2)
                 ...
                 volume(n) ratio(n)
*          global ratio
```

where volume(i) is a volume number and ratio(i) its associated ratio. All "ratio" parameters are real numbers.

- By default, the ratio is equal to 1. In this case, the aim is to have a population of photons equal to the population of neutrons.
- This parameter can be increased (or decreased), which leads to multiplying the splitting and Russian roulette thresholds by "global ratio" (or (“ratio(i)” for a specific volume i).
- If the choice of ratio per volume is done (with the keyword LIST), the ratio of non listed volumes is kept to 1.

## 22.16 OPTION ALL_COLLIDED_FLUX_PT_POINTS

This option enables a score contribution of all flux points (FLUX_PT) at each deterministic collision. By default a flux point is randomly selected. This option is useful if the point flux scores are used to obtain a picture in imagery or radiography device.

## 22.17 OPTION SUPER_HISTORY

The Super-history method consists of simulating several successive generations in a single batch without performing any renormalization of the neutron population by $k_{eff}$. Such approach has been originally used to reduce correlations between batches. This method is currently available only for the CRITICALITY simulation mode.

The SUPER_HISTORY keyword may be followed by the LATENT <nb_latent_generations> keyword to specify the number of additional generations simulated per batch. The default behavior is 10 latent generations.

## 22.18 OPTION WIELANDT_METHOD

The Wielandt method has been originally adapted to Monte Carlo calculations for convergence acceleration purposes. The method consists in subtracting an extra fission source term on both sides of the transport equation, which yields

$$\mathbf{T}\psi - \mathbf{S}\psi - \frac{1}{k_e}\mathbf{F}\psi = (\frac{1}{k_0} - \frac{1}{k_e})\mathbf{F}\psi, \tag{22.22}$$

where **T**, **S** and **F** denote the usual loss, scattering and fission operators, respectively, $k_0$ is the fundamental eigenvalue and $k_e$ is the Wielandt control parameter, to be chosen as $k_e > k_0$. If Eq. (22.22) is to be solved by power iteration, the fission operator term on the left of the equation is interpreted as additional fission neutrons that will be tracked in the current cycle. Thus, at every collision, $n_{next}$ fission neutrons are emitted for the next cycle, and $n_{current}$ for the current cycle, according to

$$n_{next} = \frac{\nu\Sigma_f}{\Sigma_t}\frac{1}{k_w^{n-1}} + \xi \qquad n_{current} = \frac{\nu\Sigma_f}{\Sigma_t}\frac{1}{k_e^{n-1}} + \xi, \tag{22.23}$$

$\xi$ being a uniform random number. The Wielandt method is basically an eigenvalue-shift approach that reduces the dominance ratio (thus improving the convergence rate of the power iteration) while preserving the eigenfunctions. However, it may be difficult to choose a proper value for $k_e$, since the fundamental eigenvalue $k_0$ is not known in advance: large fluctuations in the neutron population may appear in the early cycles of the Wielandt method when imposing a fixed value $k_e$. Thus, It was proposed to use $\Delta = k_e - k_0$ rather than $k_e$ as the control parameter, and furthermore introduced an iteration scheme where $\Delta$ is allowed to evolve along cycles in order to mitigate population fluctuations. The average length of the (subcritical) "fission chains" simulated within each cycle has been shown to be inversely proportional to $\Delta$ : $L \simeq 1 + k_0/\Delta$. In the TRIPOLI-4® code, three different convergence schemes for $\Delta$, namely a constant $\Delta$, a linear approach to a target $\Delta$ and a user-defined function for the approach to a target $\Delta$. The default method is a linear approach where $\Delta_{start} = 1.0$ and $\Delta_{end} = 0.1$ with a decrease of 0.05 at each batch leading to a number of steps of 18.

**WIELANDT_METHOD**
\# DEFAULT
\# CONSTANT $\Delta$
\# LINEAR $\Delta_{start}\Delta_{end}$ nb_steps
\# USER nb_steps $\Delta_1 \ldots \Delta_{nb_steps}$

## 22.19 THE MULTIPLE SIMULATION MODE (NEUTRONICS-SHIELDING

## COUPLING)

### 22.19.1 OBJECTIVE

Many applications require a sequence of several separate computations, *e.g.* in order to obtain in a first step the neutron productions in the multiplying area and, in a second step, the propagation of neutrons towards detectors that are more or less distant from the multiplying area, with possible strong attenuations. However, the drawback of these two-step separate calculation schemes is to leave to the user the proper preparation of the source for the second calculation. The direct computation of the response of these detectors (*i.e.* the neutronics-shielding coupling) aims at providing in a single computation a precise distribution of the fission neutrons of the core together with their efficient propagation towards the detectors. This feature is devoted either to fixed-sources sub-criticality or to criticality configurations (with a control of the source convergence).

### 22.19.2 PRINCIPLE

The coupling consists in linking the following two steps within each simulated batch:

- Storing the fission neutrons produced either during a fixed-source criticality batch or during a criticality batch
- Using them, in the same batch, as source neutrons for the shielding step (this allows using one of the code's variance reduction techniques, to propagate the neutrons through large distances with strong attenuations). The AMS technique (see Chapter 18) was chosen for this coupling because it is specifically well suited: no normalization of the importance function is needed between the two steps of each batch since this function is only used for sorting purpose and not for weight management of particles.

The global convergence is achieved by accumulating batches, each one consisting of the two steps.
In the case of criticality computations, the second step is performed only after having discarded the requested number of batches given by the user (for instance 100 or 200 batches can be discarded).

### 22.19.3 HISTORY OF THE MULTIPLE SIMULATION MODE

The multiple simulation mode was available in previous versions of TRIPOLI-4® and present in the related user guides, from version 4 to version 9 of the code. The second step of this simulation mode used the Exponential Transform as variance reduction technique, with the strong limitation of a single energy group for the importance map (due to normalization constraints in the code), which made this former multiple mode feature not easy to use and often not very efficient (and with possible variance jump problems). This feature was then removed from the user guide of versions 10 and 11. Starting from version 12 of the code, the multiple simulation mode is back, since it has been revisited with the help of the AMS variance reduction technique (see Chapter 18), which does not need any normalization process and can thus be used as such in this context.

### 22.19.4 SYNTAX

The syntax of the multiple simulation mode is as follows:

```
SIMULATION
    ...
    MULTIPLE
        <number_steps>
*       FIXED_SOURCES_CRITICALITY
*       CRITICALITY  DISCARD  <nb_discarded>
            SCORE
*               NONE
*               <number> <scorename_1>...<scorename_n>
        SHIELDING
            SCORE
*               NONE
*               <number> <scorename_1>...<scorename_n>
            VARIANCE_REDUCTION
*               NONE
*               1 <particle_type>  AMS  SPATIAL
*               1 <particle_type>  AMS  <variance_reduction_name>
            ENERGY_INF
                <particle_type> energy
    ...
END_SIMULATION
```

where:

- <number_steps> equals 2 (which is actually linked to a more general syntax of the former multiple mode of versions 4 to 9 of the code); the same remark applies to the values “1” present in the VARIANCE_REDUCTION data of the present command block.
- <nb_discarded> is the number of discarded batches in the criticality case (of the first step). This data is mandatory (and can be 0). During the first discarded batches of the simulation, only the first step is performed by the code, then the two steps are active and contributions to the scores requested in the shielding step are taken into account.
- The scores and biasing names must have been defined in the SCORE and VARIANCE_REDUCTION command blocks (see Chapters 15 and 17) of the input deck.
- The keyword NONE indicates that it is not necessary to choose a score or a biasing.

- The AMS can be used either with a spatial importance function (with the keyword SPATIAL) or with an importance map (in space and energy). In the latter case, it has to be given through an external importance map, *e.g.* previously stored with the STORE_IN_FILE option (see Chapter 17) or provided by another code with the same file format.
- The ENERGY_INF keyword allows for an energy cut-off applying to the second step (shielding calculation). During the first step (fixed-source criticality or criticality), the lower energy is $10^{-11}$ MeV. Presently, <particle_type> equals NEUTRON.

Remark:

- The multiple mode with AMS needs the presence of an AMS command block (see Chapter 18) in the input deck.

## 22.20 STORAGE OF PARTICLE EVENTS

The storage of particle events (sources, collisions and leakage) can be done using the keywords STORE_SOURCES_IN_FILE, STORE_COLLISIONS_IN_FILE and STORE_LEAKAGE_IN_FILE. STORE_SOURCES_IN_FILE can be used as a way to restart a simulation from source points (see section 12.6.1). It can also be used to plot particle history events in T4G (see section 38.3.3.3). This feature is described in the section 31.4.

## 22.21 COUNTING HOLES WITHIN THE SIMULATION OPTIONS

### 22.21.1 OPTIONS HOLE_COUNT, HOLE_MAP, HOLE_VOLU_MAP

Here we present a family of options that allow the user to localize holes and estimate their statistical weight during the simulation (i.e. the results are bound to the systematic error on the scores of the simulation).
There are several levels of precisions for this counting :

- **Hole counting**: It counts holes with a precision of $\Lambda_h$, stores and dump the information as a global output: the **Mean hole leakage**. It is requested with keyword HOLE_COUNT.
- **Hole map**: This option allows the user to require the dumping of the position of the different holes with their respective weight in addition to the **Mean hole leakage**. It is requested with the keyword HOLE_MAP.
- **Hole map with volumes**: Same as above but with the detected nearby volumes. It is requested with the keyword HOLE_VOLU_MAP.

The method is based on a simple test. As soon as a particle is detected as leaving the geometry, it checks if there's a volume at a test point $\overrightarrow{P_t}$ defined as :

$$\overrightarrow{P_t} = \overrightarrow{P_p} + \Lambda_h . \overrightarrow{\Omega_p}$$

$\overrightarrow{P_p}$ is the position of the particle leaving the geometry and $\overrightarrow{\Omega_p}$ is its direction.
This test cannot be infinitely precise. It cannot detect holes whose length in the $\overrightarrow{\Omega_p}$ is superior to $\Lambda_h$ and holes which are closer than $\Lambda_h$ from the true boundaries of the geometry.
The $\Lambda_h$ value can be set by user with the keyword LAMBDA_HOLE, default value is $10^{-2}$. Because of the intrisic precision of TRIPOLI-4® geometry, $\varepsilon_{geo} = 10^{-4}$, a $\Lambda_h$ value inferior to it would be useless and is therefore not accepted.

### 22.21.2 PROCEDURE FOR TESTING GEOMETRY

This feature can be used as an other form of geometry check (see 7.13). The present alternative is a way of checking the sanity of complex and extended geometries with the following procedure:

- Do not initialize any useless item for the test: grids (energy, space, angle, time), responses, scores, boundary conditions, biaising, perturbations, collision bands, verification calculations, outputs, couplings ...
- Place sources in several places inside the geometry. The more sources the more precise hole localisation will be.
- Replace compositions by vacuum in all volumes thanks to keyword ALL_VOLUMES.
- Add one of the counting options above.

The data set should look like :

```
GEOMETRY
        Geometry to be tested
```

```
COMPOSITION  1
        POINT_WISE  300 name(compo)
                1
                name(isotope)  0
END_COMPOSITION
```

```
GEOMCOMP
        name(compo)  ALL_VOLUMES
END_GEOMCOMP
```

```
SOURCES_LIST   nb(sources)
        SOURCE
                Description of the first source
        END_SOURCE
        ...
        SOURCE
                Description of the last source
        END_SOURCE
END_SOURCES_LIST
```

```
SIMULATION
        BATCH   nb(batches)
        SIZE   size
        PARTICLE   nb(particles)  <PARTICLE_TYPE>
        <HOLE_COUNTING_OPTION>
        *  LAMBDA_HOLE   value
END_SIMULATION
```

## Defining simulation parameters for the criticality calculation example

```
SIMULATION
          CRITICALITY
          BATCH 800
          SIZE 500
          EDITION 10
          PARTICLE
               1
               NEUTRON
          ENERGY_INF
               NEUTRON
               1.E-11
          KEEP_RESULT
END_SIMULATION
```

# Chapter 23

# CALCULATION VERIFICATION DEFINITIONS

Definitions begin with the keyword CALCULATION.
This command is used to make a number of checks regarding:

- problem geometry (sub-command VOLUME);
- problem compositions (sub-command DENSITY).
- problem mass (sub-command MASS).

```
            CALCULATION
*                 (SET)  VOLUME
                        Volume calculation parameters
                  END_VOLUME
*                 DENSITY
                        Density calculation parameters
                  END_DENSITY
*                 MASS
                        Mass calculation parameters
                  END_MASS
            END_CALCULATION
```

where the keyword SET means that the values found by the calculation are assigned to volumes for the score results (which are then given per $cm^3$ for volume-based scores and by $cm^2$ for surface-based scores).

## 23.1 VOLUME CALCULATIONS

To define calculation verifications, it is possible to request a calculation of the volumes defined in GEOMETRY command. The volumes to be calculated are contained in a window defined in a reference frame specified by the user.

The syntax is as follows:

```
VOLUME
        FRAME <FRAME_TYPE>
                Ox Oy Oz
                ix iy iz
                (jx jy jz)
                kx ky kz
        WINDOW
                Umin Vmin Wmin
                Umax Vmax Wmax
        PRECISION precision
        MAX_POINTS maxpoints
        SUM
                Volume sum parameters
        STORE_IN_FILE Flag
END_VOLUME
```

It is necessary to define:

- the window definition reference frame, defined using the type of coordinates (Cartesian, cylindrical or spherical), the origin **O**, and the associated axes **i**, **j**, and **k** (see section 4.5);
- two points, corresponding to the minimum and maximum dimensions, define the window. These points are `Umin`, `Vmin`, `Wmin` on the one hand and `Umax`, `Vmax`, `Wmax` on the other, with $U$, $V$, $W$ being the coordinates $(x,y,z)$, $(r,\theta,z)$ or $(r,\phi,\theta)$ depending on the type of reference frame selected;
- precision of the calculation relatively to the volume values;
- maximal number of sobol points for the volumes calculation. The default value for `maxpoints` is $10^9$ points.
- **SUM** triggers the sum of volumes calculation. *V*olume sum parameters are needed for this option.
- **STORE_IN_FILE** is an optional keyword. If used calculated values are then stored in an ASCII file. Flag is set to 0 to create a file with data stored in **VOLSURF** format suffixed by ".vs". Flag is set to 1 to create in input file of T4G suffixed by ".vvls". However values calculated with the **SUM** option are not stored.

The calculation of the volumes is not exact but corresponds to an estimate of the actual volume with the calculation based on an algorithm involving pseudo-random numbers.
The calculation is performed in reference frame $(O,\mathbf{i},\mathbf{j},\mathbf{k})$, with the type of coordinates chosen. (Preference shall be given to a calculation reference frame adapted to the geometry.): `<FRAME_TYPE>` can be `CARTESIAN`, `SPHERE`, or `CYLINDER`.
The line `jx jy jz` is only given in the `CARTESIAN` case.

In the case where the **SUM** is triggered, *V*olume sum parameters can be defined as follows:

```
SUM
        nb-volu(1)
        volu(1)...volu[nb-volu(1)]
SUM   ALL
```

- The keyword **SUM** can be called as many times as wanted.
- The keyword **ALL** implies that there is nothing after except another **SUM** or **END_VOLUME**

## 23.2 DENSITY CALCULATIONS

It is also possible to request calculation of densities (in g/cm$^3$) of compositions defined as the number of atoms in the `COMPOSITION` command. It is necessary to determine whether the calculation relates to:

- all the compositions;
- only part of them.

The density calculation is an exact one:

$$\rho = \sum_{isotopes} \frac{N_i A_i}{\mathcal{N}}$$

where:

- $N_i$ is the number of atoms per unit volume of isotope $i$;
- $A_i$ is the atomic weight of isotope $i$;
- $\mathcal{N}$ is Avogadro's number.

The densities of all the compositions can be requested:

```
DENSITY
        ALL
END_DENSITY
```

or requested only for `nb(compo)` compositions:

```
DENSITY
    LIST
        nb(compo)
            name(compo_1)
            ...
            name(compo_n)
END_DENSITY
```

## 23.3 MASS CALCULATIONS

It is also possible to request calculation of mass (in g). The mass calculations are based on the previous results of volume values and densities. Therefore theses calculations have to be defined before.

In particular one must be aware that the calculated mass are only the mass which are inside the window defined in the volume calculation section.

Three different kinds of mass can be calculated. The mass of :

- a volume.
- a composition.
- a nucleus.
- an element.

The calculation parameters can be the keywords **ALL** or **LIST** following the syntax described in the density section.

**MASS**
**COMPOSITION**
*C*ompositon calculation parameters
**END_COMPOSITION**
**NUCLEI**
*N*uclei calculation parameters
**END_NUCLEI**
**ELEMENT**
*E*lement calculation parameters
**END_ELEMENT**
**END_MASS**

## Defining calculations for the criticality calculation example of chapter 5

```
CALCULATION
   VOLUME
      FRAME
         CARTESIAN
            0.  0.  0.
            1.  0.  0.
            0.  1.  0.
            0.  0.  1.
      WINDOW
         -150.  -150.  -150
         150.  150.  150.
      PRECISION 0.01
   END_VOLUME
   DENSITY
      LIST 4
         INOX_STEEL
         AIR
         SOLUTION_U
         CONCRETE
   END_DENSITY
   MASS
      VOLUME
         ALL
      END_VOLUME
      COMPOSITION
         LIST 4
            INOX_STEEL
            AIR
```

```
            SOLUTION_U
            CONCRETE
      END_COMPOSITION
      NUCLEI
         ALL
      END_NUCLEI
      ELEMENT
         LIST 2
            U
            O
      END_ELEMENT
   END_MASS
END_CALCULATION
```

# Chapter 24

# KIJ MATRIX CALCULATION

The Kij matrix gives the number of neutrons produced by fission in the volume i from a neutron emitted in the volume j.
His highest eigenvalue is equal to the Keffective of the system and the corresponding eigenvector represents the neutron sources in the volumes.
Only volumes containing a fissile material are taken into consideration, therefore the size of the matrix is equal to the number of these volumes.

The calculation is also possible with an `EXTENDED_MESH`. In this case the size of the matrix is equal to the number of cells (all cells are taken into consideration even if some of them do not contain any fissile material).

**Change warning:** Since version 11, an additionnal keyword `KIJ_MATRIX` must be included in the simulation block if the Kij matrix is to be calculated. This allows for having multiple simulations, not all having Kij calculations.

## 24.1 SYNTAX

The general syntax used to define Kij matrix is as follows :

```
MATRIX
        KIJ_MATRIX
        EXTENDED_MESH
                VTK
                Extended mesh definition
END_MATRIX
```

For the optional definition of an extended mesh, see section 15.8.
The optional keyword `VTK` allows to get 3D PARAVIEW outputs of the fundamental mode.

## 24.2 OUTPUT

A Keffective sensibility matrix is also given in mono-processor mode. The standard deviation of each matrix coefficient is given in a matrix. Each cell represents by how much the Keffective would be changed if this cell in the kij matrix were shifted by one standard deviation.

# Chapter 25

# DISPLAY OF COLLISIONS SITES

This feature allows for the display of the locations where collisions occur during a simulation. For example, the aim can be to display the efficiency of a chosen variance reduction technique. The simulation is carried out on a relatively few number of particles and the display carried out on 2D-views of the geometry. Thus we have to keep in mind that this feature only gives the user an indication of the collision site locations to be used for checks or adjustments (to the chosen variance reduction technique for example). Several options can help the user to analyze the current simulated case.

## 25.1 SYNTAX

The general syntax used to define collisions sites is as follows:

```
            COLLISION_SITES
                <PARTICLE_TYPE>
                height
                DISCARD nb(batches)
#               SCORE
#               SCORE_SEGMENTS
#               SOURCE
#               NB_COLLISION nb(collisions)
                GRID
                    nb(bounds)
                    energy(1)... energy(nb(bounds))
                COLOR
                    nb(colors)
                    <COLOR(1)> nb(groups_1)
                        group(1)... group(nb(groups_n))
                    ...
                    <COLOR(nb(colors))> nb(groups_n)
                        group(1)... group(nb(groups_n))
*               GRAPH
                    Px Py Pz
                    Lu Lv
                    <AXIS>
                    Flag
                     FILE filename
*               GRAPH_CONTOURS
                    Px Py Pz
                    Lu Lv
                    <AXIS>
                    Flag
                     FILE filename
            END_COLLISION_SITES
```

### 25.1.1 TYPES OF PARTICLES

Presently, <PARTICLE_TYPE> can be NEUTRON or PHOTON.
In order to obtain plots of collisions sites, for either neutrons or photons, the data block COLLISION_SITES ... END_COLLISION_SITES must first be written for neutrons and a second block for photons. During the execution, a message specifies which particle is related to each plot.

### 25.1.2 VARIATION OF THE PARAMETER HEIGHT

This parameter refers to the maximal distance to the view plane (defined by GRAPH or GRAPH_CONTOURS) on which the point (collision site) will be projected. This value is specified by the user. For geometries of about one meter, a choice between 1 and 10 for this parameter gives satisfactory results. Of course, the larger height is, the more points there will be on the graphic. So, we have to find a compromise between visual quality and the representation of reality. However, the user may want to deliberately project a full part of a geometry on this 2D-view plane...

### 25.1.3 DISPLAY FOR "FULL" 2D-VIEWS OR CONTOUR-2D-VIEWS OF THE GEOMETRY

- The keyword GRAPH indicates that a "full" 2D-view is requested (it is the same type of 2D-view that we can obtain for the drawing of equi-importance, for example). The volumes are colored.
- the keyword GRAPH_CONTOURS indicates that we are asking for a 2D-view with only the geometry contours. On the screen, we see white contours on a black background, in Postscript files we see black contours on a white background. On the screen, the contours are plotted point by point through two scannings (one vertically and one horizontally).
- Superimposed on these 2D-view are the points showing the collisions sites.

Restrictions:

- One of the two keywords GRAPH or GRAPH_CONTOURS must be present.
- Several 2D-views may be requested using several GRAPH or GRAPH_CONTOURS commands. However, in a COLLISION_SITES command, the 2D-view type must be the same (we cannot mix GRAPH and GRAPH_CONTOURS).

The data in the GRAPH or GRAPH_CONTOURS blocks are of the same type as the data used for the drawing of equi-importance maps:

- **Px Py Pz** indicates the coordinates (in the geometric frame) of the point in the bottom left part of the display window ;
- **Lu and Lv** indicate this window dimensions ;
- <AXIS> is X for a yPz window, Y for a xPz window, and Z for a xPy window;
- Several graphs may be requested. Flag is 0 if a Postscript file is not required. In this case, a window is displayed and if the user *a posteriori* decides to dump the associated Postscript file, he can do it with the **Shift-P** command in the graphical window. Otherwise Flag is 1 and there is no display window but the associated Postscript file is dumped.
- The option FILE allows for the user to specify the associated file name. By default, PostScript files are named "collisions_sites_N.ps", otherwise they are named "filename_N.ps" where *N* is the number of the graph. They are numbered in the order in which the graphs are requested.

The following keywords (DISCARD, SCORE, SCORE_SEGMENTS, SOURCE, NB_COLLISION, GRID and COLOR) are optional. They allow for additional conditions on the projected points and additional display options useful for more detailed analyses.

### 25.1.4 OPTION DISCARD

The optional keyword DISCARD enables to avoid the display of the collision sites of the nb(batches) first batches. This can be useful :

- in SHIELDING mode : setting nb(batches) to 5 enables to discard the 5 first batches during which adjustments of importance maps are performed by the code (simulation with an importance map non fixed group to group),
- in CRITICALITY mode : nb(batches) can be set to the number of batches needed for the convergence of the source.

Using the option DISCARD enables to simulate more particles (since the collision sites of the first batches are not stored), cf. sub-section 25.3. This option is compatible with options SOURCE, SCORE and SCORE_SEGMENTS detailed in the following sub-sections.

### 25.1.5 OPTIONS SCORE AND SCORE_SEGMENTS

The SCORE and SCORE_SEGMENTS options enable to display the whole particle history (birth site, collision sites and boundary condition sites) only for particles eventually contributing to the scores.

- The SCORE option displays only points (projections of the collision sites) and is compatible with COLOR and DISCARD options. When this option is used simultaneously with the SOURCE option (see sub-section 25.1.6), only source sites of contributing trajectories are displayed. When it is used simultaneously with the NB_COLLISION option and "0" as nb(collisions) (see sub-section 25.1.7), displayed points are source sites of contributing trajectories and production sites of secondary particles which eventually contribute to the scores.
- The SCORE_SEGMENTS option also displays segments between the different sites. It can be used with the DISCARD option and must be used with the GRAPH_CONTOURS keyword (see sub-section 25.1.3). When using the SCORE_SEGMENTS option, the COLOR option is not allowed. The user must keep in mind that among displayed *trajectories*, if all points are not projected on the 2D-view window (regardless its dimensions and the height parameter), some of the displayed segments may then not be real trajectory parts of the simulation because two non-consecutive sites may be joined.

### 25.1.6 OPTION SOURCE

This option enables the display of source sites instead of collision sites. It is also compatible with DISCARD and COLOR options. When this option is used simultaneously with the SCORE option, only source sites of contributing trajectories are displayed. Other options of this chapter are incompatible. The SOURCE option is also available simultaneously with the EXPLOITATION mode of Green's function files (see chapter 35) and can thus help to analyze the contents of Green's function files.

### 25.1.7 OPTION NB_COLLISION

This option enables the display of sites of the nb(collision)-th collisions only. The score estimator (see chapter 15) must be TRACK, SURF or COLL. When this option is used with "0" as nb(collision) and simultaneously with the SCORE option, displayed points are source sites of contributing trajectories and production sites of secondary particles which eventually contribute to the scores. Other options of this chapter are incompatible.

### 25.1.8 OPTIONAL DEFINITIONS FOR AN ENERGY GRID AND RELATED COLORS

The keywords GRID and COLOR are optional; however, COLOR is defined only if GRID is defined.

#### 25.1.8.1 Definition of an energy grid, keyword GRID :

The definition of a grid in the COLLISION_SITES command is made in exactly the same way as in the GRID_LIST and VARIANCE_REDUCTION commands. Remember that the list of energies has to be monotonic decreasing and that the number of bounds is the number of groups plus one.
The grid may or may not be same as that of the biasing grid. The advantage in defining the grid here is that it can exist even if no variance reduction technique is defined.
Some further indications regarding the aspect and the number of graphics :

- if GRID is not defined, we only have a graphic for all the energies of the simulation;
- if GRID is defined but COLOR isn't defined, we have a graphic for each group of the grid;
- if GRID and COLOR are defined, we have only one graphic and the colors of the points are defined by the user for each group.

#### 25.1.8.2 Color definition, keyword COLOR :

The number of colors defined here, nb(colors), has to be lower or equal to the total number of groups in the grid (nb(bounds)-1). If it is strictly lower, it means that the user has used the same color for several groups.
The choice of the colors is the same as for the optional definition of the colors of volumes for the geometry.
The possible values for <COLOR(i)> are as follows: BLACK, BLUE, BROWN, CORAL, CYAN, GOLD, GRAY, GREY, GREEN, MAGENTA, NAVY, ORANGE, PINK, PLUM, RED, SALMON, VIOLET, WHITE, YELLOW.
Each color name (nb(colors) altogether) is followed by the number of groups to be displayed using this color and their list. The number of the groups is taken between 1 and nb(bounds)-1. These numbers match the order of user's input into the GRID command.

- When COLOR is defined, all the groups must have a color (whether or not it is shared by other groups).
- If COLOR is not defined, collisions sites will be **dark** points for GRAPH and **red** points for GRAPH_CONTOURS.

## 25.2 ORDER OF APPARITION FOR THE DISPLAY WINDOWS

The display windows appear on the screen **at the end of the simulation** and in the following order:

- exterior loop : for each particle type corresponding to a `COLLISION_SITES` command;
- intermediary loop : for each 2D-view type defined by the `GRAPH` (or `GRAPH_CONTOURS`) command;
- interior loop : for each group of the associated `GRID` (if defined and if `COLOR` is not).

## 25.3 NUMBER OF PARTICLES IN THE SIMULATION

By default, when the `COLLISION_SITES` command is used, the maximum allowed number of particles for the simulation is 1000 in the `SIMULATION` command of the data files.
The data fields `BATCH` (number of simulated batches) and `SIZE` (number of particles for each batch) must verify :
**BATCH $\star$ SIZE $\leq$ 1000**
We will use, for example, a maximum of 10 batches with 100 particles.

When the option `DISCARD` is used, the maximal number of simulated batches can then be increased by the number nb(batches) discarded within this graphical command block : we will use, for example, a maximum of 10 + nb(batches) batches with 100 particles.

The simulation is performed with a small number of particles in order to obtain a reasonable number of points to store for the display. However, if the user wants to obtain collisions sites for more particles, the following keyword is added to the `SIMULATION` command:

**MAXP_FOR_COLLISION_SITES** m

The threshold will then be m instead of 1000 particles (consequently, the user will have to manage his free memory space).

## 25.4 NUMBER OF PROJECTED POINTS

The number of projected points on the display window appears in the execution window when each graphic is drawn. It should be compared, for example, to the collision number indicated in the result file of the simulation (before the printout of the scores).

# Chapter 26

# THE OUTPUT FILE

## 26.1 PRINTING OUT THE OUTPUT FILE

The TRIPOLI-4® output file consists of:

- the name of the dictionary and the data file;
- an echo of the input, for reference;
- if a variance reduction technique is used, intermediate results on cross-sections and $k$ coefficients;
- features of the source;
- the average number of collisions per particle with the associated standard deviation, given batch by batch;
- the requested results with the required frequencies.

## 26.2 WHEN A VARIANCE REDUCTION TECHNIQUE IS USED

If a variance reduction technique is used, collapsing of the cross-sections of the isotopes making up the composition is carried out for each composition. Each time the names of the processed isotopes are given, followed by the matrix of the collapsed scattering cross-sections (upper triangular, as only one group is retained for the thermal domain) for the material and the type of particle under consideration (neutron or photon). The matrix elements are macroscopic scattering cross-sections.

Example (with 4 energy groups):

```
DUMP CONDENSED MATRIX FOR MATERIAL: CONCRETE

PARTICLE: NEUTRON

6.96297e-02 1.03105e-01 1.40319e-02 1.41036e-03
0.00000e+00 2.75304e-01 1.05449e-01 7.52780e-03
```

```
0.00000e+00 0.00000e+00 5.32216e-01 1.69207e-01
0.00000e+00 0.00000e+00 0.00000e+00 1.29906e+00
```

The corrected transport cross-sections (total cross-section minus the product of the scattering cross-section and the cosine of the average deviation angle) are given by biasing group (in decreasing energy).

Example (with 4 energy groups):

```
transport corrected cross-sections:  2.09351e-01 3.89181e-01 7.01549e-01 1.84914e+00
```

With automatic biasing, the following lines indicate, only for information, the values of the biasing coefficient $k$, calculated using the Placzek equation, for the different values of β (0, 0.1, 0.2, 0.5, and 1.). The value of $k$ is calculated on the basis of the first detector only. The last value of $k$ printed corresponds to the actual choice of the value of β made by the user.

Example

```
placzeck k values , beta = 0.00000e+00
1.10537e-01 5.52686e-02 2.76343e-02 4.14515e-02

placzeck k values , beta = 1.00000e-01
1.45290e-01 1.51002e-01 2.26503e-01 3.39755e-01

placzeck k values , beta = 2.00000e-01
1.64969e-01 1.97704e-01 2.96556e-01 4.44834e-01

placzeck k values , beta = 5.00000e-01
1.88416e-01 2.61919e-01 3.92878e-01 5.89317e-01
number of ponderation groups fixed :  1

placzeck k values , beta = 1.00000e+00
1.88416e-01 2.82624e-01 4.23936e-01 6.35903e-01
number of ponderation groups fixed :  1

placzeck k values , beta = 1.00000e+00
1.88416e-01 2.82624e-01 4.23936e-01 6.35903e-01
number of ponderation groups fixed :  1
```

The value of $k$ must not exceed 0.9 times the value of the total cross-section. If solving the Placzek equation gives an excessively large value, k is set to 0.9 times the total cross-section. When this is done, the code prints a message indicating the number of transport groups for which k has thus been chosen (the line `number of ponderation groups fixed:  1` in the example).

If the monitoring option is selected (the default case), an estimate of splitting (with standard deviation) and the normalization factor for the biasing groups are supplied. In all cases, biasing statistics are indicated, at least for the first batch (for the first ones, in the case where a variance reduction technique is specified).

The number of collisions in the detector cell is also provided. If it is too low, the biasing parameters need to be changed. Important: numbering begins at 0 here.

The biasing statistics indicate the number of particles in each group, as well as the ratio of the weight and the weight imposed by the importance maps (with associated standard deviation). The normalization factor is used when there is a weighting factor to be applied.

Example

```
estimation of splitting on 5 trajectories:
mean:  6.00000e-01 sigma_n = 6.00000e-01
..........
edition statisque de ponderation
ponderation group:  0
total population:  0.00000e+00
weight discrepancy:  0.00000e+00
sigma:  0.00000e+00
..........
number of collision in cell of detector 0:  3
group:  0 normalisation factor = 1.00000e+00
..........
```

Note: If, for a material, the name begins with `VOID`, the value of $k$ is set at half the smallest value of the other materials.
In the event of problems with biasing, it is possible, as a last resort, to only consider a single group, with an assigned coefficient $k$. In such cases, biasing must be made manually (`DATA`).

## 26.3 OUTPUTS RELATING TO SOURCES

The printed output file gives the following quantities for each source:

- energy intensity integrated in accordance with the definition;
- energy intensity integrated accounting for restrictions concerning the domain due to the energy cut-offs `Einf` and `Esup`: the source no longer emits beyond [`Einf`, `Esup`];
- angular intensity;
- time intensity;
- geometrical intensity;
- total intensity integrated (product of partial intensities), initially without energy bounds;
- then true total intensity integrated (with energy restrictions [`Einf`, `Esup`]).

The code then indicates the intensity of the source, possibly imposed with `INTENSITY`, then the intensity of the source accounting for any global norm (imposed by `NORM`).
Finally, the total intensity as the sum of the intensity of all the sources is supplied (in particles, after the words GLOBAL SIMULATION INTENSITY).

In the result part of the output file the code is also displaying the source intensity depending on the mode of operation (mono-processor or parallel) :

- mono-processor : the code issues a statement “RESULTS ARE GIVEN FOR SOURCE INTENSITY" then the value of the source intensity
- parallel : the simulators do not send this information to the collector, therefore the collector is indicating : “RESULTS ARE GIVEN FOR SOURCE INTENSITY : unavailable”, but it is still possible to find the intensity in each simulator output file (GLOBAL SIMULATION INTENSITY).

## 26.4 OUTPUTS FOR EACH BATCH

This relates to the statistics on the collision, the number of particles, and the number of secondary particles if any.
Statistics on “mean number of deposited” are also given for each batch in case of use of the standard variance reduction technique (i.e., the exponential transform) during the simulation. This information gives the mean, on the source particles of the current batch, of all splitting operations for the particles and their progeny coming from splitting as well.

## 26.5 COMPLETE OUTPUTS

Complete outputs corresponding to responses are supplied at the requested frequency. Each time, every score is printed, with a list of its attributes (name of response, type of particles, and estimator). The score is a value integrated over the energy group. The standard deviation is given with the score.
The results are given in decreasing energy group order. Only results in groups whose values are not zero are given unless the execution option `-a` is specified (cf. chapter 1) .
The 99.8% confidence interval is indicated with its minimum value (lower) and maximum value (upper) in the case of a bootstrap calculation (in which the normal (Gaussian) distribution hypothesis is not applied).

## 26.6 CRITICALITY CALCULATIONS

In criticality mode, if the code has to produce 0.5 neutrons, the calculation algorithm actually produces 1 neutron with a probability of 0.5.
`"KEFF at step"` first gives the *keff* tallied by the KSTEP estimator. This printout takes the `DISCARD` option into account. It appears before all result edition (just before source intensity).
Then, combined *keffs* are given after the response part. These *keffs* take into account the `DISCARD` option too.
The code prints the values of *keff* (indicating the number of batches discarded) with three estimators:

- the KSTEP estimator, which accounts for the neutrons produced by the code at each stage;

- the KCOLL estimator, which accounts for the average number of neutrons that the code should produce (estimation based on the collision);
- the KTRACK estimator, which accounts for the average number of neutrons that the code should produce (estimated using the path).

First the code writes the *keffs* alone, then edition is done combining them two by two. Then, the code gives the result of all three *keffs* combined ("full combined estimator"). All the combined estimators take into account the inter-estimator correlations to produce an "almost unbiased" mean and standard deviation. In some conditions (for example with highly correlated estimators), the estimation of the combined estimator is numerically dangerous. In this case, the output of the value is disabled and replaced by a "not yet converged" output.
Eventually, the code produces the three previous types of *keffs* with an automatic discard estimation. This corresponds to the historic edition of *keffs*.
All *keff* estimators should be statistically equivalent.
Remark: refer to the `DISCARD` option of chapter 22 for more information regarding these estimators.

## 26.7 SUB-CRITICALITY CALCULATIONS

The results are given in the same manner for a sub-critical calculation with sources.
Care is nevertheless required: $1 \cdot 10^{-11}$ must be used as the lower energy value of the simulation. Otherwise, the two estimators KSTEP and KCOLL will not give the same value. This is because KSTEP, which uses the neutrons monitored by the code, will undergo the energy cut-off, whereas KCOLL and KTRACK, which account for the expected neutrons, will not be affected by the cut-off.

# Chapter 27

# STORING PARTICLES

This command is used to store the characteristics (energy, position, direction, weight) for the particles crossing a given list of boundaries.

- For a given dump file, the user can request a list of boundaries.
- An additional keyword is required in order to set a maximum for the size of the dump file in Mega bytes.

The syntax is as follows:

```
STORAGE
      nb(items-stored)
            filename(1)
            LIMIT_SIZE nb_Mo(1)
            <PARTICLE>
            E_max E_min
            < FILTER > VOLU LIST nb_1(volu)
                  volume(1) volume(2) ... volume(nb_1)
             FRONTIER LIST nb_1(boundaries)
                  num_1,1(vol_1) num_1,1(vol_2)
                  ...
                  num_1,i(vol_1) num_1,i(vol_2)
                  num_1,nb1(vol_1) num_1,nb1(vol_2)
            ...
            filename(n)
            LIMIT_SIZE nb_Mo(n)
            <PARTICLE>
            E_max E_min
            < FILTER > VOLU LIST nb_n(volu)
                  volume(1) volume(2) ... volume(nb_n)
             FRONTIER LIST nb_n(boundaries)
                  num_n,1(vol_1) num_n,1(vol_2)
                  ...
                  num_n,i(vol_1) num_n,i(vol_2)
                  num_n,nbn(vol_1) num_n,nbn(vol_2)
END_STORAGE
```

where:

- `nb(items-stored)` is the number of files to be created;
- `filename(i)` is the beginning of the name of a series of dump files: the name filename(i) is suffixed by "_1_.stock". When this dump file is written, and once the size of nb_Mo(i) has been reached, the code automatically generates a new dump file with the name: filename(i), suffixed by "_2_.stock", and so forth, as many times as necessary in order to store all of the particles during the simulation.
- `nb_Mo(i)` the maximum size for a dump file, given in Mega bytes.
- `<PARTICLE>` is the type of particles to be stored (currently, only `NEUTRON` and `PHOTON` are available). The option `ALL_PARTICLE` may be used in order to store both the neutrons and photons in the same file). The particles are only written for the interval [Emax,Emin];
- `<FILTER>` is optional and may have the values `FILTER_OR`, `FILTER_AND`, `FILTER_SEQ`, `FILTER_TARGET`. The filter is thereafter defined by inputting a list of volume numbers. The different possible filters are defined in the same way as for the scores (see chapter 15 for information on scores).

- `num(vol1)` and `num(vol2)` are the volumes that define the boundary in the direction from 1 toward 2 for the file to be created.

The file contains the following data in the given order:

- a header showing the boundary list, the energy bounds $E_{max}$ and $E_{min}$, the type of information to be stored for each particle, as well as the general simulation data (number of batches, size of the batches, type of particles, intensity of the source, and the simulation mode ("`SHIELDING`", "`CRITICALITY`" or "`FIXED_SOURCES_CRITICALITY`")).
- markers showing the beginning and end of each batch: "`BEGIN_OF_BATCH` i" and "`END_OF_BATCH` i" followed by the statistic weight of the batch: "`WEIGHT_OF_BATCH` $W_b$".
- the particles are stored between these batch markers. For each particle stored, the following information is given:
  - the name of the particle,
  - the energy $E$ in MeV,
  - $(x, y, z)$, defining the position,
  - $(u, v, w)$, defining the direction,
  - $W_t$, the weight of the particle.

Note : the format of this file has changed over the TRIPOLI-4 versions.

It must also be noted that the storage of particles may also be requested using the `SCORE` command in the case of a `FLUX_PT` ("next-event" type) estimator on a target surface (see chapter 15).
These outputs can be used to generate sources for a calculation with another code or with TRIPOLI-4 (in the case of a source resulting from source files for the global / local zooming approach, see paragraph 12.6.2 of the chapter on sources).

# Chapter 28

# DEFINING PERTURBATIONS

## 28.1 GENERAL

Apart from the usual scores, it is possible to calculate scores for materials with different compositions (although the nuclei must be the same).
A number of perturbations can be requested in the same calculation. Theses perturbations can be calculated either by the correlated sampling method (keyword `PERTURBATION`) or by the Taylor series approach (keyword `TAYLOR_PERTURBATION`).

The syntax of the command is as follows:

**PERTURBATION_LIST**
**nb(perturb)**
**PERTURBATION**
*Data on perturbation 1*
**END_PERTURBATION**
**TAYLOR_PERTURBATION**
*Data on perturbation 2*
**END_TAYLOR_PERTURBATION**
**...**
**PERTURBATION**
*Data on perturbation* `nb(perturb)`
**END_PERTURBATION**
**END_PERTURBATION_LIST**

where `nb(perturb)` is the number of perturbations.

## 28.2 DATA ON A PERTURBATION

A perturbation is characterized by:

- a type (see below);

- a list of volumes with an identical composition, which corresponds to the composition to be perturbed (only these volumes are affected by the perturbation);
- perturbation data (multiplication factors);
- a list of the scores for which outputs are required (neutron scores only can be requested);
- a mode used to output the results (value of score or difference compared to the unperturbed score).

The TRIPOLI-4® code can calculate two types of perturbed scores:

- perturbations for given concentrations,
- perturbations for total density of materials.

It is not possible to mix density and concentration perturbations in a single perturbation.
The point-based estimator `FLUX_PT` is not compatible with any perturbation.

## 28.3 DENSITY AND CONCENTRATION PERTURBATIONS

For perturbations with the correlated sampling method (default), the syntax is as follows:

```
PERTURBATION
      <CONCENTRATION_OR_DENSITY> nb(compo)
            name_compo(1) VOLUME LIST nb(volu)
                  num_volu(1) ... num_volu(n)
             FACT fact(1)... fact(n)
            ...
            name_compo[nb(compo)] VOLUME LIST nb(volu)
                  num_volu(1) ... num_volu(n)
             FACT fact(1)... fact(n)
      SCORE LIST nb(scores)
            num_score(1) ... num_score(n)
      MODE DELTA or MULTI
END_PERTURBATION
```

while for the Taylor series approach, the syntax is:

```
TAYLOR_PERTURBATION
      <CONCENTRATION_OR_DENSITY> nb(compo)
            name_compo(1) VOLUME LIST nb(volu)
                  num_volu(1) ... num_volu(n)
             FACT fact(1)... fact(n)
            ...
            name_compo[nb(compo)] VOLUME LIST nb(volu)
                  num_volu(1) ... num_volu(n)
             FACT fact(1)... fact(n)
      ORDER order
      SCORE LIST nb(scores)
            num_score(1) ... num_score(n)
      MODE DELTA or MULTI
END_TAYLOR_PERTURBATION
```

where:

- `<CONCENTRATION_OR_DENSITY>` is `CONCENTRATION` for concentration perturbations or `DENSITY` for density perturbations;
- `nb(compo)` is the number of compositions to be perturbed;
- `name_compo(i)` is the name of the material of the `i`th perturbation;
- `nb(volu)` is the number of volumes associated with the composition;
- `num_volu(1) ...num_volu(n)` is a list of the volumes affected by the perturbation;
- `fact(1)...fact(n)` is a list
  - with one multiplication factor for each nucleus in the case of a concentration perturbation: for three nuclei, list `1.1 0.9 1.0` means that the concentration of the first nucleus defined is increased by 10 per cent, that of the second one reduced by 10 per cent, and that of the third remains unchanged.
  - a single element for a density perturbation: the multiplication factor for the perturbation (1.1 means that the density is increased by 10 per cent).
- `order` is the order of the Taylor series (1 or 2).
- `nb(scores)` and {`num_score(i)`} are, respectively, the number and indices for the scores (in the input order of the `SCORE` command, counting from 1) for which the outputs are required;
- line `MODE DELTA or MULTI` means that differential results are requested (difference relative to the unperturbed score) `DELTA` instead of absolute results `MULTI`.

In criticality mode, available only for the correlated sampling method, the code calculates the perturbed values for *keff*. If there are no options, this corresponds to a criticality calculation, and only the perturbed *keff* values are calculated.

**Important:**
At the present time, the concentration perturbation can only relate to one material (`nb(compo)=1`).
The concentration perturbation is not available for a composition using natural elements, i.e. with the -NAT or -NAT-SPLIT suffix.

## 28.4 PARTIAL CROSS-SECTION PERTURBATION

### 28.4.1 SYNTAX

It is possible to perturb the cross-sections of all interaction types (elastic, discrete inelastic, etc.). These perturbations can be defined for several energy ranges, i.e., in an energy grid, and, as such, can differ by group. However, the perturbations of differential cross-sections (with respect to anisotropy or energy transfer) are not (yet) implemented. These perturbations are qualified as deterministic as opposed to a random perturbation of these cross-sections (see next section) to allow for uncertainty concerning the cross-sections.
Since the code deals with point-wise cross-sections, the perturbed cross-sections are defined by multiplication coefficients applied to the original cross-sections. It is important to guarantee consistency (the total cross-section must be equal to the sum of the cross-sections). If, for example, an absorption cross-section is increased (for example, $n \rightarrow \gamma$), it is possible to accordingly increase the total cross-section or, conversely, to accordingly reduce the cross-sections of other interactions (for example a $n \rightarrow p$) in order to maintain consistency in the cross sections. These carry-over rules are left to the discretion of the user and are given in a file whose path is included in the perturbation command.
The syntax is as follows:

```
PERTURBATION
    DETERMINISTIC
        NUCLEI nb(nuclei)
            nb_compo(nucleus-1)
                name_compo(1,1) VOLU LIST nb(volu)
                    num_volu(1) ... num_volu(n)
                ...
                name_compo[nb(compo),1] VOLU LIST nb(volu)
                    num_volu(1) ... num_volu(n)
             NUCLEUS name_nucleus(1)
             DATA name(file-1)
            ...
            nb_compo(N)
                name_compo(1,N) VOLU LIST nb(volu)
                    num_volu(1) ... num_volu(n)
                ...
                name_compo[nb(compo),N] VOLU LIST nb(volu)
                    num_volu(1) ... num_volu(n)
                name_nucleus(1) DATA name(file)
             NUCLEUS name_nucleus(N)
             DATA name(file-N)
    SCORE LIST nb(scores)
        num_score(1) ... num_score(n)
    MODE DELTA
END_PERTURBATION
```

The definitions of the terms are as follows:

- `DETERMINISTIC` is a keyword specifying deterministic perturbations.
- `NUCLEI nb(nuclei)` corresponds to the number of nuclei to be perturbed (two in the example below involving iron-56 and iron-54).
- Then, a list of the perturbations is given (for each nucleus). The following are indicated for each: the number of compositions (here, the two compositions that contain iron-56 for the first perturbation), the names of the compositions (`IRON_BULK` and `IRON`), the volumes associated with the compositions (18 in the case of `IRON_BULK` and 2 in that of `IRON`), the name of the nucleus (after the keyword `NUCLEUS`), a data file defining carry-over between the sections (after the keyword `DATA`).
- Then a list of the perturbed scores is given with the quantity and the numbers of the scores (16 scores in this case). Even for a criticality calculation, at least one score must be specified.
- Then, the perturbation mode (absolute value by default);

Example:

```
PERTURBATION
DETERMINISTIC  NUCLEI 2
2
IRON_BULK VOLUME LIST 18
8 20 21 22 23 24 25 255 26 27 28 29 30 31 32 33 34 35

IRON VOLUME LIST 2
9 12
NUCLEUS FE56 DATA Tripoli4/Test/Pertu/Det/fe56


2
IRON_BULK VOLUME LISTE 18
8 20 21 22 23 24 25 255 26 27 28 29 30 31 32 33 34 35
IRON VOLUME LIST 2
9 12
NUCLEUS FE54 DATA Tripoli4/Test/Pertu/Det/fe54

SCORE LIST 16 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
END_PERTURBATION
```

### 28.4.2 THE CARRY-OVER FILE

Here is an example of a carry-over file.

```
O16 300 endfb6

3 1.0000E-05 2.40000E+06 2.90000E+06 20.E+06

PERTU 2 2 102


CARRY_OVER 1 51


1.  0
1.  0

1.01 0
1.  0
```

```
1.01 0.1
1.  0
```

The file contains five different pieces of information, described below:

- The first line contains three fields that respectively specify the perturbed nucleus, the temperature of the nucleus, and the library from which the nucleus is derived.
- The next two lines contain the energy grid used for defining the perturbation.
  Important:
  - when defining the grid, the number of groups is given (and not the number of bounds to be read), then the bounds of the grid.
  - the energy must be in eV and not in MeV.
- The word PERTU precedes the list of the perturbed interactions. It is followed by the number of perturbed interactions, then by the number used by NJOY to encode each interaction. Important: these codes are not those used for referencing interactions in scores, but those given in the NJOY manual [35]. In the example, two interactions are perturbed, corresponding to elastic scattering (NJOY number 2) and $n \rightarrow \gamma$ capture (NJOY number 102).
- The keyword `CARRY_OVER` (required even if carry-over is not used) here defines carry-over relating to the NJOY 51 interaction code (discrete inelastic level 1). When carry-over is not used, one must nevertheless give an interaction code (for example 51), but then all carry-overs should be zero.
- Finally, there are the carry-overs. They are given for each group in the order in which the grid is given at the top of the file, then, for each perturbed interaction, in the order in which it is given after keyword PERTU. The factor by which the original cross-section is multiplied must be specified in order to yield the perturbed cross-section and the extent to which the perturbation is carried over to interaction 51. Thus:
  - In the first group, no perturbations apply to the cross-sections (factor of first column equal to 1) and the carry-overs are zero.
  - In the second group, the elastic scattering cross-section is increased by 1% and there is a zero carry-over to discrete inelastic level 1, thus the increase in the scattering cross-section is reflected in the total. Radiative capture is unchanged.
  - In the last group, the elastic scattering cross-section is increased by 1%. But 10% of the cross-section increase is removed from the discrete inelastic level 1, and the remaining 90% is carried over to the total which is increased accordingly.

Note: If a carry-over is too large, it may result in negative cross-sections. In such cases, the code breaks and displays a message indicating the energy and cross-sections involved. In such cases, it is necessary to change the carry-over or the perturbation.

| Interaction | Code | Particle |
|---|---|---|
| **Elastic scattering** | 2 | neutron |
| **Discrete inelastic and continuous scattering** | 4 | neutron |
| **n→ 2n** | 16 | neutron |
| **n→ fission** | 18 | neutron |
| **Level 1 discrete inelastic scattering** | 50+1 | neutron |
| **Level i discrete inelastic scattering** | 50+i | neutron |
| **Level 40 discrete inelastic scattering** | 90 | neutron |
| **Continuous inelastic scattering** | 91 | neutron |
| **Absorption. Photon emission** | 102 | neutron |
| **Absorption. Proton emission** | 103 | neutron |
| **Absorption. Alpha emission** | 107 | neutron |

Table 28.1: NJOY interaction codes for carry-over file

## 28.5 COVARIANCE PERTURBATIONS

The syntax is as follows:

```
PERTURBATION
      COVARIANCE
            NUCLEI nb(nuclei)
                  nb_compo(nucleus-1)
                        name_compo(1,1) VOLUME LIST nb(volu)
                              num_volu(1) ... num_volu(n)
                        ...
                        name_compo[nb(compo),1] VOLUME LIST nb(volu)
                              num_volu(1) ... num_volu(n)
                   NUCLEUS name_nucleus(1)
                   DATA name(file-1)
                  ...
                  nb_compo(N)
                        name_compo(1,N) VOLUME LIST nb(volu)
                              num_volu(1) ... num_volu(n)
                        ...
                        name_compo[nb(compo),N] VOLUME LIST nb(volu)
                              num_volu(1) ... num_volu(n)
                        name_nucleus(1) DATA name(file)
                   NUCLEUS name_nucleus(N)
                   DATA name(file-N)
      SCORE LIST nb(scores)
            num_score(1) ... num_score(n)
      MODE DELTA
      PERTU_PERIOD nb(batch)
END_PERTURBATION
```

Where:

- `COVARIANCE` is a keyword indicating covariance perturbations;
- `NUCLEInb(nuclei)` is the number of nuclei to be perturbed (2 in the example below involving iron-56 and iron-54);
- Then a list of the perturbations is given (for each nucleus). For each, the following is given: the number of compositions (here the two compositions that contain iron-54 for the first perturbation), the name of the compositions (`IRON_BULK` and `IRON`), the volumes associated with the composition (18 in the `IRON_BULK` case and 2 in the `IRON` case), the name of the nucleus (after keyword `NUCLEUS`), and the covariance data file (after the keyword `DATA`).
- Then a list is given of the perturbed scores with the number and the indices for the scores (16 scores in this case);

- Then the perturbation mode (absolute value by default);
- Then the sampling period (as a number of batches).

Example:

```
PERTURBATION
COVARIANCE NUCLEI 2


2
IRON_BULK VOLUME LIST 18
8 20 21 22 23 24 25 255 26 27 28 29 30 31 32 33 34 35

IRON VOLUME LIST 2
9 12
NUCLEUS FE56 DATA Test/Pertu/Covar/fe56.cov.user.data.2


2
IRON_BULK VOLUME LISTE 18
8 20 21 22 23 24 25 255 26 27 28 29 30 31 32 33 34 35
IRON VOLUME LIST 2
9 12
NUCLEUS FE54 DATA Test/Pertu/Covar/fe54.cov.user.data.2

SCORE LIST 16 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
PERTU_PERIOD 400
END_PERTURBATION
```

# Chapter 29

# STANDARD PERTURBATION THEORY

By combining the general framework of the Standard Perturbation Theory with Monte Carlo estimators for the adjoint flux in criticality calculations, it is possible to estimate (without approximations) first-order reactivity perturbations and k-eigenvalue sensitivity coefficients to nuclear data [84]. The standard perturbation theory (SPT) [2] expresses first-order reactivity perturbations and first-order sensitivities of $k$ to variations in the system parameters on the basis of the critical forward flux $\varphi$ and the critical adjoint flux $\varphi^\dagger$, both taken in the reference configuration. The critical adjoint flux is estimated by using the neutron importance function, which can computed by resorting either to the IFP (Iterated Fission Probability) or a history-based method (Wielandt or Super-History), over a user-defined number of latent generations (the IFP 'cycles'). For typical systems, a cycle length of 20 is sufficient in order to attain the convergence of the importance function within the critical simulations. The choice between IFP or history-based methods relies on a trade-off between memory usage and computation time, see the comment section 29.2.4. It is worth noting that the adjoint weighted methods are mutually exclusives. Scores related to the Standard Perturbation Theory are available in `CRITICALITY` simulation mode only.

## 29.1 REACTIVITY PERTURBATIONS

The reactivity of a system containing fissile materials is defined as $\rho = (k-1)/k$, where $k$ is the fundamental eigenvalue (i.e., the effective multiplication coefficient). It is often interesting to compute the variation $\Delta\rho$ corresponding to a perturbation of the nuclear data, including concentrations, densities, cross sections and average fission yield. According to SPT, it can be shown that the first-order reactivity variation due to a system perturbation is given by

$$\Delta\rho = \frac{\left\langle \varphi^\dagger, \left(\Delta\mathcal{S} + \frac{1}{k}\Delta\mathcal{F} - \Delta\mathcal{C}\right)\varphi \right\rangle}{\left\langle \varphi^\dagger, \mathcal{F}\varphi \right\rangle}, \tag{29.1}$$

where $\Delta\mathcal{S} = \mathcal{S}^* - \mathcal{S}$, $\Delta\mathcal{C} = \mathcal{C}^* - \mathcal{C}$ and $\Delta\mathcal{F} = \mathcal{F}^* - \mathcal{F}$ are the variations in the scattering, collision and fission operators, respectively [2]. Here

$$\mathcal{S}f = \int \Sigma_s(r, E' \to E, \Omega' \to \Omega) f(r, E', \Omega')\, \mathrm{d}E'\mathrm{d}\Omega' \tag{29.2}$$

is the scattering operator, $\mathcal{C} = \Sigma_t$ is the collision operator, and

$$\mathcal{F} f = \frac{\chi(E)}{4\pi} \int \nu(E')\Sigma_f(r,E') f(r,E',\Omega')\mathrm{d}E'\mathrm{d}\Omega' \tag{29.3}$$

is the fission operator of the integro-differential form of the $k$-eigenvalue equation.
For the reactivity change due to a perturbation in the collision operator, we have

$$\Delta\rho_{\mathcal{C}} = \frac{\langle \varphi^\dagger, \Delta\mathcal{C}\varphi \rangle}{\langle \varphi^\dagger, \mathcal{F}\varphi \rangle}, \tag{29.4}$$

where $\Delta\Sigma_t$ is the change in the total cross section due to the system perturbation. The contribution to a reactivity change due to a variation in the scattering operator is given by

$$\Delta\rho_{\mathcal{S}} = \frac{\langle \varphi^\dagger, \Delta\mathcal{S}\varphi \rangle}{\langle \varphi^\dagger, \mathcal{F}\varphi \rangle}, \tag{29.5}$$

where $\Delta\Sigma_s$ is the variation of the scattering cross section due to the system perturbation. For a perturbation in the fission operator, the corresponding reactivity variation is given by

$$\Delta\rho_{\mathcal{F}} = \frac{\langle \varphi^\dagger, \frac{1}{k}\Delta\mathcal{F}\varphi \rangle}{\langle \varphi^\dagger, \mathcal{F}\varphi \rangle}. \tag{29.6}$$

Prompt and delayed fission contributions are singled out by separately computing the prompt $\Delta\rho_{\mathcal{F}_p}$ and delayed $\Delta\rho_{\mathcal{F}_d}$ fission terms. The total reactivity change is finally provided by

$$\Delta\rho = \Delta\rho_{\mathcal{S}} + \Delta\rho_{\mathcal{F}_P} + \Delta\rho_{\mathcal{F}_d} - \Delta\rho_{\mathcal{C}}. \tag{29.7}$$

A refinement of the first-order formulation would include a perturbed denominator in Eq. 29.1 [2], namely,

$$\Delta\rho = \frac{\langle \varphi^\dagger, \left(\Delta\mathcal{S} + \frac{1}{k}\Delta\mathcal{F} - \Delta\mathcal{C}\right)\varphi \rangle}{\langle \varphi^\dagger, \mathcal{F}_*\varphi \rangle}. \tag{29.8}$$

In order to take into account the perturbed fission operator $\mathcal{F}_*$ at the denominator of Eq. 29.8, a correction factor can be included in the Monte Carlo estimators for Eqs. 29.4, 29.5 and 29.6.

### 29.1.1 ITERATED FISSION PROBABILITY SYNTAX

Adjoint-weighted first-order reactivity perturbations $\Delta\rho$ can be applied to concentrations, densities, cross sections and/or fission neutron yields (prompt, delayed and/or total). Such calculations are introduced by defining an additional block in the input file, by the keyword `IFP_PERTURBATION_LIST` . The general syntax using the IFP method is the following:

```
IFP_PERTURBATION_LIST
        <NB_PERTU>
        PERTU_TYPE_1
        <NB_COMPO_1>
        <COMPO_NAME_1>
        VOLUME LIST <NB_VOLU_1> <VOL_1_1> ... <VOL_1_N>
        <DATA_1>
        ...
        PERTU_TYPE_M
        <NB_COMPO_M>
        <COMPO_NAME_M>
        VOLUME LIST <NB_VOLU_M> <VOL_M_1> ... <VOL_M_N>
        <DATA_M>

        IFP_CYCLE_LENGTH <CYCLE_LENGTH>
        <REACTIVITY_CONTRIBUTIONS>
END_IFP_PERTURBATION_LIST
```

where DATA denotes additional information required for each perturbation. The perturbation list includes the number of perturbations to be computed and, for each of them, the perturbation type, the number of material compositions where each perturbation is applied, the composition names, and the associated list of volumes, and the additional information.

The perturbation type can be:

- CONCENTRATION for reactivity perturbations due to variations in the nuclide concentrations, and in this case the additional data must be provided in the form

  ```
  FACT <COEFF_1> ... <COEFF_P>
  <ORDER>
  ```

  i.e., we assign the perturbation coefficients for each nuclide in the composition. For instance, a factor of 1.01 means that the concentration for the given nuclide is perturbed by 1 percent upwards. A factor of 1 means that the nuclide is not perturbed. The last keyword ORDER is used to specify whether the user wants to compute the first order perturbation (as in Eq. 29.1), in which case the keyword is FIRST_ORDER, or the refined perturbation including the perturbed fission denominator (as in Eq. 29.8), in which case the keyword is PERTURBED_DENOMINATOR;

- DENSITY for reactivity perturbations due to variations in the material composition density, and in this case the additional data must be provided in the form

  ```
  FACT <COEFF>
  <ORDER>
  ```

i.e., we assign the perturbation coefficient for the composition density. For instance, a factor of 1.01 means that the density for the given composition is perturbed by 1 percent upwards. The last keyword ORDER is used to specify whether the user wants to compute the first order perturbation (as in Eq. 29.1), in which case the keyword is FIRST_ORDER, or the refined perturbation including the perturbed fission denominator (as in Eq. 29.8), in which case the keyword is PERTURBED_DENOMINATOR;

- SECTION for reactivity perturbations due to variations in the neutron partial cross sections, and in this case the additional data must be provided in the form

```
NUCLEUS <NUC_NAME> REACTION <REACTION_CODE>
ENERGY <NB_GROUPS> <E_min> ... <E_i>  ... <E_max>
FACT <COEFF_group_1> ... <COEFF_group_NB>
<ORDER>
```

i.e., we assign first the nucleus and the reaction to be perturbed, then the energy range of the perturbations (possibly decomposed over several energy groups), and finally the perturbation coefficients for each group. The reaction index is given by using the usual Tripoli-4 codes. For instance, a factor of 1.01 in the energy group $[E_3, E_4]$ means that the selected cross section for the given nucleus is perturbed by 1 percent upwards only in the chosen energy interval $E_3 < E < E_4$. The last keyword ORDER is used to specify whether the user wants to compute the first order perturbation (as in Eq. 29.1), in which case the keyword is FIRST_ORDER, or the refined perturbation including the perturbed fission denominator (as in Eq. 29.8), in which case the keyword is PERTURBED_DENOMINATOR;

- NU_FISSION for reactivity perturbations due to variations in the fission neutron yield, and in this case the additional data must be provided in the form

```
NUCLEUS <NUC_NAME> <NU_TYPE>
ENERGY <NB_GROUPS> <E_min> ... <E_i>  ... <E_max>
FACT <COEFF_group_1> ... <COEFF_group_NB>
<ORDER>
```

i.e., we assign first the (fissile) nucleus and the type of fission neutron yield, then the energy range of the perturbations (possibly decomposed over several energy groups), and finally the perturbation coefficients for each group. The fission neutron yield <NU_TYPE> can be PROMPT, TOTAL or DELAYED. For DELAYED, we can further specify:

```
DELAYED TOTAL
```

in order to perturb the total $\nu_d$, or

```
DELAYED FAMILY <FAMILY_NUMBER>
```

in order to perturb only the $\nu_d^i$ pertaining to a specific precursor family $i$. The other parameters are defined as above. The last keyword ORDER is used to specify whether the user wants to compute the first order perturbation (as in Eq. 29.1), in which case the keyword is FIRST_ORDER, or the refined perturbation including the perturbed fission denominator (as in Eq. 29.8), in which case the keyword is PERTURBED_DENOMINATOR.

Finally, the IFP cycle length must be chosen, in order to adjust the convergence of the neutron importance function. A value of 20 is typically recommended.

The optional keyword REACTIVITY_CONTRIBUTIONS can be used in order to decompose each perturbation sought by the user with respect to the collision term, the scattering term, and the prompt and delayed fission terms, as given in Eq. 29.7.

We provide hereby an example corresponding to a few representative perturbations:

```
IFP_PERTURBATION_LIST

    6

DENSITY 1
IRON VOLUME LIST 3 2 5 7
FACT 1.05
FIRST_ORDER

DENSITY 1
WATER VOLUME LIST 1 4
FACT 0.95
PERTURBED_DENOMINATOR

CONCENTRATION 1
FUEL VOLUME LIST 2 1 6
FACT 1.1 1.10 1.0
PERTURBED_DENOMINATOR

SECTION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
REACTION 33
ENERGY 1  1.E-11  1.964033E+01
FACT 1.05
FIRST_ORDER
```

```
NU_FISSION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
DELAYED FAMILY 5
ENERGY 1  1.E-11  1.964033E+01
FACT 0.99
PERTURBED_DENOMINATOR

NU_FISSION 1
MF VOLUME LIST 1 1
NUCLEUS U235
TOTAL
ENERGY 1  1.E-11  1.964033E+01
FACT 0.60
PERTURBED_DENOMINATOR

IFP_CYCLE_LENGTH 10

END_IFP_PERTURBATION_LIST
```

### 29.1.2 WIELANDT SYNTAX

The syntax for the Wielandt history-based method is very similar to the IFP one, where keywords of the IFP are switched to Wielandt ones. The keywords `IFP_PERTURBATION_LIST` , `END_IFP_PERTURBATION_LIST` and `IFP_CYCLE_LENGTH` can be replaced respectively by `WIELANDT_PERTURBATION_LIST`, `END_WIELANDT_PERTURBATION_LIST` and `WIELANDT_CYCLE_LENGTH` to trigger the perturbation using the Wielandt method to compute the adjoint function. Otherwise, the whole syntax is the exact same as for IFP perturbations. It is worth noting that computing adjoint weighted quantities using the Wielandt method requires that the Wielandt method is triggered in the SIMULATION block and that the expected length of the Wielandt subcritical fission chain is greater (or equal) to the cycle length used for computing adjoint weighted data.
Below is an example using the Wielandt method to a few representative perturbations:

```
WIELANDT_PERTURBATION_LIST
    6
DENSITY 1
IRON VOLUME LIST 3 2 5 7
FACT 1.05
FIRST_ORDER

DENSITY 1
WATER VOLUME LIST 1 4
FACT 0.95
```

```
PERTURBED_DENOMINATOR

CONCENTRATION 1
FUEL VOLUME LIST 2 1 6
FACT 1.1 1.10 1.0
PERTURBED_DENOMINATOR

SECTION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
REACTION 33
ENERGY 1  1.E-11  1.964033E+01
FACT 1.05
FIRST_ORDER

NU_FISSION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
DELAYED FAMILY 5
ENERGY 1  1.E-11  1.964033E+01
FACT 0.99
PERTURBED_DENOMINATOR

NU_FISSION 1
MF VOLUME LIST 1 1
NUCLEUS U235
TOTAL
ENERGY 1  1.E-11  1.964033E+01
FACT 0.60
PERTURBED_DENOMINATOR

WIELANDT_CYCLE_LENGTH 10

END_WIELANDT_PERTURBATION_LIST
```

### 29.1.3 SUPER-HISTORY SYNTAX

The syntax for the Super-History based method is very similar to the IFP one, where keywords of the IFP are switched to Super-History ones. The keywords `IFP_PERTURBATION_LIST`, `END_IFP_PERTURBATION_LIST` and `IFP_CYCLE_LENGTH` can be replaced respectively by `SUPER_HISTORY_PERTURBATION_LIST`, `END_SUPER_HISTORY_PERTURBATION_LIST` and `SUPER_HISTORY_CYCLE_LENGTH` to trigger the perturbation using the Super-History method to com-

pute the adjoint function. Otherwise, the whole syntax is the exact same as for IFP perturbations. It is worth noting that computing adjoint weighted quantities using the Super-History method requires that this method is triggered in the SIMULATION block and that the number of latent generations in the Super-History is greater (or equal) to the cycle length used for computing adjoint weighted data.
Below is an example using the Super-History method to a few representative perturbations:

```
SUPER_HISTORY_PERTURBATION_LIST
    6
DENSITY 1
IRON VOLUME LIST 3 2 5 7
FACT 1.05
FIRST_ORDER

DENSITY 1
WATER VOLUME LIST 1 4
FACT 0.95
PERTURBED_DENOMINATOR

CONCENTRATION 1
FUEL VOLUME LIST 2 1 6
FACT 1.1 1.10 1.0
PERTURBED_DENOMINATOR

SECTION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
REACTION 33
ENERGY 1  1.E-11  1.964033E+01
FACT 1.05
FIRST_ORDER

NU_FISSION 1
FUEL VOLUME LIST 2 1 6
NUCLEUS U235
DELAYED FAMILY 5
ENERGY 1  1.E-11  1.964033E+01
FACT 0.99
PERTURBED_DENOMINATOR

NU_FISSION 1
MF VOLUME LIST 1 1
NUCLEUS U235
TOTAL
```

```
ENERGY 1  1.E-11  1.964033E+01
FACT 0.60
PERTURBED_DENOMINATOR

SUPER_HISTORY_CYCLE_LENGTH 10

END_SUPER_HISTORY_PERTURBATION_LIST
```

## 29.2 K-EIGENVALUE SENSITIVITY COEFFICIENTS

According to SPT, the first-order sensitivities $S_{x^j}$ of the $k$-eigenvalue to a nuclear data parameter $x^j$ (including cross sections, average fission yield, fission spectra, and scattering distributions) are given by

$$S_{x^j} \equiv \frac{x^j}{k}\frac{\delta k}{\delta x^j} = \frac{\left\langle \varphi^\dagger, \left(\frac{\delta \mathcal{S}}{\delta x^j} + \frac{1}{k}\frac{\delta \mathcal{F}}{\delta x^j} - \frac{\delta \mathcal{C}}{\delta x^j}\right)\varphi \right\rangle}{\left\langle \varphi^\dagger, \frac{1}{k}\mathcal{F}\varphi \right\rangle}, \tag{29.9}$$

where $\delta\mathcal{S}$, $\delta\mathcal{F}$ and $\delta\mathcal{C}$ are the infinitesimal changes in the scattering, fission and collision operators, respectively, due to a corresponding change in the generic nuclear data parameter $x^j$ for the nuclide $j$ [2].
In practice, the energy-resolved sensitivity coefficients are actually integrated over energy intervals, i.e.,

$$S_{x^j,g} = \int_{E_{g-1}}^{E_g} S_{x^j}(E)dE, \tag{29.10}$$

where $E_{g-1}, E_g$ are the lower and upper bounds, respectively, of the energy group $g$.
For cross section and multiplicity sensitivities, the energy appearing in $S_{x^j,g}$ is that of the incident neutron. On the contrary, sensitivities to fission neutron spectra $\chi(E' \to E)$ are conventionally provided as a function of the emitted fission neutron energy. Fission spectra are usually isotropic, so that the associated sensitivity coefficients are not decomposed with respect to the angle. Finally, the sensitivity coefficients $S^{f^j}_{g',g,m}$ for scattering transfer functions $f_s(r, E' \to E, \mu)$ will be computed with respect to the emitted and incident neutron energies and to the direction cosine $\mu$.
Fission spectra and scattering transfer functions are normalized probability densities. The standard SPT estimators for Monte Carlo criticality calculations do not take into account the normalization of $\chi$ and $f_s$, and lead to the so-called unconstrained sensitivity coefficients. It is actually possible to enforce the normalization constraint on the Monte Carlo estimators, which leads to the so-called constrained sensitivity coefficients

$$\begin{aligned} \hat{S}^{\chi^j}_{g',g} &= S^{\chi^j}_{g',g} - \chi^j_{g',g}\sum_g S^{\chi^j}_{g',g} \\ \hat{S}^{f^j}_{g',g,m} &= S^{f^j}_{g',g,m} - f^j_{g',g,m}\sum_g\sum_m S^{f^j}_{g',g,m}. \end{aligned} \tag{29.11}$$

In principle, different normalization schemes might be conceived for $\chi^j_{g',g}$ and $f^j_{g',g,m}$. We have used the most

common definition, for which the coefficients $\chi^j_{g',g}$ and $f^j_{g',g,m}$ appearing in Eqs. 29.11 read

$$\chi^j_{g',g} = \frac{\int_{E_g}^{E_{g+1}} dE \int_{E_{g'}}^{E_{g'+1}} dE' \chi^j(E' \to E)\nu(E')\sigma_f(E')}{\int_0^{\infty} dE \int_{E_{g'}}^{E_{g'+1}} dE' \chi^j(E' \to E)\nu(E')\sigma_f(E')} \tag{29.12}$$

$$f^j_{g',g,m} = \frac{\int_{\mu_{m-1}}^{\mu_m} d\mu \int_{E_g}^{E_{g+1}} dE \int_{E_{g'}}^{E_{g'+1}} dE' f^j(E' \to E,\mu)\nu_s\sigma_s(E')}{\int_{-1}^{1} d\mu \int_0^{\infty} dE \int_{E_{g'}}^{E_{g'+1}} dE' f^j(E' \to E,\mu)\nu_s\sigma_s(E')}, \tag{29.13}$$

where $\nu_s$ is the neutron multiplicity for scattering reactions. The refinement of the incident neutron energy mesh plays an important role for constrained sensitivity coefficients: the use of coarse meshes for the incident energy might lead to non-negligible biases due to integration errors [84]. This issue can be overcome by resorting to finer meshes, which in turn demands an increased memory burden.

In the exact same way as for perturbations, the adjoint function can be estimated using either an IFP method or a History-based method, either Wielandt or Super-History.

### 29.2.1 ITERATED FISSION PROBABILITY SYNTAX

Adjoint-weighted first-order sensitivity coefficients $S_{x^j,g}$ of the $k$-eigenvalue with respect to nuclear data can be applied to cross sections, fission neutron yields (prompt, delayed and/or total), fission spectra (prompt, delayed and/or total) and/or scattering transfer functions. Such calculations are introduced by defining an additional block in the input file, by the keyword `IFP_SENSITIVITY_LIST`. The syntax is the following:

```
IFP_SENSITIVITY_LIST
        <NB_SENSITIVITIES>

        SENSITIVITY_TYPE_1
        <NB_SENSITIVITIES_1>
        <DATA_1>
        ...
        SENSITIVITY_TYPE_M
        <NB_SENSITIVITIES_M>
        <DATA_M>

        IFP_CYCLE_LENGTH <CYCLE_LENGTH>
END_IFP_SENSITIVITY_LIST
```

where `DATA` denotes additional information required for each sensitivity coefficient. The sensitivity list includes the number of different sensitivity types to be computed and, for each of them, the sensitivity type and the number of sensitivities per each type.

The sensitivity type can be:

- SECTION for sensitivities to the neutron partial cross sections, and in this case the additional data for each sensitivity must be provided in the form

  ```
  NUCLEUS <NUC_NAME>
  REACTION <REACTION_CODE>
  ENERGY GRID <ENERGY_MESH_NAME>
  ```

  i.e., we assign first the nucleus and the reaction for which the sensitivity is computed, then the (incident) energy mesh over which the sensitivity coefficients are decomposed. The mesh for sensitivities must be defined in the usual mesh block and must cover the simulation domain for the neutrons. The reaction index is given by using the usual Tripoli-4 codes;

- NU_FISSION for sensitivities to the fission neutron yield, and in this case the additional data for each sensitivity must be provided in the form

  ```
  NUCLEUS <NUC_NAME>
  <NU_TYPE>
  ENERGY GRID <ENERGY_MESH_NAME>
  ```

  i.e., we assign first the (fissile) nucleus and the type of fission neutron yield, then the (incident) energy mesh over which the sensitivity coefficients are decomposed. The mesh must be defined in the usual mesh block and must cover the simulation domain for the neutrons. The fission neutron yield <NU_TYPE> can be PROMPT, TOTAL or DELAYED. For DELAYED, we can further specify:

  ```
  DELAYED TOTAL
  ```

  for the sensitivity to the total $\nu_d$, or

  ```
  DELAYED FAMILY <FAMILY_NUMBER>
  ```

  for the sensitivity to the $\nu_d^i$ pertaining to a specific precursor family $i$;

- CHI_FISSION for sensitivities to the fission spectra, and in this case the additional data for each sensitivity must be provided in the form

  ```
  NUCLEUS <NUC_NAME>
  <CHI_TYPE>
  ENERGY GRID <ENERGY_MESH_NAME>
  INCIDENT_ENERGY GRID <INCIDENT_ENERGY_MESH_NAME>
  <NORMALIZATION_FLAG>
  ```

  i.e., we assign first the (fissile) nucleus and the type of fission spectra $\chi$. The fission spectra <CHI_TYPE> can be PROMPT, TOTAL or DELAYED. For DELAYED, we can further specify:

```
DELAYED TOTAL
```

for the sensitivity to the total $\chi_d$, or

```
DELAYED FAMILY <FAMILY_NUMBER>
```

for the sensitivity to the $\chi_d^i$ pertaining to a specific precursor family $i$. The first energy mesh is given with respect to outgoing energies; the second is with respect to incident energies. The keyword <NORMALIZATION_FLAG> can be either <UNCONSTRAINED_COEFFICIENTS> for un-normalised sensitivity coefficients, or <CONSTRAINED_COEFFICIENTS> if the sum of the sensitivity coefficients must yield 1 (see Eq. 29.11). For constrained coefficients, the incident energy mesh should be sufficiently fine ;

- SCATTERING_TRANSFER_FUNCTION for sensitivities to the scattering transfer functions $f_s$, and in this case the additional data for each sensitivity must be provided in the form

```
NUCLEUS <NUC_NAME>
REACTION <REACTION_CODE>
ENERGY GRID <ENERGY_MESH_NAME>
INCIDENT_ENERGY GRID <INCIDENT_ENERGY_MESH_NAME>
COSINUS GRID <NB_POINTS> P1 ... PN
<NORMALIZATION_FLAG>
```

i.e., we assign first the nucleus and the (scattering) reaction for which the sensitivity is computed. The list of allowed diffusion reactions is provided in Tab. 29.1. The first energy mesh is with respect to outgoing energies; the second is with respect to incident energies; the third mesh is with respect to the scattering cosine, between $-1$ and 1. The keyword <NORMALIZATION_FLAG> can be either <UNCONSTRAINED_COEFFICIENTS> for un-normalised sensitivity coefficients, or <CONSTRAINED_COEFFICIENTS> if the sum of the sensitivity coefficients must yield 1 (see Eq. 29.11). For constrained coefficients, the incident energy mesh and the cosine mesh should be sufficiently fine.

- SECTION_DEFAULT for triggering the calculation of sensitivities with respect to a given set of cross section reactions (codes : 21, 28, 30, 31, 32, 52, 33) for all nuclides in the system.

```
SECTION_DEFAULT ENERGY GRID <ENERGY_MESH_NAME>
```

- NU_FISSION_DEFAULT for triggering the calculation of sensitivities with respect to fission nu (total, prompt and delayed) for all nuclides in the system.

```
NU_FISSION_DEFAULT ENERGY GRID <ENERGY_MESH_NAME>
```

Finally, the IFP cycle length must be chosen, in order to adjust the convergence of the neutron importance function. A value of 20 is typically recommended.

We provide hereby an example corresponding to a few representative sensitivities:

| NEUTRON DIFFUSION INTERACTIONS | CODE |
|---|---|
| **Elastic scattering** | 21 |
| **Discrete inelastic scattering** | 22 |
| **Continuous inelastic scattering** | 23 |
| **n** $\rightarrow N\cdot$**n** | 24 |
| **n** $\rightarrow$ **2n** | 242 |
| **n** $\rightarrow$ **3n** | 243 |
| **n** $\rightarrow$ **4n** | 244 |
| **n** $\rightarrow$ **2n** + $\alpha$ | 245 |
| **n** $\rightarrow$ **n** + **2**$\alpha$ | 272 |
| **n** $\rightarrow$ **n** + $\alpha$ | 25 |
| **n** $\rightarrow$ **n** + **p** | 26 |
| **Discrete and continuous inelastic scattering** | 28 |
| **Coherent elastic scattering** | 30 |
| **Incoherent elastic scattering** | 31 |
| **Incoherent inelastic scattering** | 32 |
| **Total neutron scattering** | 42 |
| **Other interactions (continuous melange)** | 105 |

Table 29.1: Allowed neutron interaction codes for sensitivities to the scattering transfer functions.

```
IFP_SENSITIVITY_LIST

4

NU_FISSION 2

NUCLEUS U235
PROMPT
ENERGY GRID DEC_33G

NUCLEUS U235
DELAYED TOTAL
ENERGY GRID DEC_33G

CHI_FISSION 2

NUCLEUS U235
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_33G
CONSTRAINED_COEFFICIENTS

NUCLEUS U235
```

```
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_1G
UNCONSTRAINED_COEFFICIENTS

SCATTERING_TRANSFER_FUNCTION 1

NUCLEUS U238
REACTION 2
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_1G
COSINUS GRID 1 -1.0 1.0
UNCONSTRAINED_COEFFICIENTS

SECTION 3

NUCLEUS U235
REACTION 104
ENERGY GRID DEC_1G

NUCLEUS U235
REACTION 33
ENERGY GRID DEC_33G

NUCLEUS U238
REACTION 52
ENERGY GRID DEC_33G

IFP_CYCLE_LENGTH 10

END_IFP_SENSITIVITY_LIST
```

### 29.2.2 WIELANDT SYNTAX

The syntax for the Wielandt history-based method is very similar to the IFP one, where keywords of the IFP are switched to Wielandt ones. The keywords `IFP_SENSITIVITY_LIST`, `END_IFP_SENSITIVITY_LIST` and `IFP_CYCLE_LENGTH` can be replaced respectively by `WIELANDT_SENSITIVITY_LIST`, `END_WIELANDT_SENSITIVITY_LIST` and `WIELANDT_CYCLE_LENGTH` to trigger the sensitivity calculations using the Wielandt method to compute the adjoint function. Otherwise, the whole syntax is the exact same as for IFP sensitivities. It is worth noting that computing adjoint weighted quantities using the Wielandt method requires that the Wielandt method is triggered in the SIMULATION block and that the expected length of the Wielandt subcritical fission chain is greater (or equal) to the cycle length used for computing

adjoint weighted data.
Below is an example using the Wielandt method to a few representative sensitivities:

```
WIELANDT_SENSITIVITY_LIST 5

SECTION_DEFAULT ENERGY GRID DEC_33G

SECTION 1
NUCLEUS U235
REACTION 104
ENERGY GRID DEC_238G

NU_FISSION_DEFAULT ENERGY GRID DEC_33G

CHI_FISSION 2

NUCLEUS U235
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_33G
CONSTRAINED_COEFFICIENTS

NUCLEUS U235
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_1G
UNCONSTRAINED_COEFFICIENTS

SCATTERING_TRANSFER_FUNCTION 1

NUCLEUS U238
REACTION 2
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_1G
COSINUS GRID 1 -1.0 1.0
UNCONSTRAINED_COEFFICIENTS

WIELANDT_CYCLE_LENGTH 10

END_WIELANDT_SENSITIVITY_LIST
```

### 29.2.3 SUPER-HISTORY SYNTAX

The syntax for the Super-History based method is very similar to the IFP one, where keywords of the IFP are switched to Super-History ones. The keywords `IFP_SENSITIVITY_LIST`, `END_IFP_SENSITIVITY_LIST` and `IFP_CYCLE_LENGTH` can be replaced respectively by `SUPER_HISTORY_SENSITIVITY_LIST`, `END_SUPER_HISTORY_SENSITIVITY_LIST` and `SUPER_HISTORY_CYCLE_LENGTH` to trigger the sensitivity calculations using the Super-History method to compute the adjoint function. Otherwise, the whole syntax is the exact same as for IFP sensitivities. It is worth noting that computing adjoint weighted quantities using the Super-History method requires that this method is triggered in the SIMULATION block and that the number of latent generations in the Super-History is greater (or equal) to the cycle length used for computing adjoint weighted data.
Below is an example using the Super-History method to a few representative sensitivities:

```
SUPER_HISTORY_SENSITIVITY_LIST 5

SECTION_DEFAULT ENERGY GRID DEC_33G

SECTION 1
NUCLEUS U235
REACTION 104
ENERGY GRID DEC_238G

NU_FISSION_DEFAULT ENERGY GRID DEC_33G

CHI_FISSION 2

NUCLEUS U235
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_33G
CONSTRAINED_COEFFICIENTS

NUCLEUS U235
TOTAL
ENERGY GRID DEC_33G
INCIDENT_ENERGY GRID DEC_1G
UNCONSTRAINED_COEFFICIENTS

SCATTERING_TRANSFER_FUNCTION 1

NUCLEUS U238
REACTION 2
ENERGY GRID DEC_33G
```

```
INCIDENT_ENERGY GRID DEC_1G
COSINUS GRID 1 -1.0 1.0
UNCONSTRAINED_COEFFICIENTS

SUPER_HISTORY_CYCLE_LENGTH 10

END_SUPER_HISTORY_SENSITIVITY_LIST
```

### 29.2.4 COMMENTS

Memory storage can become a serious issue for IFP-based sensitivity calculations: since IFP requires the storage of the entire neutron genealogy over the latent generations, the memory needs can dramatically increase when computing a huge amount of sensitivity coefficients for reactor configurations with a large number of neutrons per cycle [84]. This is especially true for the superposed-cycle implementation of the IFP method, for which an additional factor $M$ is required for storing the neutron generations in memory. We have implemented a history-based scoring strategy for sensitivity calculations with IFP. In order to reduce the dependence of the memory utilization on the number of simulated particles, only non-vanishing contributions to the bilinear form are recorded at collision events. This strategy is effective in reducing the memory footprint: the memory footprint is reduced from $O(M \cdot s)$ to $O(M \cdot c)$, where $s$ is the number of requested sensitivity coefficients and $c$ is the number of collisions per neutron generation. The average number of collisions per neutron generation is very small for fast systems (only 2.5 for Godiva), but even for thermal systems the memory gain is clear. The price to pay is a small bookkeeping overhead.

A way to circumvent the storage of the entire neutron genealogy over the latent generations is to actually simulate the progeny of the neutron over the number of latent generations cutting of the proportionality of the storage with respect to the number of neutrons in the batch. Both the Super-History and the Wielandt methods rely on simulating fission trees (a neutron and its progeny) over several generations to determine the next batch neutrons before simulating the next neutron of the current batch. It is this simulation characteristic that we are using to compute the adjoint function. Using these “history-based” approaches (Super-History or Wielandt) usually yields very few memory overhead independently of the number of simulated particles. However, the counterpart is the increase in computation time which is multiplied (on average) by the number of latent generations since we are simulating several generations in a single batch.

# Chapter 30

# GENERALIZED PERTURBATION THEORY

The Standard Perturbation Theory framework can be extended to several response functions using the Generalized Perturbation Theory

$$S_{\sigma_p}(R) = \frac{\sigma_p}{R}\frac{\partial R}{\partial \sigma_p} \tag{30.1}$$

$R$ being a response function. Typical formulations for $R$ include ratios of forward-weighted linear forms (such as reaction rates) and ratios of forward and adjoint-weighted bi-linear forms (such as kinetics parameters). A preliminary implementation of the GPT for reaction rate ratios is available in the version 12. These preliminary scores related to the Generalized Perturbation Theory are available in `CRITICALITY` simulation mode only with the SUPERHISTORY method.

## 30.1 REACTION RATE RATIO SENSITIVITY COEFFICIENTS

Let us consider a response function $R$ in the form of ratios of reaction rates, namely,

$$R = \frac{R_1}{R_2} = \frac{\langle \Sigma_1 \varphi \rangle}{\langle \Sigma_2 \varphi \rangle}. \tag{30.2}$$

Assuming first-order perturbations and neglecting higher-order contributions, the use of the Differential Operator Sampling method (DOS) coupled to the Super-History method leads to the following formulation at a given power iteration cycle:

$$\begin{aligned} S_{\sigma_p}(R) &= S_{\sigma_p}(R_1) - S_{\sigma_p}(R_2) \\ &= \frac{1}{R_1}\frac{1}{W}\sum_{n}^{neut}\sum_{c}^{coll} h_{n,c}\left(\frac{Q'}{Q} + \frac{h'_{n,c}}{h_{n,c}} + \frac{c'_{n,c}(R_1)}{c_{n,c}(R_1)}\right) c_{n,c}(R_1) \\ &- \frac{1}{R_2}\frac{1}{W}\sum_{n}^{neut}\sum_{c}^{coll} h_{n,c}\left(\frac{Q'}{Q} + \frac{h'_{n,c}}{h_{n,c}} + \frac{c'_{n,c}(R_2)}{c_{n,c}(R_2)}\right) c_{n,c}(R_2), \end{aligned} \tag{30.3}$$

where $Q$ denotes the fission source, $h_{n,c}$ the neutron life probability, $c_{n,c}(R_i)$ is the neutron random contribution to the reaction rate $R_i$, $i = \{1,2\}$, and $W$ is the total neutron weight at the beginning of a cycle. Primes

denote the scaled partial derivative of the quantity with respect to the nuclear data $\sigma_p$. The contributions of $h_{n,c}$ and $c_{n,c}(R_i)$ are expressed as:

$$\frac{h'_{n,c}}{h_{n,c}} = \frac{\chi'}{\chi} + \sum_{b}^{c} \left( \frac{T'}{T}(\mathbf{r}_b \rightarrow \mathbf{r}_{b+1}) + \frac{C'}{C}(E_b, \Omega_b \rightarrow E_{b+1}, \Omega_{b+1}) \right) \tag{30.4}$$

$$\frac{c'_{n,c}(R_i)}{c_{n,c}(R_i)} = \frac{\Sigma'_i}{\Sigma_i}, \tag{30.5}$$

where $\chi$ represent the fission spectrum, $T(\mathbf{r}_c \rightarrow \mathbf{r}_{c+1})$ is the flight operator between collision points $c$ and $c+1$ and $C(E_c, \Omega_c \rightarrow E_{c+1}, \Omega_{c+1})$ is the collision operator that changes the energy and direction of the neutron. All these random contributions are computed for neutrons in the last generation of the Super-History. The fission source derivatives are expressed as

$$\frac{Q'}{Q} = \sum_{j}^{ancestors} \left( \frac{h'_{j,c}}{h_{j,c}} + \frac{c'_{j,c}(k_{eff})}{c_{j,c}(k_{eff})} - \frac{k'_j}{k_j} \right), \tag{30.6}$$

where

$$\frac{c'_{n,c}(k_{eff})}{c_{n,c}(k_{eff})} = \frac{(\nu\sigma_f)'}{\nu\sigma_f}, \tag{30.7}$$

and can be computed using the sensitivities to the life probability of the ancestor and to the fission event that they induced, leading to the current neutron: this strategy was extensively described in Ref. [86]. Since Eq. (30.3) involves a difference of two first-order sensitivities, the terms $k'/k$ cancel out and their contribution can be ignored in the simulation. The fission source derivatives are computed in the generations preceding the last one.

### 30.1.1 SUPER-HISTORY SYNTAX

In the exact same manner as for the $k$-eigenvalue, first-order sensitivity coefficients of the reaction rate ratio $R$ with respect to nuclear data can be applied to cross sections, fission neutron yields (prompt, delayed and/or total), fission spectra (prompt, delayed and/or total) and/or scattering transfer functions. Such calculations are introduced by defining an additional block in the input file, by the keyword `GPT_SUPER_HISTORY_SENSITIVITY_LIST`. The syntax is the following:

**GPT_SUPER_HISTORY_SENSITIVITY_LIST**
<NB_SENSITIVITIES>

SENSITIVITY_TYPE_1
<NB_SENSITIVITIES_1>
<DATA_1>
...
SENSITIVITY_TYPE_M
<NB_SENSITIVITIES_M>
<DATA_M>

SUPER_HISTORY_CYCLE_LENGTH <CYCLE_LENGTH>
<NUMERATOR>
<DENOMINATOR>
**END_SUPER_HISTORY_SENSITIVITY_LIST**

where DATA denotes additional information required for each sensitivity coefficient. The sensitivity list includes the number of different sensitivity types to be computed and, for each of them, the sensitivity type and the number of sensitivities per each type. The sensitivities in DATA are requested using the exact same syntax as for the *k*-eigenvalue sensitivities.
Beyond the nuclear data for which the sensitivity coefficients are requested, the reaction rate ratio must be defined. It is performed using the GPT_NUMERATOR and GPT_DENOMINATOR keywords. These reaction rates can be defined using a nucleus, a reaction code and one (or a list of) volume(s) or composition(s). The selection of these parameters are performed in the following form :

```
NUCLEUS <NUC_NAME>
REACTION <REACTION_CODE>
* VOLU <id>
* COMPOSITION <id>
```

where <id> denotes either a single identifier or a list identifiers preceded of the LIST keyword and the number of identifiers. An example is provided below :

```
GPT_SUPER_HISTORY_SENSITIVITY_LIST 5

  SECTION_DEFAULT ENERGY GRID DEC_33G

  SECTION 1
  NUCLEUS U235
  REACTION 104
  ENERGY GRID DEC_238G

  NU_FISSION_DEFAULT ENERGY GRID DEC_33G
```

```
  CHI_FISSION 2

  NUCLEUS U235
  TOTAL
  ENERGY GRID DEC_33G
  INCIDENT_ENERGY GRID DEC_33G
  CONSTRAINED_COEFFICIENTS

  NUCLEUS U235
  TOTAL
  ENERGY GRID DEC_33G
  INCIDENT_ENERGY GRID DEC_1G
  UNCONSTRAINED_COEFFICIENTS

  SCATTERING_TRANSFER_FUNCTION 1

  NUCLEUS U238
  REACTION 2
  ENERGY GRID DEC_33G
  INCIDENT_ENERGY GRID DEC_1G
  COSINUS GRID 1 -1.0 1.0
  UNCONSTRAINED_COEFFICIENTS

  SUPER_HISTORY_CYCLE_LENGTH 10
  GPT_NUMERATOR NUCLEUS PU239 REACTION 33 COMPOSITION MOX
  GPT_DENOMINATOR NUCLEUS U235 REACTION 52 VOLU LIST 2 1 2
END_SUPER_HISTORY_SENSITIVITY_LIST
```

## 30.2 LIMITATIONS

The Generalized Perturbations Method implementation in the TRIPOLI-4 code is not yet a final implementation. It has been validated using comparisons with the SCALE6.2.4, APOLLO3 software and published results on several benchmarks. However, it holds several limitations :

- It can be used only with the Super-History simulation method (no IFP version exists for the version 12)
- Some response functions are not yet available such as flux, neutron production rate
- Reaction rates decomposed over an energy binning are not yet handled.
- Some derivatives may not work properly : for instance the sensitivity to a partial reaction of the reaction rate may be inaccurate
- No convergence indicator is provided yet

These limitations should be overcome in the next version.

# Chapter 31

# SPECIAL OUTPUTS

This command is used to output data intended to be displayed as plots or directly looked up by the user in a text format. The syntax is as follows:

```
      OUTPUT
            LANGUAGE  <language>
*           XVGR
                  Data for scores display
            END_XVGR
*           SECTION
                  Data for cross-sections display
            END_SECTION
*           INFO  NUCLEI
                  Data for info on nuclei
      END_OUTPUT
```

where: `<language>` is either `FRENCH` or `ENGLISH` (the default is an output in English).

## 31.1 XVGR PLOTS

The `XVGR` option is used to create plots from the code results in XVGR format. The syntax is as follows:

```
XVGR
        ENERGY_INF  E_inf
        ENERGY_SUP  E_sup
        TYPE  <type>
        EDITION  nb(edit)
        LETHARGY
        HISTO
END_XVGR
```

or:

```
XVGR
        < ENTROPY_TYPE >
END_XVGR
```

where:

- ENERGY_INF (optional) allows to set the lower bound $E_{inf}$ of x axis, default is $1 \cdot 10^{-11}$.
- ENERGY_SUP (optional) allows to set the upper bound $E_{sup}$ of x axis, default is $2 \cdot 10^{+01}$.
- TYPE (optional) allows to select the graphical interpolation type from LIN_LIN, LIN_LOG, LOG_LIN or LOG_LOG, default is LOG_LIN.
- EDITION (optional) allows to select a multiple of the printout frequency nb(edit), default is 1. Files are not created for each printout but only for a subset of one printout out of nb(edit). For example if the frequency output requested in the simulation block (see chapter 22) is 100 but no EDITION option is requested in the OUTPUT XVGR command block, then XVGR files are created every 100 batches. Now, if addionally an OUTPUT XVGR edition is requested and nb(edit) is set to 5, then XVGR files are created every 500 batches.
- LETHARGY (optional) produces score/lethargy values in XVGR files.
- HISTO (optional) produces XVGR files in histogram interpolation mode.
- <ENTROPY_TYPE> is either SHANNON or BOLTZMANN.

The code produces as many plots (which can be displayed with XVGR, XMGR, GRACE and XMGRACE) as there are spectrum printouts in the output file (**plot filenames** are attributed in the output file after the line "Next score corresponds to file").
In a criticality calculation, two files are created (for kcoll and kstep estimators), showing the variation in keff as a function of the number of batches (files are updated at each batch).
If entropy is asked, then one file is produced and updated at each batch (spectrum plots are not produced). This option does not work with angle or time grids.
It is also limited to flux, current and reaction rates response types.

## 31.2 CROSS-SECTION DATA AND PLOTS

The SECTION option enables to output neutron or photon macroscopic or microscopic cross-sections. The output values are those directly used by the code during the simulation, after the nuclear data reading and the initialization step of the code. The output format is a two-column ASCII format giving (energy, cross-section) couples. The cross-sections may then be plotted with XMGRACE or any other graphical tool. The syntax is as follows:

```
SECTION
        n
        <output_type_1> name_1 temperature_1
        ...
        <output_type_n> name_n temperature_n
        ENERGY_INF  E_inf
        ENERGY_SUP  E_sup
        NB_POINTS  nb(points)
        PARTICLE  <particle_type>
        UNITS  ENERGY  <unit>
END_SECTION
```

where:

- n is the number of cross-section outputs requested
- for each cross-section output i, <output_type_i> is either MATERIAL (for a macroscopic cross-section), NUCLEUS (for a total microscopic cross-section) or INTERACTION (for a microscopic cross-section of a given interaction). In this latter case, the keyword INTERACTION must be followed by the interaction code and the keyword NUCLEUS, before entering name_i. This interaction code is defined in the same way as for the reaction rates input data of the response types chapter (see section 13.2)
- name_i is the name of one of the compositions presents in the COMPOSITION command block (in case of a macroscopic cross-section) or the name of a nuclide present in one or several compositions (in case of a microscopic cross-section)
- temperature_i is an integer and is the temperature of the composition or nuclide
- $E_{inf}$ is the lower energy requested for the output of each cross-section (the default is 1e-11 MeV for neutrons and 1e-3 MeV for photons)
- $E_{sup}$ is the upper energy requested for the output of each cross-section (the default is 20 MeV)
- nb(points) is the total number of points requested for the output of each cross-section (the default is 500 000 points). Cross-sections are interpolated by the code at nb(points) regularly spaced energies between $E_{inf}$ and $E_{sup}$. For photon cross-sections, the user is advised to keep a minimum of 10 000 points because the interpolation of the evaluation is generally not linear-linear between points.

- <particle_type> is either NEUTRON or PHOTON (the default is NEUTRON)
- <unit> is either MEV for MeV or EV for eV (the default is MEV).

The five last lines of the OUTPUT SECTION command block are optional.
The names of the produced files are such as "xs_output_<NAME>_<TEMP>K.dat" for the MATERIAL and NUCLEUS cases, with <NAME> the name of the requested composition or nuclide and <TEMP> the requested temperature. The file names are such as "xs_output_<NAME>_<TEMP>K_int<NUM>.dat" for the INTERACTION case, with <NAME> the requested nuclide, <TEMP> the requested temperature and <NUM> the requested interaction code.
The user will take advantage of a log-log scale when displaying these data with a graphical tool.

## 31.3 INFORMATION ON SIMULATED NUCLEI

The INFO NUCLEI option enables to produce a specific text file containing a list of interactions simulated by the code for the different nuclides presents in the data file (needed for the transport or for fictive reaction rates tallies). The syntax is as follows:

**INFO NUCLEI**
**filename**

where filename is the name of the text file produced.

For each nuclide, a list of neutronic and/or photoatomic and photonuclear reactions is provided, depending on the types of simulated particles and simulation mode. Information on photonuclear reactions is given only if the PHOTO_NUCLEAR simulation mode is requested by the user (see chapter 22). The information consists of the interaction name, the TRIPOLI-4® interaction code (as given in the section 13.2), the MT number of the ENDF format (see reference [35]), the interaction threshold, and an indication of presence of any photon production data (for coupled neutron-gamma simulations only). This information comes from the nuclear data, as retrieved by TRIPOLI-4® for its simulation step.

## 31.4 T4G DEDICATED OUTPUTS

Two dedicated outputs are available for T4G: meshes and point events. Their format is easily readable and understandable by the user (similar to the Open DX general format from IBM, see reference [23]).

### 31.4.1 MESH OUTPUTS

Mesh outputs are

- Tabulated sources: STORE_IN_FILE <output_filename> just after the TABULATED keyword in the GEOMETRIC_DISTRIBUTION of the source block;

- Importance maps: `STORE_IN_FILE ASCIIG` (block `VARIANCE_REDUCTION`);
- Mesh results: `STORE_IN_FILE` (block `EXTENDED_MESH`), see also `NORMALIZE`.

Communications with the T4G tool are described in the chapter 38. It is however reminded that this tool is made aware of the mesh outputs by the mean of the <inputDeck>.meshes file using the following format

```
name <meshfile0>.general [frame <ox> <oy> <oz> <ux> <uy> <uz> <vx> <vy> <vz>]
name ...
```

### 31.4.2 POINT EVENT OUTPUTS

Point event outputs must be called from the `SIMULATION` block using the keywords

- Source points: `STORE_SOURCES_IN_FILE`;
- Collision points: `STORE_COLLISIONS_IN_FILE`;
- Leakage points: `STORE_LEAKAGE_IN_FILE`.

As the output size may be voluminous, dump is done in a per batch/cycle basis and can either be in ASCII or BINARY mode.

**STORE_XXX_IN_FILE**
**#ASCII**
**#BINARY**
***GENERAL***
**LIST <nbbatch> <batch_1> <batch_2> <batch_nbbatch>**

For use in T4G, point files must declared by the mean of the <inputdeck>.points file using the following syntax

```
name <pointfile0>.general
name ...
```

The current list of fields can be found in the corresponding header (`.general` file), for example:

```
field = x, y, z, ox, oy, oz, energy, simulation_weight, sid, pid, time, mxs, nbcoll,
        parttype, status, cellid, nucleus, interaction_type, sampled_fission
```

We can find the following quantities

- x, y, z, t, position and time coordinates of the event
- ox, oy, oz, direction coordinates of the event
- simulation_weight, statistical weight of the particle during the event
- sid, pid are the source particle identifier and the particle identifier

- mxs, macroscopic total cross section at the location and during the event (can be used to calculate a flux)
- cellid, volume identifier
- energy, nucleus, interaction_type, sampled_fission, should be self explanatory

# Chapter 32

# CONNECTION WITH THE DARWIN/PEPIN2 CODE

It is possible to prepare an input deck for the DARWIN-PEPIN2 code [99], in order to perform an activation calculation. Creation of a deck for fuel depletion calculations is not yet implemented.
The syntax is as follows:

```
PEPIN
        GRID name(grid)
        SCORE_MODE <MODE>
        HISTORY nb(steps)
                time(init) <UNIT>
                time(1) <UNIT> value(1)
                ...
                time(final) <UNIT> value(final)
        VOLUME nb(volu)
                num(vol1)...num(voln)
END_PEPIN
```

- The energy grid name must correspond to a known grid: "***3_GRS***", "***SAILOR***", "***APOLLO_99***", "***GAM2***", "***APOLLO_172***", "***APOLLO_281***" or "***TRIPOLI_315***".
- `ISOTOPES` means that the composition for INTERPEP is the one that was input in the TRIPOLI-4® data set. By default, the values for module INTERPEP are rebuilt according to the natural composition of the materials.
- `<MODE>` = `TRACK` or `COLL`, indicating the type of flux (volume related) for calculating file INTERPEP. This is because, in a simulation, it is possible to calculate the scores for a given volume using the "track length" estimator and the "collision" estimator. The keyword is used to indicate which type of estimator is retained (there must, therefore, be a score with this estimator) for use in the creation of the file for INTERPEP.

- `nb(steps)` is the number of steps (for burnup) plus the number of intersteps (relating to a zero flux).
- The initial time must be indicated, then the final time of the interval i and the value by which the flux will be multiplied during the interval, e.g.,

```
 0 SECO
30 SECO 1
 1 MINU 0
 3 MINU 4
```

  In this case, for the interval (0 - 30 s), the flux is multiplied by 1, during the interval (30 s - 1 min) by 0 (interstep), then by 4 for (1 - 3 min).
- The units `<UNIT>` possible are `SECO` (s), `MINU` (min), `HOUR` (h), `YEAR` (year).
- This is followed by the number of volumes for which results are required, and a list of them.
- A filename **interpep** and a filename **spallation** are created, and the outputs are given by second and by unit volume as the volumes are indicated in `VOLSURF` or calculated and allocated by `CALCULATION` (default 1).

# Chapter 33

# PARALLELISM

A parallel version of the TRIPOLI-4® code can be used. It is possible to use a number of separate machines for a single simulation, or to carry out parallel calculation on several processors on a massively parallel machine.

**IMPORTANT**: As of version 12, the TRIPOLI-4 code is launched in parallel using utilities as mpirun or srun (as MPI codes). See the example in the section 33.3.2 for the new launch procedure. To adapt with the parallel launching tool (mpirun, srun), the user will want to use the '-m' command line argument (see introduction in section 33.1).

## 33.1 INTRODUCTION

A parallel application is executed in the same way as a conventional one. The code manages execution of the different processes.

It is, however, also necessary to indicate:

- a parallelism file,
- the type of communication,

and, for this reason, the launch syntax for a calculation becomes:

**tripoli4**
-d <data_file>
-s <format>
-c <dictionary_file>
*-l <language>*
*-o <results_file>*
**-p <parallelism_file>**
**-t <communication_type>**
***-m <parallel_rank_variable_name>***
***-i <nb_processes>***
***-M <MTU_SIZE>***
***-S***
***-E***

Where:

- <parallelism_file> is the name of the file containing hosts to be used for parallelism and whose syntax is given in a following section (see 33.3).
- <communication_type> can now only be **bsd**. It means the communications are using the internal communication library coded with BSD TCP sockets.
- <parallel_rank_variable_name> is the name of the environment variable containing the rank of the parallel process. This variable depends on the utility used for launching TRIPOLI-44 in parallel. Usual launchers are mpirun or srun. Common names of this environment variable are : RMS_RANK, SLURM_PROCID, MPIRUN_RANK, OMPI_COMM_WORLD_RANK
- <nb_processes> sets the number of simultaneously initializing processes (which is currently 64 by default).
- MTU_SIZE is the maximum size in bytes of packets in TCP communication. This tuning is available only in **bsd** mode and can be useful to increase performance of parallel mode on computer clusters. The correct parameter is to be derived from the specifications of the network interfaces of the cluster. A too small value yields to parallelism low performance due to too much lag between communication packets. **Warning:** A too big value can lead to incorrect results or parallelism instability.
- option -S: dispatches equal active charge (*ie* same number of batches) among all simulators if possible. If the number of simulators divide the number of batches asked in the block SIMULATION, each simulator will have the same number of batches which is the required number of batches divided by the number of simulators. Otherwise, the first simulators will process one more batch. If DISCARD is asked, the same number of active (non-discarded) batches are dispatched among simulators. Using -S is not recommended in shielding mode with variance-reduction as a batch canceled due to too much splitting among the first 4 will lead to missing batches and thus to a dead lock (all processes blocked waiting for the others to make next action).

- option -E: the input deck is no longer given in simulator output files. However, the command line option -E is now available if the input deck is still needed.

The process executed by the user is called the **monitor**.
It is essential that the name of the monitor machine coincides with the name of the execution machine.
Its role is to monitor the proper execution of the parallel application.

Regarding the other processes, one of them, the **collector** (whose output is written in the file with the postfix **1th_task**), has the function of collecting all the results. The other processes, called **simulators** (whose output is written in the files with the postfixes **2th_task**, **3th_task**, etc.), carry out independent simulations which all start with the same data (that given in the input file), but use a different sequence of the random number generator. At the end of each batch, each simulator send its batch results to the collector. When the collector has collected the number of batches demanded by the user in the input file, it tells the monitor to stop all the simulators.
Care must be paid for criticality calculations, since the number of discarded batches indicated by the user in the input file applies to each simulator, while the total number of batches refers to the whole simulation (see complements in section 33.4).

## 33.2 LAUNCHING TOOLS

As of version 12, the TRIPOLI-4 code is launched in parallel using utilities as mpirun or srun (as MPI codes).

### 33.2.1 MPIRUN OR SRUN TOOLS

To use TRIPOLI-4 with mpirun or srun, you must first install it (if not available) on your Linux distribution using your package manager (apt, yum...). The specific tool man page will guide you for running the executable.

You must get the name of the rank environment variable and declare it to TRIPOLI-4 (using -m argument, see above). Usual names are:

- `SLURM_PROCID` for SLURM;
- `OMPI_COMM_WORLD_RANK` for OpenMPI;
- `MPIRUN_RANK` or `RMS_RANK`.

To help you finding this environment variable, you may use the command

```
mpirun -n 2 env | grep RANK
```

You must also prepare a graphe file coherent with the usage of the mpirun (or srun) tool. Especially, the number of hosts declared in the graphe file must be the same as the number of hosts asked to mpirun (or srun).

### 33.2.2 LPAR* TOOLS

lpar is a tool for launching in parallel. Since the version 12, the command name has to be changed to lparm as the default build of TRIPOLI-4 is suitable for use on large clusters (the underlying TRIPOLI-4 is using the mpirun tool or equivalent). It allows the user to run TRIPOLI-4 in parallel on a node with multiple CPUs without having to prepare the parallelism host file. The user must only provide the command line in mono-processor and the number of CPUs to be used for the calculation. It is greatly recommended, as a preliminary test, to process a few batches (3-5) in mono-processor.

Then, the calculation can be launched in parallel using the following command line

```
lparm -n <number_of_processors> <command_line_in_monoprocessor>
```

In the previous command, the exact same command as in mono-processor can be used (reducing error possibilities).

## 33.3 THE PARALLELISM FILE

The syntax of the file is as follows:

**PROCESS**
  **nb(process)**
    **process monitor <program> <machine(m)>**
    **process <name(collector)> <program> <machine(0)>**
    **process <name(simulator$_1$)> <program> <machine(1)>**
    **...**
    **process <name(simulator$_N$)> <program> <machine(N)>**
***RUN_DIRECTORY***
  ***name(directory)***
**GRAPH**
  **name(collector) < − > name(simulator$_1$)**
  **...**
  **name(collector) < − > name(simulator$_N$)**
**END**

where:

- <program> is the name of the executable (possibly with its path), tripoli4 for instance,
- <machine(m)> is the name of the machine on which the monitor is located,
- <name(collector)> and <machine(0)> relate to the collector,
- <name(simulator$_i$)> and <machine(i)> relate to the i-th simulator

- <name(directory)> is the directory in which the tasks are run (from which certain task files can be recovered. It can, therefore, be suitable to give the working directory, as the code uses $HOME by default).

### 33.3.1 HETEROGENEOUS CLUSTERS

It is possible to run the simulation on a cluster of completely heterogeneous computers, including hardware by SUN, HP, etc.). The monitor runs the batches on the different processors.

### 33.3.2 CALCULATION ON A MACHINE WITH A RESOURCE MANAGER

The process for using TRIPOLI4 on a machine with resource manager (for example SLURM) is quite similar. It is, however, necessary to create a parallelism file named graphe.save (for the launch procedure):

```
PROCESS <N>
process monitor      /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 1th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 2th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 3th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 4th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 5th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
process 6th_task     /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene
...
process <N-1>th_task /home/tripoli4/tripoli4.11/CODE/bin/irene/static_tripoli4  irene

GRAPH
1th_task <->  2th_task
1th_task <->  3th_task
1th_task <->  4th_task
1th_task <->  5th_task
1th_task <->  6th_task
...
1th_task  <->  <N-1>th_task
FIN
```

It should be noted that the name of the node on which each task is to run is not specified. It is the code that, depending on the number of nodes allocated to it by the launch shell, creates a filename graphe.port, which needs to be destroyed after execution, and an execution file (tache.csh):

```
#!/bin/sh
#BSUB -W 0:15
#BSUB -n 6

cd <working_directory>
```

```
# cleanup
\rm -f *.update graphe graphe.port last_simulation*

# new graph file done by script inigraphlsb
inigraphlsb $PROG

#execution
srun /home/tripoli4/tripoli4.9/CODE/bin/linux-intel-64/static_tripoli4 -d <data_file>\
                  -s NJOY -c <dictionary_file> -p graphe -t bsd

# cleanup
\rm -f *.update graphe graphe.port last_simulation*
```

All that is then required is to submit the task to the parallel machine:

```
bsub < tache.sh
```

#### 33.3.2.1 Results

At launch time, filenames

`last_simulation.<process_name>.res`

are created. It is in these files that are written any errors encountered during the initialisation phase of the parallel simulation.
When the simulation starts, other files are created

`<data_file>.<process_name>.res` .

For the collector, the file contains the results of the different printouts requested. For the simulators, the file contains the characteristics for the simulation for each batch processed. The printouts are created as the results are collected by the collector and as simulation proceeds for the simulators.
As for file `results_file` (see chapter 1), it contains the results of the monitor (not affected by the results of the simulation). It contains information on parallelisation, process initialisation, communication statistics, etc.

#### 33.3.2.2 Advices regarding execution

- Remember that the monitor machine must be the same one that launches the execution.
- The traffic (i.e., the number of messages exchanged), must not be prohibitively large. As the processors send a message to the collector at the end of each batch, it is better if the running time of each batch is not too short. In particular, the number of particles per batch must not be too small. Otherwise, the traffic can prevent communication and crash the machines involved in the simulation (the operating system buffers being full), and even bring the entire network down.

It is possible to use the PACKET_LENGTH option defined in chapter 22 in order to reduce the amount of communications.

- Moreover, it is highly recommended to launch exactly one process per processor. This also includes the monitor and the collector, even if they do not seem to use as much CPU resources as a simulator. They are communicating with all processes, so that too much delay receiving some messages can result in a loss of the whole calculation. Their role is critical for the computation.
- When launching several multi-processor calculations, it is highly recommended to run each case in separate directories otherwise that may lead to failure.

## 33.4 DIFFERENCES BETWEEN DISCARD PROCESSING IN MONO-PROCESSOR AND PARALLEL CALCULATIONS

Due to both theoretical and practical considerations, the discard processing in criticality between the mono-processor and parallel modes are different.

In the criticality mode, the whole calculation is a fix point algorithm initialized by the user (specifying the initial source). The first part of the calculation must dispatch the sources according to the fundamental mode of the critical Boltzmann equation. The accumulation of scores only starts after this first part is finished. A second reason to discard this part is for the Central Limit theorem. The Central Limit theorem allows to calculate the standard deviation of the mean and have confidence intervals. One hypothesis of this theorem is the stationarity of the series on which it applies.
Multiple discards are used depending on the aim. All are, of course, used to ensure the stationarity of the series of the results. However, they apply on different parts of the calculation:

- source convergence discard: either in manual mode (keyword DISCARD) or in automatic mode (no keyword DISCARD), it applies on the collecting of the scores. In mono-processor mode, no score except Keffs is calculated before the end of this part. In parallel, no scores at all is sent before the end of this part. For the automatic mode, when the fluctuations of the kcoll is considered stabilized, the sources are supposed to be converged and the message "kcoll is converged at batch" is displayed.
- discards on the series of results. Results consists of 2 kinds of scores which are processed differently. Mainly, we have Keffs and the other scores.
  - "Best discard" is used for Keffs. An evaluation is done on the first third of the simulation to find the "best discard" which will yield the minimal standard deviation.
  - DISCARD (or automatic discard) is used again for scores.

We gather in the following table the application of every discard depending on the calculation part.

| Calculation part | auto discard | manual discard | best discard |
|---|---|---|---|
| Source convergence | yes | yes | no |
| Fluxes, reaction rates... | yes | yes | no |
| Global scores (Beta, global, etc...) | yes | yes | no |
| Keffs | Mono/no | Mono/no | yes |
| | Para/yes | Para/yes | yes |

## 33.5 INCOMPATIBILITIES

The following options or commands are not available in parallelism mode:

- Option `-j` (NJOY, see chapter 1);
- `SOURCE_STORAGE` (see section 12.6.2);
- Green's function files exploitation (see chapter 35);
- `RANDOM` option with DRAND48_RANDOM random number generator: only the MT19937_RANDOM Mersenne Twister random number generator is allowed (see chapter 22).
- `GRAPH_CONTOURS` or `GRAPH_NO_CONTOUR` command in the `AUTO_COUPLING` option (see section 17.7);
- `GRAPH` option in `VARIANCE_REDUCTION` command (see section 17.8);
- Display of collisions sites (see chapter 25);
- The `IDT` block in the `VARIANCE_REDUCTION` block (see chapter 20);
- Command line option -S in shielding mode with variance reduction (see above);
- Entropy calculation (however, the user can alternatively study the source convergence in monoprocessor mode and use the result for its parallel run, see subsection 15.9.2).

# Chapter 34

# COLLISION FILES

This command makes it possible, during a simulation, to store as a binary file the attributes of particles, i.e., particle type (currently neutron or photon), the direction, time, and weight of particles passing through given volumes, the crossing of given boundaries or involvement in collisions in any of the fixed volumes.
Once the simulation is completed, the data stored can be read in order to rapidly calculate new scores. The scores must, of course, depend on only the information stored. For instance, if all the collision points in a volume have been stored, it is then possible, in a later phase, to calculate all the reaction rates by the collision estimator in the volume.
All types of collection and response functions can be reproduced from the dumped records. However, the scores using the sub-command FILTER, which enables only the particles which pass through certain volumes to be counted, are not compatible with the collision files. (It would be necessary to store the history of the particles, which would be costly in terms of memory and disk usage.)

## 34.1 CREATION OF A COLLISION FILE

The syntax is as follows for the collision file creation phase:

```
COLLISION_TAPE
        CREATION
                number of files
                name of file_1
                        <PARTICLE_TYPE>
        *               FRONTIER
                                nbzones
                                Emax(1) Emin(1) volume_1 volumebis_1
                                ...
                                Emax(nbzones) Emin(nbzones) volume_nbzone volumebis_nbzone
        *               VOLUME
                                nbzones
                                Emax(1) Emin(1) volume_1
                                ...
                                Emax(nbzones) Emin(nbzones) volume_nbzone
        *               CELL
                                nbzones
                                Emax(1) Emin(1) volume_1
                                depth_1 ni(1) nj(1) nk(1) ... ni(depth_1) nj(depth_1) nk(depth_1)
                                ...
                                Emax(nbzones) Emin(nbzones) volume_nbzones
                                depth_nbzones ni(nbzones) nj(nbzones) nk(nbzones) ...
                                        ni(depth_nbzones) nj(depth_nbzones) nk(depth_nbzones)

                name of file_2
                ...
                name of file_n
        END_CREATION
END_COLLISION_TAPE
```

The meanings of the fields are as follows:

- Number_of_files corresponds to the number of files to be created.
- <PARTICLE_TYPE> corresponds to the type of particles for which information is stored.
- This is followed by an indication of the type of geometrical zone affected by the storage, i.e., FRONTIER, VOLUME, and CELL.
- The number of geographical zones where information is stored.
- Emax and Emin correspond to the maximum and minimum energies for the information stored.

- A list is then given of the geometrical elements for which information is stored. In the case of boundaries, it should be borne in mind that each boundary is defined by a volume-number pair, and there is, therefore, an equal number of volume-number pairs to be defined as there are declared storage zones. In the case where the geometrical zone is a lattice cell, it is necessary to specify the volume number and then, as in the score command (see Section 15.7), it is necessary to specify the depth of the lattice one likes to access and, finally, the number of the cell concerned for each depth number.

The user can create as many files as desired.

## 34.2 USE OF A COLLISION FILE

The use of data stored in a file is introduced using the keyword EXPLOITATION.
The syntax is as follows:

```
COLLISION_TAPE
        EXPLOITATION
      number of files
            name of file 1
            ...
            name of file n
        END_EXPLOITATION
END_COLLISION_TAPE
```

The data are, therefore:

- the number of files to be used,
- a list of filenames to be used.

All other commands of the code, and their meanings, remain unchanged. In particular, the SCORE command can be used to obtain new reaction rates with little cost.

# Chapter 35

# GREEN'S FUNCTION FILES

## 35.1 OBJECTIVE

This feature allows for parametric studies where the aim is to determine the incidence of a source variation on the scores (for example, a parametric study related to a new loading of the core).
This feature also allows for the contribution of the source areas for a required response to be determined (in this case, there is no variation of the source). The goal is to obtain scores that are conditional to the creation of particles in these source areas.

## 35.2 PRINCIPLE

During a reference simulation performed once for all, "Green's functions" (or "transfer functions") are stored in files. These files are then re-used in a suitable way (renormalization of sources, reconstitution of scores) in order to process a different configuration than the reference configuration (first type of exploitation) or to re-express scores as conditional scores to the sources (second type of exploitation).

The denomination "Green's function" includes several pieces of information stored when a particle contributes to the score in the reference simulation:

1. "initial" information: the number of the source where the particle is created, coordinates, energy, statistical birth weight (removed since version 12), and source amplitude at the particle initial position and energy.
2. "final" information: particle energy and statistical weight when the particle contributes to the score in the required tallying area, its contribution (quantity $\chi$ defined in chapter 15 on scores, whose expression depends on the required type of estimator).

One interest of this approach is the reduction in the computation time (because the history of particles is not simulated during the exploitation phase ) with a satisfactory quality for the results. The drawback is the huge amount of storage volume required.

## 35.3 LIMITATIONS AND HYPOTHESES

1. Processed particles are either NEUTRONS or PHOTONS. Combined NEUTRONS-PHOTONS problems are also possible.
2. We must treat a SHIELDING or a FIXED_SOURCES_CRITICALITY type problem (CRITICALITY computations are excluded).
3. This feature can be used only for **tabulated sources**. Thus, analytic sources and point sources are excluded.
4. When using the EXPLOITATION mode, geometric data, composition, and biasing (if defined, in case of use of a variance reduction technique) have to be strictly the same as the ones defined during the CREATION phase.
5. Constraints on the possibilities of modification of the source : if we use a new source during the exploitation (second mode of use), only the geometric distribution intensity and/or the value of the coefficient COEFF can be modified. The spatial distribution of the source and the energetic, angular, and time distributions must remain identical. However, these constraints are not tested by the code during the exploitation phase. This is the user's responsability.
6. The allowed types of geometric elements for the tallying of scores are only VOLUMES or FRONTIERS (no points and lattice cells). During the exploitation of the files, the scores can be required by LIST and/or SUM; on the other hand, during the creation phase each file corresponds to only one VOLUME or FRONTIER.
7. Currently, the only allowed response during exploitation are fluxes, reaction rates, and the H*(10) dose equivalent rate (user can still use ANS77). Current requests are not possible.
8. Each created file corresponds to a useful tallying area and to a given type of estimator.
9. It is necessary to keep in mind the following **approximation**, in the case of an exploitation of the files with a change of source (second mode of use) : the transport in the modified source is supposed to be the same as in the original source.
10. The **biasing**, used during the exploitation phase with a change of source, is the same as the one used for the original source (the code doesn't need a second definition of the biasing during the exploitation phase, and doesn't check the existence or the coherence of the biasing). During the exploitation phase, the new source initialization, biased if there is biasing, is indeed computed with the biasing defined in the data file of the creation phase.
11. Finally, for this same exploitation mode, a part of the phase space that has been less fully explored during the original simulation will obviously give a poor precision in the exploitation phase. It is thus appropriate to use a biasing of the original simulation that is adapted as much as possible to the various cases of exploitation under investigation.
12. Starting from version 12, Green's function files are compatible with AMS variance reduction technique (see Chapter 18). It should be noted that a significant increase of the Green's function files size might

occur when AMS technique is used compared to the Green's function files size when INIPOND is used (see Chapter 17) for the same configuration. By construction of the AMS algorithm, more particles might, indeed, reach score zones and be stored in the Green's function files. The number of simulated particles in creation phase should therefore be chosen carefully.

13. Warning : the format for Green's function files was modified from version 5 to version 6 and from version 11 to version 12, the compatibility with the previous format is kept nevertheless. However, concerning version 4, starting from version 4.3, the format of these files is incompatible with earlier versions (4.2 to 4.0).

## 35.4 COMMAND STRUCTURE FOR USER DATA

### 35.4.1 CREATION PHASE

The structure command is as follows:

```
GREEN_FUNCTION_TAPE
      XDR_BUFFER buffer_size
      CREATION
            n
            FILE_NAME filename_1
                  <PARTICLE_TYPE>
                  <ESTIM_NAME>
*                 VOLUME numvol_1
*                 FRONTIER numvol_1(1) numvol_1(2)
            ...
            FILE_NAME filename_n
                  <PARTICLE_TYPE>
*                 VOLUME numvol_n
*                 FRONTIER numvol_n(1) numvol_n(2)
                  <ESTIM_NAME>
      END_CREATION
END_GREEN_FUNCTION_TAPE
```

where:

- n is the number of required files
- buffer_size is the size of the buffer used for the Green's function files dump. The keyword XDR_BUFFER is optional. When it is not used, the default buffer size is set to $2.10^9$.
- <PARTICLE_TYPE> is either NEUTRON for pure neutron problems, PHOTON for pure photon problems or PHOTON_FROM_NEUTRON_SOURCE for coupled neutron-photon problems ("initial" information are then related to the source neutrons and "final" information are related to the elementary contributions to the

photon scores; in this case, when these Green's function files are re-used, the user will just have to define photon responses in the directive RESPONSES).

- filename_i is the name (chosen by the user) of the file associated to the $i^{th}$ requested file.

  The files are created in the tree structure, at the level of the call to TRIPOLI-4® in mono-processor mode. If the creation takes place in parallel mode, the code will automatically open one file for each simulator, with the appropriate parallel suffix $< j >$th_task (for the $j^{th}$ simulator). The use of the graph file command RUN_DIRECTORY enables to create the files in a requested directory in parallel mode (see chapter 33).

- <ESTIM_NAME> is the type of estimator. It can only be COLL, TRACK, or SURF.

- The volumes are defined by their number numvol_i and the surfaces are defined by the numbers of the two delimiting volumes, numvol_$i_1$ and numvol_$i_2$.

**Note:** In addition to the creation of Green's function files, the user can use the simulation performed during the creation phase to ask for standard scores (using the SCORE command).

### 35.4.2 EXPLOITATION PHASE

The structure command is as follows:

```
GREEN_FUNCTION_TAPE
    EXPLOITATION
        n
        FILE_NAME filename_1 filename_2... filename_n
*       NEW_SCORES
*       SOURCES_CONTRIBUTIONS
        ...
    END_EXPLOITATION
END_GREEN_FUNCTION_TAPE
```

where:

- n is the number of files to be exploited.

- filename_1 to filename_n are the exploited files.

  It is highly recommended that the filenames of the creation and exploitation blocks are exactly the same. Specifically, if the creation phase took place in a parallel run, the code will automatically open the files with the parallel suffix $< j >$th_task appended to the user filenames (*i.e.*, the suffix must not be present in the filenames given by the user).
  The code searches for these files in the tree structure where the call to TRIPOLI-4® is made. Among

these n files, only those corresponding to the tallying area of the scores required in the phase of exploitation are really used.

- **There are two kinds of exploitation:**

1. The keyword NEW_SCORES corresponds to a change of sources. Then, we use the Green's function files to compute new scores corresponding to this new source.
2. The keyword SOURCES_CONTRIBUTIONS corresponds to the computation of conditional scores taken from the simulation of the files creation. The source may be changed or remain the same as in the creation phase; however, we look for scores conditional to different cells of the tabulated source and to different energy groups of the source particles.

In this last case of exploitation, additional data have to be given as follows:

```
SOURCES_CONTRIBUTIONS
    GRID  n
        val1 val2... valn
*       SOURCES_NUMBER  <LIST_OR_SUM> m
            num1 num2 ... numm
*       SOURCES_INTER_VOLU  <LIST_OR_SUM> m
            volunum1 volunum2 ... volunumm
*       SOURCES_ALL_TABULATIONS  m
            num1 num2 ... numm
```

where :

- GRID gives the energy grid of the source (in n - 1 groups). The grid has to be in a decreasing order.
- <LIST_OR_SUM> equates to LIST or SUM. m is the number of volumes in the list or the sum required;
- With respect to spatial considerations, the user can either:

1. use the keyword SOURCES_NUMBER to input the numbers of the sources (index from 0 to the number of sources in the directive SOURCES_LIST - 1). The conditional score will be given for each source (case LIST) or for the full set (case SUM).
2. use the keyword SOURCES_INTER_VOLU to define the volumes (with any of the forms known by the geometry of TRIPOLI-4®) including sources or parts of sources. Only the intersection of these volumes (for which the number is given) and the sources are retained.
   **Be careful**, these volumes have to be defined before in the geometry as virtual volumes (with the keyword FICTIVE) and are put on top of the geometry. Moreover, these volumes have to be disconnected. The conditional score will be given for each virtual volume (case LIST) or globally for the full set (case SUM).

3. use the keyword SOURCES_ALL_TABULATIONS to request contributions on all cells of the tabulated sources whose numbers are mentioned from num1 to numm (index from 0 to the number of sources in the directive SOURCES_LIST - 1); this is implicitly a list of sources, since the sum has no meaning here.

**Note:** During the exploitation phase of Green's function files, the user can request the display of source sites (see subsection 25.1.6) in order to analyze the birth sites of particles involved in the data storage of the Green's function files.

## 35.5 USE OF PARALLELISM

Working in parallel mode is similar to working in a single-processor mode. It should be noted, however, that the exploitation phase is always performed in mono-processor mode.
**Warning:** It is necessary to request at least one score in parallel mode. In this version of Tripoli4, the parallel mode does not work unless a result is requested.
As previously noted, the directory for Green's function files is the same as that where the simulators were executed. In parallel mode, this directory may vary depending upon the execution mode. The RUN_DIRECTORY command must be used in the graph file (see chapter 33) in order to indicate this specific directory.

## 35.6 ASPECT OF THE OUTPUT FILE

### 35.6.1 CREATION PHASE

After the standard printing of the required scores is completed, the output file contains information about the number of particles collected for each file (i.e., each potential tallying area for the exploitation phase). This information is introduced using "NUMBER OF CONTRIBUTING PARTICLES" at the end of the output file.

### 35.6.2 EXPLOITATION PHASE

The information (present or absent) used to indicate a file exploitation instead of a real simulation is as follows:

1. absence of statistical information on the biasing (in the case where a variance reduction technique is used) for the first six batches, usually tallied at the beginning of the simulation.
2. "mean weight leakage : unknown": before the printout of the scores (in fact, the statistics about particles leakage cannot be input since there is no simulation of the particles trajectories)
3. "exploitation time" instead of "simulation time" at the end of the output file.

- **Case** NEW_SCORES **:** The printing of new scores is done in the standard form because these scores have the same structure as the usual scores.

- **Case** SOURCES_CONTRIBUTIONS **:** In this case, the printing of scores changes. For each required score, the different tallying variables are given in the following order:

1. first, by group of energy sources (outer loop): this is introduced using "SOURCE SPECTRUM STEP NUMBER"
2. then, by source number "SOURCE NUMBER" or by virtual source volume "SOURCE VOLUME" (for a LIST) or globally for the full set with "SUM ON SPECIAL SOURCE NUMBERS" or "SUM ON SPECIAL SOURCE VOLUMES" (for a SUM)
3. then by (U,V,W) cell indices (relatively to the VAR_U, VAR_V and VAR_W variables defined in the syntax of the tabulated sources, cf. section 12.2.2.3) if the option SOURCES_ALL_TABULATIONS is used
4. then, by group of tallying energy (inner loop): this is introduced using "SPECTRUM RESULTS", as for usual scores.

The energy integrated scores are written only when a single source number or a single virtual volume number is mentioned in LIST or when a SUM is used. They are not written if the option SOURCES_ALL_TABULATIONS is used.

## 35.7 INCOMPATIBILITIES

The following features are incompatible with the Green band calculations:

- VOV, BOOTSTRAP, and COMPLETE options of the scores;
- current scores.

Some other incompatibilities may be found in the section limitations (35.3).

# Chapter 36

# TRIPOLI-4/ROOT COUPLING

ROOT is an object oriented software package used for processing and analyzing data and has been developed by the CERN. More information on this software is available on the ROOT website at the following URL: http://root.cern.ch/ . This chapter illustrates the different possibilities offered by coupling TRIPOLI-4® and ROOT.

## 36.1 INSTALLATION PROCEDURE

In order to use the functionalities afforded by ROOT, it is first necessary to install the software itself. The instructions for this are found on the website mentioned, above. TRIPOLI-4® version 12 is coupled with ROOT version 6.12.06-90-g4816da3.
The TRIPOLI-4® code must then be compiled using the **--enable-root** option as follows:

```
./configure --enable-root
make
```

Be sure that the ROOT environment variables ($ROOTSYS, $LD_LIBRARY_PATH, and $PATH) have been properly defined (as indicated on the CERN website) before attempting this.

## 36.2 USING A ROOT GEOMETRY

### 36.2.1 DEFINING THE GEOMETRY

The TRIPOLI-4® code is capable of using a geometry defined with ROOT. Constructing a geometry input deck using ROOT is described in great detail in the chapter "The Geometry Package" of the ROOT user's manual. Once the geometry has been entered, TRIPOLI-4® can use ROOT to navigate in it.
It should be noted that throughout the TRIPOLI-4® input deck we no longer allude to volumes by their numbers but by their names. The name of a volume is given by its path followed by its occurrence number. For example, if in the definition of a ROOT geometry there are two occurrences of volume B in volume A, then volume A will be designated as /A_1, volume B as /A_1/B_1, and the second occurrence of B as /A_1/B_2 . In addition, the `GEOMCOMP` command is no longer required: in creating a ROOT geometry, the user must associate a composition name, as it is defined in the `COMPOSITION` command, for each ROOT volume.
The ROOT geometry can be entered either via a file or directly via C++ or Python code within the

**ROOT_GEOMETRY** directive.

#### 36.2.1.1 Defining the geometry via a file

There are 4 different possibilities to specify a ROOT geometry via a file:

- a ROOT geometry file. In order to do this, the user must save his or her TGeoManager in a ROOT file.
- a ROOT script (with extension .C) interpreted on-the-fly
- a ROOT script (with extension .C) first compiled and then executed on-the-fly
- a PyROOT script (ROOT with Python bindings)

The user must then use the following syntax block to replace the command block `GEOMETRY`.

```
ROOT_GEOMETRY
        file(name)
END_ROOT_GEOM
```

where `file(name)` can be :

- a file name with extension .root
- a ROOT script with extension .C to be interpreted on-the-fly
- a ROOT script with extension .C+ to be compiled and then executed on-the-fly. This should be faster than the previous option.
- a PyROOT script (reactor physics *templates* can be used if the $PYTHONPATH environment variable is properly set)

#### 36.2.1.2 Defining the geometry directly in C++ or Python code

For quality insurance purposes, another way of defining the ROOT geometry has been made available to users. It consists in defining the geometry directly via C++ or Python code embedded in the **ROOT_GEOMETRY** directive of the TRIPOLI-4® input deck, preceded and followed by the keyword **CPP_ROOT_GEOM** or **PYTHON_ROOT_GEOM** . Such a geometry definition appears like this in the input deck (for a C++ example):

```
ROOT_GEOMETRY
CPP_ROOT_GEOM
        gSystem →Load("libGeom");
        TGeoManager *mgr = new TGeoManager("Geom","composite shape example");
        TGeoMaterial * neutre = new TGeoMaterial("NEUTRE",
        mgr→GetElementTable()→GetElement(0),1);
        ...
        ...
        mgr→Close();
CPP_ROOT_GEOM
END_ROOT_GEOM
```

Starting from TRIPOLI-4 version 10, it is no more necessary to export the ROOT geometry in a ROOT file and then specify the same ROOT file just after the closing **CPP_ROOT_GEOM** or **PYTHON_ROOT_GEOM** keyword. The TGeoManager object is built on-the-fly and directly used in the simulation.
Also, it is important to be aware that such an input deck does not allow to embed complex set of multiple C++ or Python class and files and shall therefore be restricted to simple geometries declared in a unique file.

### 36.2.2 DEFINING BOUNDARY CONDITIONS

As with a native TRIPOLI-4® geometry, the ROOT geometry allows for boundary conditions to be applied. The command used for this is as follows:

**ROOT_LIMIT**
**name(vol1) name(vol1') <TYPE_CONDITION>**
**…**
**name(voln) name(voln') <TYPE_CONDITION>**
**END_ROOT_LIMIT**

where `<TYPE_CONDITION>` may be equal to one of the following values: `REFLECTION`, `TRANSLATION` (works only on box), or `COSINUS`. Unlike in native TRIPOLI-4® geometry, the boundary conditions in a ROOT geometry are oriented (signed) and are attached to surfaces where two volumes are in contact. As such, the boundary condition `<TYPE_CONDITION>` is applied to particles transported from volume `name(voln)` to volume `name(voln')`. One can also set `name(voln')` to `*ALL*`, in which case the boundary condition is applied to all particles that penetrate the volume `name(voln)` (warning: unlike in the first case, the use of `*ALL*` keyword reverses the logic used by the boundary condition concerning the sequence of input volumes).

In the case of the rotation boundary condition, the syntax is as follows:

**num(vol1) num(vol1') ROTATION** $X_c\ Y_c\ Z_c\ u_x\ u_y\ u_z\ \alpha$

see boundary limit chapter 9 for parameters details if needed.

### 36.2.3 DEFINING A SCORE

In the case of a score on a volume it is possible to ask for the scores on all the volumes who are inside it without having to ask for them one by one.
Example:

```
2 TRACK
GRID GRID_ABSORPTION
VOLUME LIST 1 /TOP_1/boite_1/*
```

Response 2 is requested with estimator `TRACK` for the grid indicated in all the volumes who are declared (in the ROOT geometry definition) as inside the "/TOP_1/boite_1" volume.

Notes: the sum of the results can be made by using the keyword `SUM` instead of `LIST`.

### 36.2.4 HOW TO CHECK A ROOT GEOMETRY

Some tools are available to check the user's ROOT geometry. Besides usual TRIPOLI-4® checking tools (c.f. 7.13.2), Root also offers convenient tools to check on the geometry. The user is advised to read carefully the ROOT user guide to fully take advantage of all ROOT geometry possibilities.

### 36.2.5 CAVEAT: USE OF OVERLAPPING NODES

ROOT offers the possibility to construct geometries with overlapping volumes. To use this feature, the volumes must be added to the construction tree with the `TGeoVolume::AddNodeOverlap()` method. However, particle tracking in overlapping volumes does not seem to be fully supported in ROOT, according to the ROOT developers[1]. The use of ROOT geometries constructed with `TGeoVolume::AddNodeOverlap()` is therefore not recommended in TRIPOLI-4. Users are encouraged to replace calls to `TGeoVolume::AddNodeOverlap()` with `TGeoVolume::AddNode()` and to modify the volume shapes using Boolean operators to suppress any existing overlap.

### 36.2.6 EXAMPLE

A simple example using the ROOT geometry is described here. The ROOT input deck in C++ used to construct the geometry indicated by figure 36.2 is given in figure 36.1. In this file it should be noted, in particular, the geometry/composition association made in the TGeoMedium class constructor. It is the name of the material declared at this point in the construction of the geometry that is reused later in the TRIPOLI-4® input deck (see figure 36.3).

[1] https://root-forum.cern.ch/t/unexpected-navigation-behaviour-with-overlapping-nodes/26618

```
gSystem->Load("libGeom");
TGeoManager *geom = new TGeoManager("test_conditions", "Boundary Conditions Test");
TGeoVolume *bigbox, *ball, *mesure, *elem;

TGeoMaterial *vacuum = new TGeoMaterial("vacuum", geom->GetElementTable()->GetElement(0), 1);
TGeoMedium *fuel = new TGeoMedium("FUEL", 1, vacuum);
TGeoMedium *water = new TGeoMedium("WATER", 1, vacuum);
TGeoMedium *air = new TGeoMedium("AIR", 1, vacuum);
TGeoMedium *carb = new TGeoMedium("CARBON", 1, vacuum);

world = geom->MakeBox("world", air, 100, 100, 100);
bigbox = geom->MakeBox("BigBox", air, 10, 10, 10);
bigbox2 = geom->MakeBox("BigBox2", air, 60, 60, 3);
bigbox3 = geom->MakeBox("BigBox3", air, 60, 60, 3);
bigbox4 = geom->MakeBox("BigBox4", air, 60, 3, 60);
bigbox5 = geom->MakeBox("BigBox5", air, 60, 3, 60);
volrefl = geom->MakeBox("volrefl", water, 5, 70, 70);
volrefl->SetLineColor(63);
ball = geom->MakeSphere("uranium", fuel, 0, 10);
ball->SetLineColor(2);

world->AddNode(ball, 1, new TGeoTranslation("tball", 80, 0, 0));
world->AddNode(bigbox, 1, new TGeoTranslation("tbigbox", -80, 0, 0));
world->AddNode(bigbox2, 1, new TGeoTranslation("tbigbox2", 0, 0, 90));
world->AddNode(bigbox3, 1, new TGeoTranslation("tbigbox3", 0, 0, -90));
world->AddNode(bigbox4, 1, new TGeoTranslation("tbigbox4", 0, 90, 0));
world->AddNode(bigbox5, 1, new TGeoTranslation("tbigbox5", 0, -90, 0));
world->AddNode(volrefl, 1, new TGeoTranslation("tvolrefl", 0, 0, 0));

geom->SetTopVolume(world);
geom->CloseGeometry();
world->SetVisContainers(kTRUE);
geom->SetTopVisible(kTRUE);
geom->Export("test_conditions.root");
world->Draw();
```

Figure 36.1: Sequence of ROOT commands in C++ used in defining a simple geometry

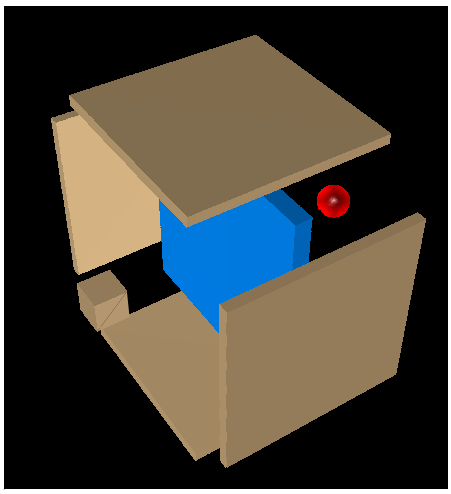


Figure 36.2: Visualization of the preceding geometry using ROOT

```
ROOT_GEOMETRY
test_conditions.root
ENDGEOM

ROOT_LIMIT
/world_1/BigBox2_1 *ALL* REFLECTION
END_ROOT_LIMIT

COMPOSITION
4
POINT_WISE 300 FUEL 1
O16 3.2967e-15

POINT_WISE 300 CARBON 1
C 100

POINT_WISE 300 WATER 1
O16 100

POINT_WISE 300 AIR 1
N14 4e-9

END_COMPOSITION

// post-processing data
TRACKS
...
END_TRACKS
```

Figure 36.3: TRIPOLI-4® input deck allowing for the previously defined ROOT geometry to be used

## 36.3 ROOT POST-PROCESSING: RECORDING PARTICLE TRACKS

It is also possible to record the set of tracks (or a sample of it, if the number of simulated particles is significant) for a simulation. The command is as follows:

**TRACKS**
*BATCH nb(batch) num(batchi) ... num(batchj)*
*VOLUME nb(vol) name(voln) ... name(voln')*
*INTERACTION nb(inter) code(interm) ... code(interm')*
*REDUCED_TRACKS*
*PATH name(file)*
**END_TRACKS**

where `nb(batch)`, `nb(vol)`, `nb(inter)`, `num(batchi)`, `name(voln)` and `code(interm)` are, respectively, the number of batches, the number of volumes, the number of interaction types, the batch number i, the name of volume n and the code for interaction m (as listed in section 13.2).

Various filters can be used, either alone or together. The `BATCH` command may be used to filter the batches to be recorded, the `VOLUME` command may be used to select the tracks that pass through certain selected volumes and the `INTERACTION` command may be used to filter the occurrence of given interaction types. If both `INTERACTION` and `REDUCED_TRACKS` are requested, then only the events corresponding to the requested interaction types are recorded, whereas if `REDUCED_TRACKS` is not present, the whole particle histories are recorded, provided one of their events at least matches a requested interaction type.
When these keywords are absent, no filtering is performed, and, as a consequence, all of the tracks are recorded (when the storage space of the system allows it).

The tracks are recorded in a file command with the extension .root, which contains all of the simulation data: the type of particle simulated, the type of collision, the position, the time, the weight, etc... With this file, it is thus possible to develop post-processing tools using the ROOT libraries. The path of this file may be specified by the `PATH` command (with `name(file)` the chosen path). Otherwise, the file is written in the current directory with a default name : `Tripoli_tracks.root` in case of native geometry and `name(geom)_tracks.root` in case of ROOT geometry.

# Chapter 37

# ENERGY CUT PER VOLUME

An energy cut can be set per volume for each particle type. This feature allows to avoid particle tracking in cells of less physical interest, in particular in the case of electron/positron transport.

The general syntax is as follows :

**VOLENERCUT**
  **<TYPE_OF_PARTICLES>**
    ***ALL default_energy_cut***
    **numvol**(1) **energy_cut**(1)
    ...
    **numvol**($n$) **energy_cut**($n$)
  ...
**END_VOLENERCUT**

where :

1. <TYPE_OF_PARTICLES> can be `NEUTRON`, `PHOTON`, `ELECTRON` or `POSITRON`
2. `ALL` *default_energy_cut* sets a default energy cut for all volumes

The default energy cut is overwritten by all following energy threshold definitions for a given volume.

The following example sets an energy threshold on electrons of 10 MeV for all volumes except 10 and 11 where the threshold is 5 MeV and an energy threshold on photons of respectively 1.0 and 0.5 MeV for volumes 10 and 13 (all others volumes have the threshold given in `SIMULATION` command).

```
VOLENERCUT
     ELECTRON
     ALL 10.0
```

```
      10  5.0
      11  5.0
      PHOTON
      10  1.0
      13  0.5
END_VOLENERCUT
```

# Chapter 38

# VISUALIZER T4G

## 38.1 INTRODUCTION TO T4G

T4G is a tool to visualize and check the data interpreted by Tripoli-4 or Root.
T4G window is divided in two areas (figure 38.1): the left pane contains the picture of the data, while the right pane contains some information about the data set or the data pointed by the mouse.

T4G manages color policy. Each policy is the representation of a same kind of parameters. One color depicts a parameter to make it easily recognizable among the others.

The available color policies are:
- "volumes", exactly 1 color per volume;
- "compositions", exactly 1 color per composition;
- "colors", colors defined by the user for some or all volumes in a *.colors file;

"Compositions", "scores" and "colors" policies are optional.

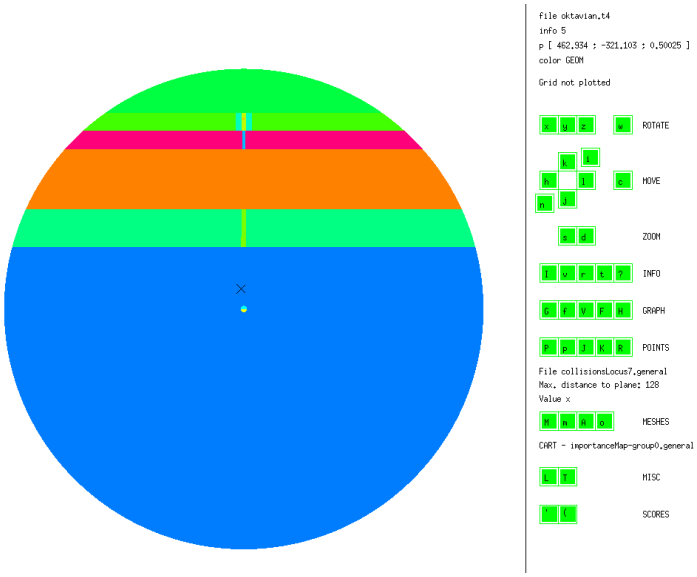



Figure 38.1: T4G window

## 38.2 HOW TO LAUNCH T4G

To run T4G with the data set `data_set`, type the following command:

```
./static_visutripoli4 <data_set> [--<option_1> <value_1> [--<option_2> <value_2> ...]]
```

where `data_set` can be a Tripoli-4 or a Root data set. The type of geometry is deduced from the file extension of the data set: `*.t4` for Tripoli-4 geometries, `*.root` for root ones. Other extensions will lead to the execution stop.
The accepted options keywords and associated values are described below:

```
--lang        : language (0 -> english, 1 -> french, 2 -> old).
--fromfile    : name of commands file
--title       : title of the GUI window
--screensize  : heigth of the window
--para        : parallel flag followed by number of tasks
--window      : list of values defining the frame of the window :
                [ox oy oz ux uy uz vx vy vz lu lv] (see section \ref{win})
--3D          : 3D visualization (if not set : 2D visu)
--help        : print the help
--expert      : disable the automatic load of the files .colors, .vvls,
                .scores, .meshes and .points.
```

If the option --expert is used then the following keywords can be used :

```
 --colors     : to load a "colors" file
```

```
--vvls       : to load a "volumes values" file
--scores     : to load a "*.dat scores" file
--meshes     : to load a "*.general meshes" file
--points     : to load a "*.general points" file
```

followed by the name of the file.

Default values are:

- lang : old;
- window : -1 -1 0.5 1 0 0 0 1 0 100 100 (see section 38.3.4);

**Parallel flag**
The parallel flag only concerns the native geometry of Tripoli-4 (the CERN/ROOT is not concerned).
It is followed by the number of tasks to use. T4G uses OpenMP for shared memory parallelisation. The screen is decomposed into multiple horizontal frames each slave process is filling with geometry. As different frames do not necessarily take similar durations to complete, do not hesitate to ask for more tasks (no more than a factor 2 though) than really available cores (or CPU threads). It allows task stealing and better handling of tail effect (tasks not finishing at the same time), so a better parallelism efficiency.

## 38.3 2D VISUALIZATION

### 38.3.1 GEOMETRY RETRIEVAL

In 2D, a geometry can be plotted in three modes: line by line volumes ("pursuit"), point by point volumes ("position"), or frontier ("contours") modes (figure 38.2).

The default mode is "pursuit".

To switch between "pursuit" and "frontier" modes, press the `V` key. To switch between "pursuit" and "position" modes, press the `E` key.

#### 38.3.1.1 "Pursuit" mode

This is the default mode. The volumes are plotted with different colors in different volumes.

#### 38.3.1.2 "Position" mode

This volume mode displays the same data than the previous one, but the algorithm used is different. This visualization mode is slower than the "pursuit" one but should show the same plot. However, incorrect geometries may lead to differencies between these plots. A good use is then to navigate through the geometry in "pursuit" mode and switch to "position" mode from time to time to check the coherency of the geometry.

#### 38.3.1.3 "Contours" mode

The frontier mode displays the frontiers between different volumes by a black line.

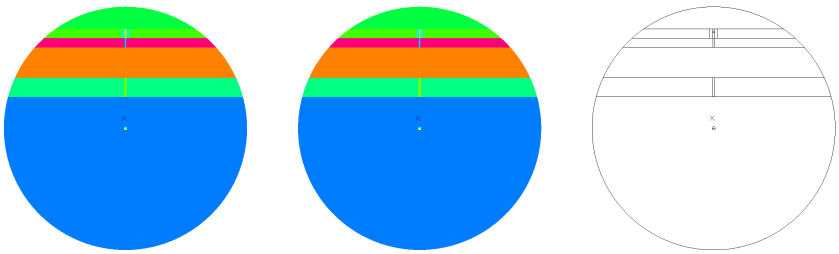


Figure 38.2: From left to right: geometry visualization in "pursuit", "position" and "contours" modes

### 38.3.2 DISPLAY OF THE COLOR POLICY

In 2D, T4G can handle several color policies which are :
- "volumes";

- "compositions", if they are available in the data set;
- "colors", if the `data_set.colors` file exists and is not empty.

Pressing `N` selects a policy between the available ones. In the information pane, at the line "info", the name or value of the policy at the place pointed by the mouse is then displayed.

Pressing `C` refreshes the display in the left pane: the current policy is displayed (figure 38.3). Its name is indicated in the right pane at the line "color" by the matching keyword:

| Policies | Keyword |
|---|---|
| volumes(geometry) | GEOM |
| compositions | COMPOS |
| colors | INFOCOLORS |

A volume may be undefined for "colors" policy. In that case, it is painted black.

The following picture (figure 38.3) shows an example of the visualization for each policy:

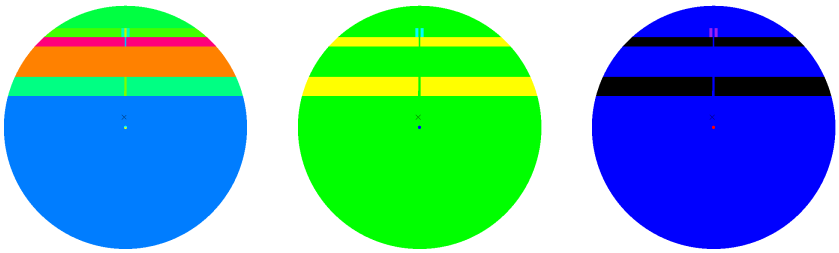


Figure 38.3: From left to right: visualization of "volumes", "compositions" and "colors" policies

"Compositions" and "colors" policies are always drawn using "pursuit" mode.

It is possible to add the contours of the geometry to the current policy by pressing key `f`.

#### 38.3.2.1 "Volumes values" file

The association of a volume and its value is defined in the "vvls" file. Its structure is the following:

```
<name of a volume>         <volume value>
<name of another volume> <volume value>
...
```

This file may be provided be a Tripoli-4 run. See section 23.1 for details. If T4G find such a file, volume score will the score defined previously divided by the value of the volume.

#### 38.3.2.2 "Colors" file

The association of volumes and colors is defined in the "colors" file. Its structure is the following:

```
<name of a color>       <number of volumes> <list of volume names>
<name of another color> <number of volumes> <list of volume names>
...
```

The color names can be written in lower or uppercase. They are the ones defined in X11 library file `rgb.txt`.

### 38.3.3 TRIPOLI-4 SCORES, MESHES AND POINTS PLOT

Data such as scores on volumes or values on meshes or points may be provided by a Tripoli-4 run. The generated files are respectively named `data_set.scores`,`data_set.meshes` and `data_set.points` files. They contain the list of the `*.dat` or `*.general` files that contain the data.

The `data_set.scores`,`data_set.meshes` and `data_set.points` files have the same structure: each line must start with the keyword `name` and then contains the name of the `*.dat` or `*.general` file.

In the meshes file, each **mesh must be defined on exactly one line**. The user can set different parameters of the mesh :

- the frame (`frame`)
- special color legend (`colorlegend`)
- min and max values (`minmax`)
- reversing the color legend (`neg`)
- a multiplying coefficient (`coeff`)

All of these parameters are optional. The user can disable the rest of the line options with the `#` keyword.

```
name <name of a *.general file>       [keyword <definition of option >] ...
name <name of another *.general file> [keyword <definition of option >] ...
..
```

For meshes and points the definition of the frame in the geometry is available and given after the keyword `frame`.If no frame is defined, the frame is the same as the geometry's one. A frame is defined by its origin `o` and two unit vectors `u` and `v`:

```
frame <ox oy oz ux uy uz vx vy vz>
```

For meshes the definition of the mesh's color legend is available and given after the keyword `colorlegend`.If no color legend is defined, the standard color legend is setted. A color legend is defined by a set of three types of parameters : the number of colors `ncolors`, the treshold of each color `lim` and the name of the color `color`.

```
colorlegend < ncolors min color0 lim1 color1 ... lim(n-1) color(n-1) max colorn >
```

The user can set the values of the min and the max of the current mesh with the following keyword : `minmax` and the values of min and max.

```
minmax < min max >
```

The user can reverse the color scale. Usually, high values are in red and low values in blue. With this mode, low values are in red and high values in blue.

```
neg
```

This option may be useful if the user wants high importances in red and low importances in blue. The quantity dumped when using the `STORE_IN_FILE` keyword is $-\log(I)$ with $I$ the importance function. Thus, using the `neg` keyword, high importances can appear in red and low importances in blue.

The user can set a multiplying coefficient for the current mesh with the following keyword : `coeff` and the value of the multiplying coefficient.

```
coeff <coeff>
```

Note that the data plotted here will be added to the current plot, but will disappear after a call to a move or an update.

The list of available actions on meshes and points are given in section 38.3.5. The main features are described below.

#### 38.3.3.1 Scores plot

If T4G finds a `data_set.scores` file in the working directory, it will load all the related scores files.

A score file defines the association of volumes and scores. Its structure is the following:

```
<name of a volume>       <score value>
<name of another volume> <score value>
...
```

Pressing left parenthesis key "`)`" switches between the available scores files. The name of the current scores file `*.dat` appears in the right pane of the T4G window in the `SCORES` section.

Pressing the key "`O`" displays the current scores (figure 38.4).

Note that the scale that is plotted at the bottom left side of the plot area is computed so as to take into account the values in all the loaded geometry. A volume may be undefined: in this case, it is painted white. Scores are drawn using "pursuit" mode.

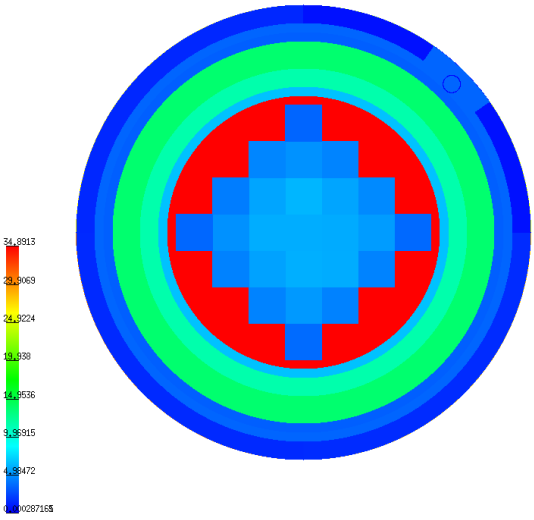


Figure 38.4: Scores on volumes

#### 38.3.3.2 Meshes plot

If T4G finds a `data_set.meshes` file in the working directory, it will load all the related meshes.

Pressing `m` switches between the available meshes. The name of the current mesh file `*.general` appears in the right pane of the T4G window in the `MESHES` section.

Pressing `T` selects the mesh frame type that will be used to convert the mesh positions into the cartesian geometry's frame. The selected frame type is indicated in the T4G window just before the mesh file. It can be one of:

| Frame type | Keyword |
|---|---|
| `cartesian` | `CART` |
| `cylindrical` | `CYL` |
| `spherical` | `SPH` |
| `spherical`$(r,\mu,\phi)$ | `SPH_MU` |

As there is no way to recognize a mesh type, it is the responsibility of the user to select the right one. If the selected type does not match the mesh type, the display will be incoherent. However, the mesh type is now included in the comments of the .general file format as long as a possible change of frame:

```
#frame_type = <frame_type>
#origin = <ox>, <oy>, <oz>
#vectors = <ux>, <uy>, <uz>, <vx>, <vy>, <vz>
```

T4G is able to interpret these comments to set a proper default.
The available possibilities for the frame type are CARTESIAN, CYLINDRICAL, SPHERICAL, SPHERICAL_MU and TRIANGLE_Z. SPHERICAL_MU is the same as spherical, but with a $\cos\theta$ instead of $\theta$. It allows to easily represent isotropic data. TRIANGLE_Z is related to meshes with prismatic cell (see extended meshes, section 15.8) used to score, in particular, hexagonal structures.

Pressing `M` superimposes the current mesh's display over previously displayed geometrical elements (figure 38.5).
Pressing `space` redraws with the current color policy (re-initialized the display). Hence to display only the current mesh, one should press `space` then `M`.

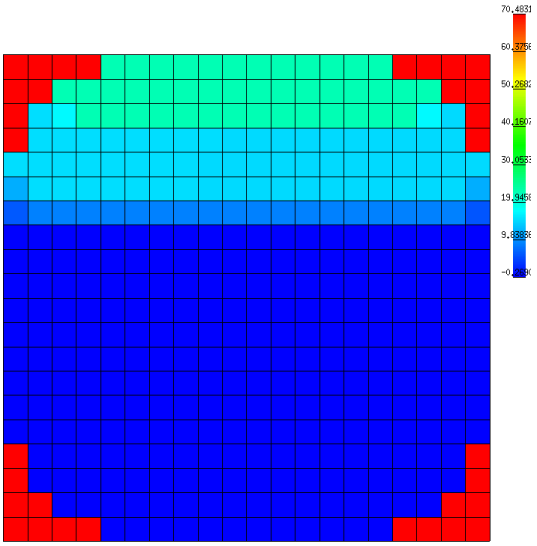


Figure 38.5: Data mesh with grid

Note that the scale that is plotted at the top right side of the plot area is computed so as to take into account the values in all the loaded meshes.

The mesh grid can be enabled/disabled with the key `B`.

A logarithmic scale can be used by pressing `L`. In this case, the keyword "Log" appears in the right part of the T4G window, under the line "color".

#### 38.3.3.3 Points plot

If T4G finds a `data_set.points` file in the working directory, it will load all the related points files.

Pressing `p` switches between the available files. The name of the current file `*.general` appears in the right pane of the T4G window in the `POINTS` section.

Pressing R switches between the different fields of points available in the current file. The field's name is indicated in the T4G window at the line "value". See section 31.4.2 for a description of available fields.

Pressing P superimposes the current field's display over previously displayed geometrical elements (figure 38.6). As it is likely to get very few points in the current plane, the plotted points are the ones located at a distance lower to the one indicated at the line "Max. distance to plane:". This distance can be lowered (key J) or increased (key K) by a factor two. The default distance is 128 cm.

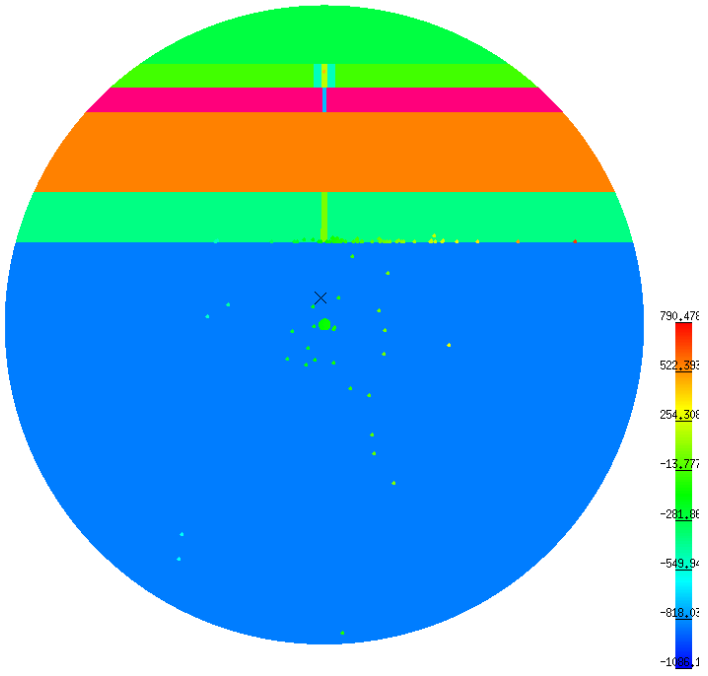


Figure 38.6: Data points

Note that, unlike the behaviour for meshes scale, T4G will plot a scale at the bottom right side of the representation pane that takes into account only the data of the current points set.

### 38.3.4 WINDOW

#### 38.3.4.1 Definition

The plot frame or "window" defines the frame in which the geometry will be drawn.

The required informations to define it are:
- o : the origin position;

- u : the first vector of the system;
- v : the second vector;
- lu : the width, along the first vector;
- lv : the height, along the second vector;

T4G admits three ways to define a window. Here they are, in order of priority:

- using the option `--window` in the command line;
- loading the file `data_set.window` from the current working directory;
- using the default frame (no option `--window`, no file `data_set.window`).

In 2D, when the user closes T4G, the last characteristics are saved in the file `data_set.window`.

#### 38.3.4.2 Default window

The default frame characteristics are:
- o : [-1., -1., 0.5];
- u : [1., 0., 0.];
- v : [0., 1., 0.];
- lu : 100;
- lv : 100.

#### 38.3.4.3 Command line option

The same frame would be defined using the following command line option :

```
--window  [ox oy  oz ux uy uz vx vy vz  lu  lv]
--window   -1 -1 0.5  1  0  0  0  1  0 100 100
```

#### 38.3.4.4 Window file

The matching `data_set.window` file would contain :

```
o -1 -1 0.5
a 1 0 0
b 0 1 0
la 100
lb 100
```

### 38.3.5 AVAILABLE KEYS IN 2D MODE

2D features are given in the following tables:

| Key | General |
|---|---|
| q or Q | Quit |
| ? | Display the help |
| space or a | Redraw with the current color policy (re-initialized the display) |
| G | Draw grid |
| = | Reload meshes and points |
| H | Hide the right pane of the T4G window |
| F | Export image: generates .tiff and .bmp pictures of the T4G window. |

| Key | Drawing policy |
|---|---|
| N | Loop over the color policy |
| C | Call the display of the current color policy |
| E | Toggle between "pursuit" and "position" drawing methods |
| V | Toggle between "pursuit" and "contours" drawing methods |

| Key | Memorize in stack |
|---|---|
| < | Add the mouse position in the stack |
| > | Remove the last position from the stack |
| ! | Set position in the stack from console input |

| Key | Information at the console |
|---|---|
| v | Display the name of the current volume, its color RGB and the current mesh value under the mouse pointer |
| I | Window definition |
| * | Enable/disable drawing of the scale |
| , | Draw a legend on the plot area : the name of the current policy under the mouse pointer (score value in scores case) |
| r | Display the ray between the mouse positions kept in stack |
| t | Display the ray from center to volume next to cursor |

| Key | Move |
|---|---|
| h or left | Translation towards x - |
| l or right | Translation towards x + |
| j or down | Translation towards y - |
| k or up | Translation towards y + |
| n or Next Page | Translation towards z - |
| i or Previous page | Translation towards z + |
| d or - | Unzoom |
| D or Shift+- | Unzoom more |
| s or + | Zoom |
| S or Shift++ | Zoom more |
| W | Zoom from positions in stack |
| x, y, z | Compute a plane cut along the specified direction |
| X, Y, Z | Rotate 90 deg around X, Y and Z axis |
| w | Rotation defined by the center of the plot and the mouse position |
| c | Center the plot to the mouse position |

| Key | Actions on Scores |
|---|---|
| O | Display the current scores |
| b | Change current scores |

| Key | Actions on Meshes |
|---|---|
| M | Superimpose the current mesh's display over previously displayed geometrical elements |
| m | Change current mesh |
| A | Select / unselect all the meshes as current mesh |
| B | Plot / do not plot the mesh grid |
| U | Increment number of isovalues |
| u | Decrement number of isovalues |
| o | Display the isovalues of the current mesh |
| _ | Load all mesh files (.meshes, .resultmeshes, .sourcemeshes and .importancemeshes) of the current working directory |
| L | Use / do not use the logarithmic mode |
| T | Change mesh type (cartesian, cyclindrical, spherical, spherical mu) |

| Key | Actions on Points |
|---|---|
| P | Superimpose the current point field's display over previously displayed geometrical elements |
| p | Change current points file |
| R | Change points field |
| J | Divide by 2 the maximum distance between current plane and points to plot |
| K | Multiply by 2 the maximum distance between current plane and points to plot |

| Key | Common to Meshes and Points |
|---|---|
| f | Display the contours of the geometry |

| Key | Commands file |
|---|---|
| g | Stop file input |

## 38.4 3D VISUALIZATION

This mode allows some kind of 3D representation of the geometry. It is considered as experimental and it is not currently known if further enhancement will be possible on this concept.

### 38.4.1 LAUNCH 3D VISUALIZATION

In order to get a 3-dimensional view of a geometry, type the command:

```
./static_visutripoli4 data_set --3D
```

where `data_set` is the name of a *.t4 or *.root file.

### 38.4.2 BASIC FEATURES

At the moment, the 3D visualization uses one mode "volume" to draw the geometry. The drawing of "compositions", "scores" and "colors" policies are not available.

#### 38.4.2.1 Exclusion of volumes

Pressing key - (minus) removes the volume located under the mouse pointer from the visualization (figure 38.7).



Figure 38.7: Left: the geometry from outside. Right: after the exclusion of the main envelop.

#### 38.4.2.2 Manipulation

All operations such as moving, centering, zooming, ... are available (figure 38.8). The related keys are given in section 38.4.5.

### 38.4.3 HIDING A PART OF A ROOT GEOMETRY WITH A SIMPLE TRIPOLI-4 GEOMETRY

In the case of a Root visualization, it is possible to use a trivial Tripoli-4 geometry to hide the Root geometry located between the user and the Tripoli-4 geometry (figure 38.9).

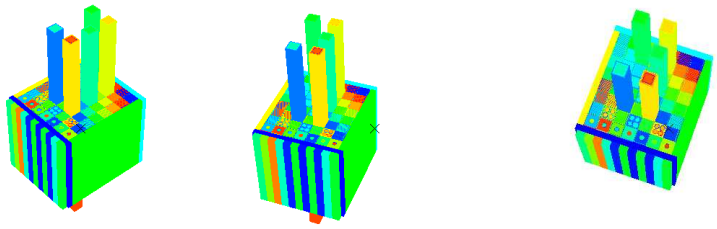


Figure 38.8: The geometry from different points of view in 3D

This option is activated with key `H`.

In this case, T4G looks for a file named `data_set.hide.t4` in the current working directory. If found, the Tripoli-4 geometry contained in this file is loaded.

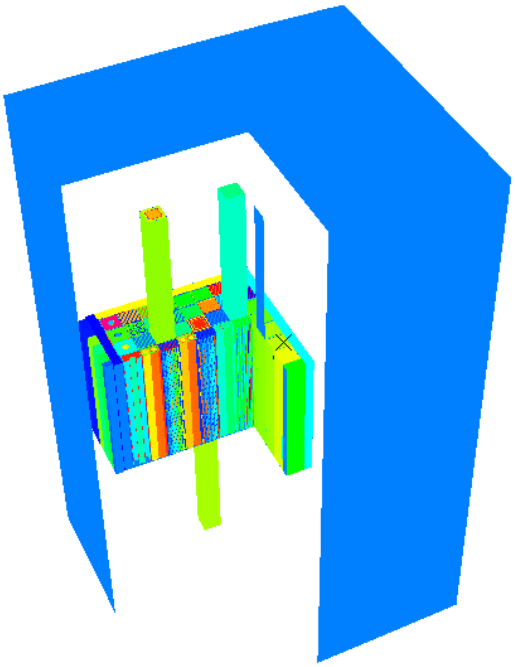


Figure 38.9: 3D Root visualization: two planes, defined by a Tripoli-4 geometry, hide a part of the Root geometry

### 38.4.4 VIEW AND SCREEN

3D visualization requires the definition of a "view" and a "screen". Those parameters are equivalent to the "window" in 2D.

#### 38.4.4.1 View

The view defines the position from where the geometry is observed.

The required informations to define it are:
- o : the position;
- theta : the theta angle;
- phi : the phi angle;
- length : the distance to the geometry;

T4G admits two ways to define a view:

- loading the file `data_set.view` from the current working directory;
- using the default view (if no file `data_set.view`);

The default view characteristics are:
- o : [0., 0., 0.];
- theta : 0.;
- phi : 0.;
- length : 200.;

The matching `data_set.view` file would contain :

```
o 0 0 0
theta 0
phi 0
length 200
```

#### 38.4.4.2 Screen

The screen contains the view port properties.

The required informations to define it are:
- screen_distance : the distance;
- pitch : the pitch;
- ilen : the width;

- jlen : the height;

T4G admits two ways to define a screen:

- loading the file `data_set.screen` from the current working directory;
- using the default screen (if no file `data_set.screen`);

The default screen characteristics are:
- screen_distance : 30. cm;
- pitch : 0.025 cm.
The matching `data_set.view` file would contain :

```
screen_distance 30
pitch 0.025
ilen 769
jlen 769
```

### 38.4.5 AVAILABLE KEYS IN 3D MODE

3D features are given in the following tables:

| **Key** | **General** |
|---|---|
| q or Q | Quit |
| space or a | Redraw |
| ? | Display the help |
| F | Export image : generates .tiff and .bmp pictures of the T4G window. |

| **Key** | **Drawing** |
|---|---|
| M | Toggle between drawing methods (only "volume simple" at the moment) |
| - | Exclude an object |
| H | Use the Tripoli-4 hiding geometry (if there is one!). Root visualization only. |

| **Key** | **Zoom** |
|---|---|
| d | Unzoom |
| D or Shift+- | Unzoom more |
| s | Zoom |
| S or Shift++ | Zoom more |

| Key | Move |
|---|---|
| l or right | Rotation towards phi - |
| h or left | Rotation towards phi + |
| k or down | Rotation towards theta - |
| j or up | Rotation towards theta + |
| r | Rotation defined by mouse position |
| c | Redefine the center of the plot |

## 38.5 CONFIGURATION FILES

Once T4G is opened, the user can manipulate the geometry by typing keys, but also by passing T4G a command file. By using the `--fromfile` option, the user can define a commands file that contains a list of commands to manipulate the geometry. This file must validate the following conditions:

- it must contain one and only one command per line;
- each special character must be written in english, matching the correspondances below:

| Key | Command |
|---|---|
| *space* | `space` |
| ? | `question` |
| * | `asterisk` |
| , | `comma` |
| < | `less` |
| > | `greater` |
| ! | `exclam` |
| _ | `underscore` |

- when T4G reaches the end of the file, T4G closes. In order to get control of T4G after a key sequence, add the command `g` to the command file. When T4G reaches a `g` key, it stops the configuration file control, and the user gets back the control of the geometry.
- the commands do not require some user's action such as keyboarding.

# Chapter 39

# T4TOMED TOOL

T4toMED is a tool to create a MED format file from one or several T4G input files.
More information on MED format is available on the SALOME Platform website at the following URL: http://www.salome-platform.org/user-section/about/med.

## 39.1 STARTING WITH T4TOMED

### 39.1.1 HOW TO LAUNCH T4TOMED

In order to translate several `*.general` T4G input files, a file suffixed by `.all` must be created as input file for T4toMED containing the list of T4G input files:

```
name      <datafile_1>.general
name      <datafile_2>.general
...
```

The syntax to run T4toMED with this data is the following:

```
static_medtripoli4 <listfile>.all
```

If a single T4G input file has to be translated, it is possible to use it as an input file for T4toMED:

```
static_medtripoli4 <T4Gfile>.general
```

The created file name is by default `<listfile>.all.med` or `<T4Gfile>.general.med` in the case of a single T4G input file.

### 39.1.2 T4TOMED OPTIONS

When using a `.all` input file listing several `*.general` T4G input files, a mesh is created for each file with the following parameters by default:

- Mesh name: `Mesh_<n>`
- Field name: `Field_<n>`

with `n` the incremented index. It is possible to group several `*.general` files in a single mesh. In this case, the `mesh` option allows the user to specify the common mesh name. As there is no check on the identity of meshes with the same name, it is the user's responsibility to ensure that the meshes with a common name are the same.
The `field` option allows the user to specify the field name corresponding to a `.general` file. As there is no check on the uniqueness of field names, it is the users's responsibility to ensure that a field name is not used more than once.

Example:

```
name     <datafile_1>.general   mesh:MyMesh field:Score
name     <datafile_2>.general
name     <datafile_3>.general   mesh:MyMesh field:Sigma
```

The limit size for the mesh and field names is `MED_TAILLE_NOM` i.e. 32 characters.

### 39.1.3 PARTICULAR DATA

Until version 9 of TRIPOLI-4, data about the frame type (`CARTESIAN` or `TRIANGLEZ`), origin and defining vectors are not specified in T4G input files. These data are now available for meshes defined with `EXTENDED_MESH` and are written as comments in the T4G input file. Example:

```
header = marker "Data\n"
format = ascii
field = importance
structure = scalar
type = float
dependency = connections
grid = 19 x 21 x 21
majority = column
positions = regular, regular, regular, 9.9, 5.1, -50.1, 5.1, -50.1, 5.1
end
#frame_type = TRIANGLEZ
#origine = 0, 0, 0
#vectors = 1, 0, 0, -0.45, 0.8, 0, 0, 0, 1
Data
12.0587
...
```

When these data are missing from the `.general` file, the frame parameters set by default are:

- frame_type: `CARTESIAN`

- origine: `0, 0, 0`
- vectors: `1, 0, 0, 0, 1, 0, 0, 0, 1`

It is then possible to add these data manually, to describe for instance a triangular mesh when using `EXTENDED_MESH`.

# Chapter 40

# DEPLETION CALCULATION

Depletion calculations allow computing $k_{effective}$, burnup, isotopic inventories and derived physical quantities (such as mass or activity) after irradiation and/or decay period. This is done by solving the Boltzmann / Bateman coupled system of equations. Bateman's equations are solved by the MENDEL [48] depletion solver, and the neutron flux is estimated by regular Monte Carlo simulation. In Monte Carlo burnup calculations, statistical uncertainties propagation has to be taken into account, which is achieved by running independent simulations. The procedure for managing independent simulations is automatized via a dedicated launcher (see section 40.2) and distinct post-processing tools (see section 40.5).

## 40.1 DEPLETION CALCULATION INPUT FILE

An input deck for a depletion calculation is structured as follows :

- a TRIPOLI-4 static input file with some modifications explained below
- an input file containing the `COMPOSITION` block to specify the initial compositions
- an input file containing the `GRID_LIST` block
- an input file to specify the irradiation history

### 40.1.1 TRIPOLI-4 INPUT FILE

Starting from a static TRIPOLI-4 input file, some modifications are needed (in addition to the `DEPLETION` block, which will be detailed below).

First, the `COMPOSITION`, `GRID_LIST`, `RESPONSES` and `SCORE` blocks have to be replaced by `FILE` commands pointing to specific files :

- `librarypath` is the path to the Mendel library file (`MendelLibrary_V0.1.BBL`), which is installed along with TRIPOLI-4. The `MENDEL_LIBRARY` line is not mandatory, but it is required for the calculation of the activities in Bq per $cm^3$;

```
FILE     directorypath/result/compos.jdd
FILE     directorypath/decoupageinit.jdd
FILE     directorypath/result/reponses.jdd
FILE     directorypath/result/scores.jdd
```

- *directorypath* is the path where the input file is located. It is specified by the DEPLETION_DIRECTORY keyword in the DEPLETION block;
- *result* is the output directory. It is specified by the RESULT_DIRECTORY keyword in the DEPLETION block. By default it is called “result";
- The file *compos.jdd* contains the COMPOSITION block describing all the materials of the system (depleted and non depleted). During the initialization, this file is automatically generated in the output directory from the *composinit.jdd* user input file (see 40.1.3);
- The file *decoupageinit.jdd* contains the GRID_LIST block (see 40.1.2);
- The files *reponses.jdd* and *scores.jdd* contain respectively the RESPONSES and SCORE block. These files are automatically generated in the output directory from the list of depleted compositions, the depletion chain and reaction rates calculation options.

Depletion calculation parameters are specified via the DEPLETION block as shown on the next page:

**DEPLETION**
*MENDEL_LIBRARY* **librarypath**
**DEPLETION_DIRECTORY directorypath**
*TEMPORAL_SCHEME <SCHEME>*
*SCRIPTS_DIRECTORY scriptdirectory*
**IRRADIATION_HISTORY <UNIT> ihfilename**
*Specification of the file describing irradiation history*
**RENORMALIZATION**
*Data related to renormalization (flux level calculation)*
**END_RENORMALIZATION**
**COMPUTE_ENERGY**
*Data related to the deposited energy calculation method*
**DEPLETED_COMPOSITIONS**
*List of depleted compositions*
**CHAIN**
*Depletion chain / depleted compositions association*
***POINTWISE_REACTION_RATES***
*Selection of nuclides whose reaction rates are computed directly by TRIPOLI-4*
***GENDF***
*GENDF cross-sections / depleted compositions association*
***VOLUMES***
*Specification of the volumes (in $cm^3$) of depleted compositions*
***ESTIMATOR <TYPE>***
*Estimator to be used for computing flux and reaction rates (track length or collision estimators)*
***ISOTOPIC_FILTERS***
*Isotopic filters to remove nuclides from transport calculation*
***RESULTS***
*Files and directory results specifications*
***END_RESULTS***
**END_DEPLETION**

where:

- `directorypath` is the calculation directory (either an absolute or a relative path);
- ***<SCHEME>*** is either `EULER` (by default), `MEAN` or `MIDPOINT`. The optional `TEMPORAL_SCHEME` command is described below.
- *`scriptdirectory`* is the directory for the user scripts (a relative path). Full directory path is therefore *directorypath/scriptdirectory*. The optional `SCRIPTS_DIRECTORY` command is described below.

- <**UNIT**> is either `TIME` if the time discretization is defined in days, or `BURNUP` if it is defined in MWd/t (Megawatts day per initial heavy nuclei ton).
- `ihfilename` is the name of the file containing irradiation history (see 40.1.4). The full path name is therefore *directorypath/ihfilename*.

The `TEMPORAL_SCHEME` command allows choosing between first order temporal scheme (`EULER` method, see 40.1) or second order methods (`MEAN`, see 40.2, or `MIDPOINT` method, see 40.3). Second order methods make the time discretization error decrease at the expense of doubling the calculation time. The `TEMPORAL_SCHEME` command is not mandatory: by default `EULER` scheme is used.

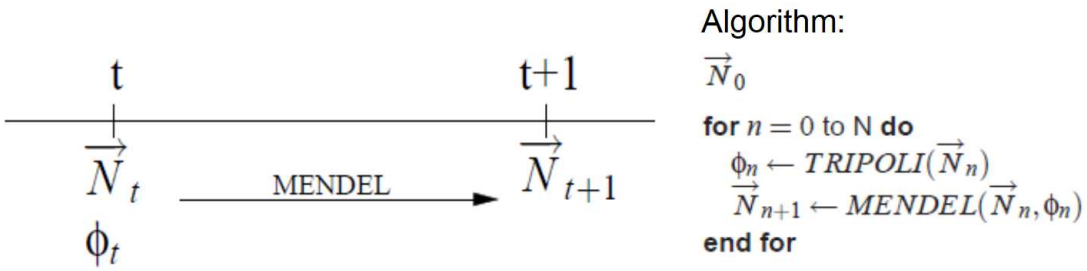



Figure 40.1: Temporal scheme: Euler

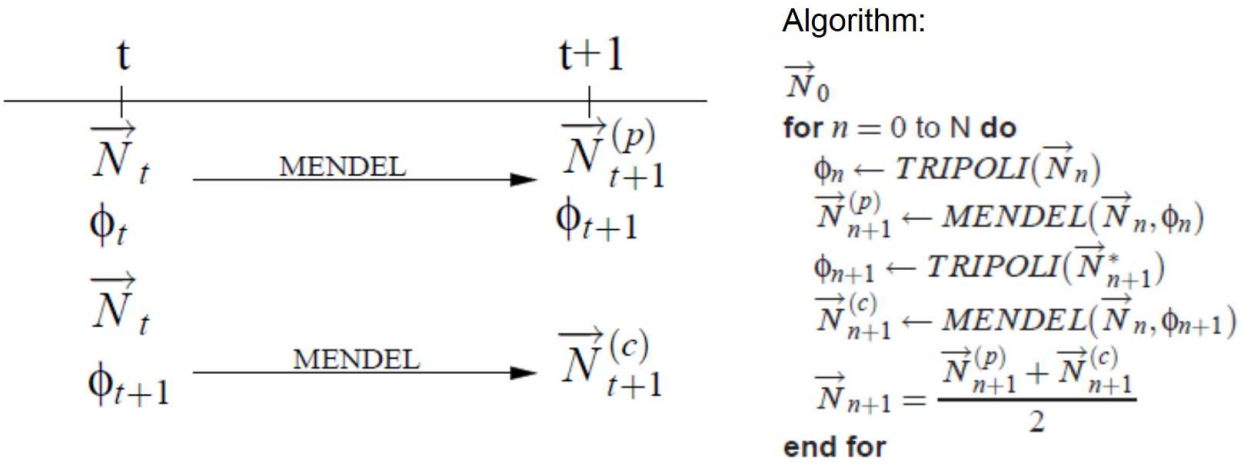



Figure 40.2: Temporal scheme: Mean

The `SCRIPTS_DIRECTORY` command can be used to adapt the default coupling script (for example to change the geometry during irradiation...).
If the temporal scheme is `EULER` (default behavior), the code will expect a script named

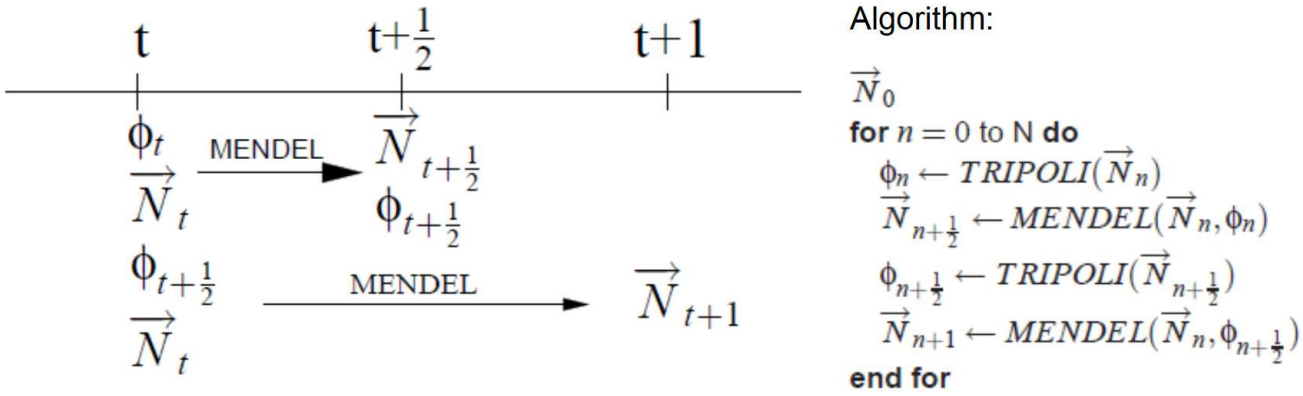



Figure 40.3: Temporal scheme: Midpoint

User_CouplageEvolution.C in the user scripts directory. If the temporal scheme is `MEAN` or `MIDPOINT`, the code will expect a script named User_Predictor.C and a script named User_Corrector.C in the user scripts directory. An example of these scripts can be found in the scripts directory of the TRIPOLI-4 distribution. The code may need to create some symbolic links to the include directory and to some scripts of the TRIPOLI-4 distribution, thus, for example the user can add a subfolder to the scriptdirectory.
With "`SCRIPTS_DIRECTORY` myscriptdir/scripts" the user will get the following directory tree :

```
myscriptdir/
|-- include -> <TRIPOLI-4 distribution path>/include
`-- scripts
    |-- AnalysisLib.C -> <TRIPOLI-4 distribution path>/scripts/AnalysisLib.C
    |-- RestartInfo.C -> <TRIPOLI-4 distribution path>/scripts/RestartInfo.C
    |-- User_CouplageEvolution.C
    `-- Utils.C -> <TRIPOLI-4 distribution path>/scripts/Utils.C
```

The `RENORMALIZATION` command allows choosing a method to normalize the flux and the reaction rates to the desired flux level. The user can choose one of the four methods: `PWPG`, `POWER`, `FLUX` or `FIXED_SOURCE`.

**RENORMALIZATION**
* **PWPG** fuelpowerdensity
* **POWER** power
* **FLUX** fluxintensity
* **FIXED_SOURCE**
***LIST***
* ***VOLU*** *nb_volu* $volu_1$ .... $volu_n$
* ***COMPOSITION*** *nb_compo* $compo_1$ .... $compo_n$
**END_RENORMALIZATION**

Where:

- PWPG: Renormalization to the power expressed in $W/g_{heavy_nuclides}$. Only depleted compositions may be selected for the calculation of the power and the initial mass of heavy nuclides. By default, without specification, the normalization is performed over all depleted compositions.
- POWER: Renormalization to the power (in $W$). The power is computed as the sum of the power of the selected depleted compositions. By default, without specification, the normalization is performed over the sum of all depleted compositions powers.
- FLUX: Renormalization to the flux intensity specified by fluxintensity. This flux corresponds to the total flux cumulated over the selected compositions or volumes (but not limited to depletion regions) integrated in both energy and space. By default, without specification, the normalization is performed over the whole geometry. The GLOBAL_SCORES keyword has to be added in the SIMULATION block.
- FIXED_SOURCE: No renormalization is performed, the flux level is defined via the norm specified in the SOURCE block.

When considering a normalization using PWPG, POWER, FLUX, the user may choose to select the volumes or compositions on which the normalization is performed using the keyword LIST. The selected volumes (or compositions) must have compositions that belong to the depleted composition list when using the keywords PWPG and POWER. If no selection of volumes or compositions is provided by the user, the normalization is performed over all depleted compositions for the PWPG and POWER normalization methods, and the whole geometry for the FLUX method. Below is an example of a normalization using the power over three volumes.

```
RENORMALIZATION
 POWER 3140 LIST VOLU 3 FUEL_1 FUEL_2 FUEL_3
FIN_RENORMALIZATION
```

The COMPUTE_ENERGY command allows choosing the method used to compute the deposited energy. The syntax is as follows:

```
      COMPUTE_ENERGY
*           ENERGY_FILE energyfilename
*           DEPOSITED_ENERGY
```

Where:

- ENERGY_FILE : The deposited energy is calculated from fission and capture reaction rates, and from the energy released by these reactions. Those energies are specified in the file energyfilename.
- DEPOSITED_ENERGY : the energy deposited **in all the depletion regions** is calculated from the DEPOSITED_ENERGY score. Non-depleted compositions therefore do not contribute (the moderator for example).

The DEPLETED_COMPOSITIONS command allows specifying the list of the depleted compositions. The syntax is as follows:

**DEPLETED_COMPOSITIONS**
**LIST** n *COMPO1 ... COMPOn*

The POINTWISE_REACTION_RATES command is used to specify the nuclides for which all the reaction rates are directly calculated by TRIPOLI-4. The reaction rates of the other nuclides are computed from multi-group cross-sections (GENDF, Groupwise Evaluated Nuclear Data File) and the multi-group flux (defined on the same energy grid). This introduces an approximation but reduces sensibly the computation time. It is recommended to compute at least U238 reaction rates directly by TRIPOLI-4. This choice of nuclides applies to all the depleted compositions. The POINTWISE_REACTION_RATES command is optional; **by default, all the reaction rates (defined in the depletion chain) are directly calculated by TRIPOLI-4**. The syntax is as follows:

```
      POINTWISE_REACTION_RATES
*           ALL
*           LIST n NUC1 ... NUCn
*           FILE nuclide_list_file
```

Where:

- ALL : All the reaction rates are calculated by TRIPOLI-4. Thus, the number of responses will be typically high ($\sim$ thousands). The use of this option will typically sensibly increase the duration of the neutron transport simulation.
- LIST : For the *n* nuclides specified in this list, reaction rates are calculated by TRIPOLI-4. LIST 0 allows reconstructing all the reaction rates from the multi-group flux and GENDF cross-sections.
- FILE : If the nuclide list is too long, it is possible to write it in the file *nuclide_list_file*. No command should be specified in it.

The GENDF command allows associating each depleted composition to GENDF cross-sections. This is only used for the reconstruction of reaction rates that are not calculated directly by TRIPOLI-4. The GENDF command must be placed after the DEPLETED_COMPOSITIONS command. Each GENDF file is produced by the GROUPR module of the NJOY code for a given temperature and weighting flux. Associating different GENDF to depleted compositions allows using:

- multi-group cross-section at different temperatures

- multi-group cross-sections weighted by a flux depending on the depletion region.

If all the reactions rates are calculated directly by TRIPOLI-4, this command is not necessary. The syntax is as follows:

```
GENDF num(gendf)
        path_gendf_file1 path_corresp_gendf_file1
        LIST M COMPO11 ... COMPO1M
        path_gendf_file2 path_corresp_gendf_file2
        LIST P COMPO21 ... COMPO2P
```

Where :

- `num(gendf)` : number of different GENDF files
- `path_gendf_file1`: absolute path of the file that contains multi-group cross-sections at a given temperature.
- `path_corresp_gendf_file1`: absolute path of the file that establishes the connection between nuclides and NJOY name and specifies the available reactions for each nuclide.
- `M`: Number of depleted compositions for which reaction rates have to be reconstructed from the GENDF file previously specified.
- `COMPO11 ... COMPO1M`: List of M depletion compositions for which reactions rates have to be reconstructed from the GENDF file previously specified.

The GENDF energy grid has to match the `DEC_FLUX_MULTIG` energy grid specified inside the file *decoupageinit.jdd* (see 40.1.2). Each depletion composition has to be associated with its own GENDF file. If only one GENDF file is used for all the depleted compositions, the following syntax can be used:

```
GENDF
    1
        path_gendf_file
        path_corresp_gendf_file
    ALL
```

The `CHAIN` command follows the same syntax as the `GENDF` command and should also be placed after the `DEPLETED_COMPOSITIONS` command:

```
CHAIN num(chain)
      chain_file1
      LIST M COMPO11 ... COMPO1M
      chain_file2
      LIST P COMPO21 ... COMPO2P
```

Where:

- `num(chain)`: Number of different depletion chains
- `chain_file1`: File containing the depletion chain description using the MENDEL depletion chain syntax.
- `M`: Number of depletion regions that will evolve according to the depletion chain previously specified.
- `COMPO11 ... COMPO1M`: List of the `M` depletion regions that will evolve according to the depletion chain previously specified.

Each depleted composition has to be associated with a depletion chain. If only one depletion chain is used, then it is possible to use the following syntax:

```
CHAIN
     1
          chain_file
     ALL
```

The `ISOTOPIC_FILTERS` command is optional. It allows defining filters on nuclides to be taken into account during the transport calculation. These filters are applied at each time step, on each depleted composition. It allows saving calculation time. Three kinds of filter are available:

- MIN_CONC: A filter on the minimal concentration required for a nuclide to be taken into account during the transport step. This filter is actually applied by default with a value of $10^{-20}$ atoms per $10^{-24} cm^3$. This keyword can be used to modify the default threshold value.
- ABSORPTION_RATIO: A nuclide is taken into account in the transport calculation if the ratio of its macroscopic disappearance rate over the one of the composition is larger than the threshold set by the user.
- FISSION_RATIO: Same behaviour as the previous filter, but applied to macroscopic fission.

The syntax is as follows:

```
ISOTOPIC_FILTERS n
        MIN_CONC conc
        ABSORPTION_RATIO abs_ratio
        FISSION_RATIO fiss_ratio
```

Where :

- n: number of isotopic filters
- `MIN_CONC` : Allows redefining the threshold value of the minimal concentration filter. This threshold will be set to *conc*.
- `ABSORPTION_RATIO` : Allows setting the threshold value of the ratio to *abs_ratio*.
- `FISSION_RATIO` : Allows setting the threshold value of the ratio to *fiss_ratio*.

Minimal concentration filter is applied first. The remaining isotopes are selected for the transport step if they satisfy at least one of the two filters (ABSORPTION or FISSION).

The volume of each depleted composition is mandatory since reaction rates or flux estimated by TRIPOLI-4 are integrated over the volume, while the MENDEL depletion solver needs volumic reaction rates. The volumes can be automatically computed by TRIPOLI-4 either for a native or a ROOT geometry (`CALCULATION` block has to be used without the `SET` keyword, see Chapter 19 for more details). The **geometrical frame must include all the depletion regions**. The automatic calculation of the volumes can lead to large computation time. Consequently, it is recommended to provide the volumes (in $cm^3$) for all depleted compositions whenever possible. This can be done by the optional command `VOLUMES`.

```
        VOLUMES
*           LIST n
                COMPO1 vol1
                COMPOn voln
*           FILE volumefilename
```

`FILE` keyword allows specifying the file name containing the association depleted regions / volumes. If `VOLUMES` command is not present inside the `DEPLETION` block, *repertorypath/volumes* is taken as default file name. The syntax of this file is the following:

```
COMPOS1         value1
.
.
COMPOSN         valueN
```

The optional keyword ESTIMATOR allows specifying the estimator used by TRIPOLI-4 for multi-group flux and reaction rates calculation. Two estimators are available: TRACK and COLL.

The RESULTS block allows changing some options concerning the results such as the ROOT output file name or the output directory name. The syntax is as follows:

```
RESULTS
        RESULT_DIRECTORY resultdirectoryname
        RESULT_FILE resultfilename.root
        T4LISTING t4listingname
        SAVE_T4LISTING 0 or 1
        WRITE_FLUX 0 or 1
END_RESULTS
```

Where:

- RESULT_DIRECTORY : Output directory name. Output files will be located in *directorypath/resultdirectoryname*.
- RESULT_FILE : ROOT file name (with ".root" extension). The full path of this file is *directorypath/resultdirectoryname/resultfilename.root*.
- T4LISTING : Name of the TRIPOLI-4 output file produced at each transport step. The full path of this file is *directorypath/resultdirectoryname/t4listingname*.
- SAVE_T4LISTING : If 0, the TRIPOLI-4 transport output file is overwritten at each time step. If 1, the TRIPOLI-4 transport output file is saved at each time step (files *t4listingname_stepnum* located in the output directory). By default, this option is set to 0.
- WRITE_FLUX : If 0, the multi-group flux for each depleted region at each time step is not written in the ROOT output file. If 1, it is saved. By default, this option is set to 1, but setting it to 0 allows saving disk space for large system simulations and/or large number of independent simulations.

### 40.1.2 ENERGY GRIDS FILE

The file *decoupageinit.jdd* contains the GRID_LIST block specifying all the energy grids required to compute one-group reactions rates (DEC_1G), two-group fission rates (DEC_2G) and the multi-group fluxes (DEC_FLUX_MULTIG). The syntax is as follows:

```
GRID_LIST    3
   DEC_1G  2 2.0000E+01  1.0000E-11
   DEC_2G  3 2.0000E+01  0.4  1.0000E-11
   DEC_FLUX_MULTIG  4  20.   1.   4.E-7 1.E-11
END_GRID_LIST
```

The name of these three energy grids (DEC_1G , DEC_2G and DEC_FLUX_MULTIG) must not be modified. If the DEC_FLUX_MULTIG energy grid is modified, it has to be coherent with the GENDF energy grid. By default, DEC_FLUX_MULTIG has 11515 groups.

### 40.1.3 INITIAL COMPOSITIONS FILE

This file contains the COMPOSITION block in which all the initial compositions of the system (depletion regions or not) are specified. It follows the syntax described in Chapter 10 except that depleted compositions can only be defined with the POINT_WISE keyword (in $10^{24}$ atoms per $cm^3$) or with the DENSITY keyword (density in $g \cdot cm^{-3}$ and the mass fraction of each isotope).
At initialization step, depleted compositions described with the DENSITY keyword (if any) are converted to atoms per $cm^3$ (POINT_WISE keyword). This file is stored in the result directory (*resultdirectoryname/compos.jdd*). It will be updated at each depletion step.

### 40.1.4 IRRADIATION HISTORY FILE

This file, whose name is specified by the command IRRADIATION_HISTORY, allows defining the time or burnup discretization and possibly the duration of decay periods. The syntax is the following:

```
FACT  fact  STEP  t1  t2  t3  REPEAT N t4  DECAY  dec1
```

The FACT keyword followed by a positive real number allows specifying the fraction of the flux level (defined using the RENORMALIZATION command) that will be applied to the steps following the next STEP keyword. t1, t2, t3 are the **durations** of the steps (either in $MWd/tHN$ or days depending on the units chosen in the IRRADIATION_HISTORY command). The REPEAT keyword followed by the number of occurrences and the duration of the steps can be used to simplify the syntax.
The optional DECAY keyword is used to perform a decay period. The duration of this decay period is given in days. The duration after a DECAY keyword can be followed by another DECAY keyword or FACT keyword (even if the fraction of the flux level is unchanged).
The following irradiation history example is represented on FIG. 40.4.

```
FACT  1  STEP  2
FACT  0.5  STEP  REPEAT  2  1
FACT  0.8  STEP  2  DECAY  2
FACT  1  STEP  2
```

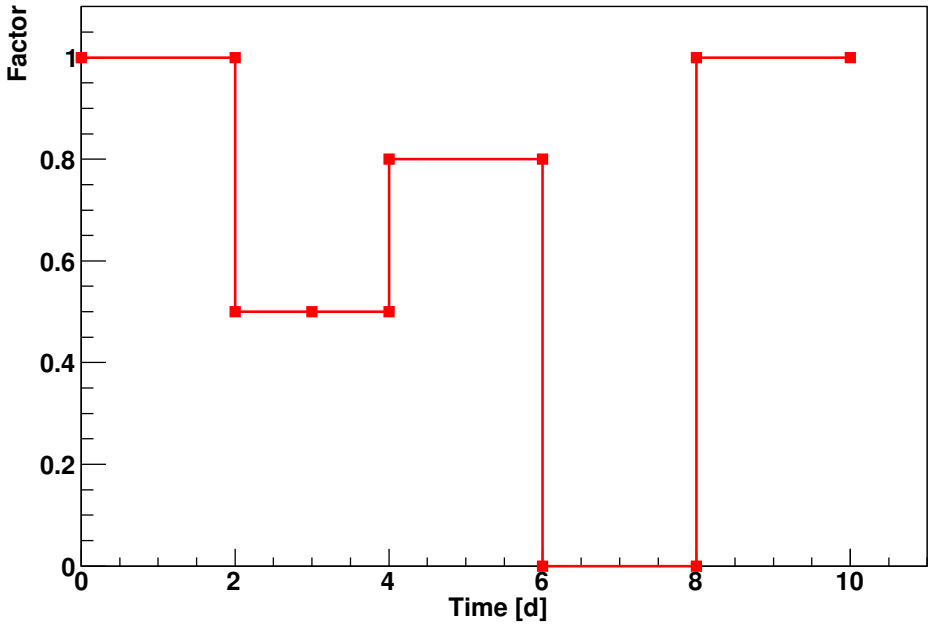



Figure 40.4: Irradiation history example

## 40.2 LAUNCHING INDEPENDENT SIMULATIONS

In Monte Carlo burnup calculations, statistical uncertainties propagation has to be taken into account, which is achieved by running independent simulations. The procedure for managing independent simulations is automatized via a dedicated launcher that allows launching them in parallel.

A depletion calculation input deck must be modified as follows :

1. **a *resultdirectoryname* must be specified** with the `RESULT_DIRECTORY` command in the `RESULTS` block
2. **no random seed must be set** in the `SIMULATION` block

Independent simulations launchers named “launch_jobT4_**IndepSimu**_<arch>.sh" can be found in the “launcher" directory of the TRIPOLI-4 distribution. “arch" depends on the hardware architecture used. The following architectures are available: workstations, slurm for clusters using SLURM or gridengine for clusters using SGE.

The use is :

```
launch_jobT4_IndepSimu_workstations.sh <machines-file> <nbsimu> <startrank>
               -d <inputfile> [-s <format>] [-c <pathfile>] [-l <language>] [other T4 options...] ...

launch_jobT4_IndepSimu_slurm.sh <nbproc> <nbsimu> <startrank> <time>
               -d <inputfile> [-s <format>] [-c <pathfile>] [-l <language>] [other T4 options...] ...
```

where, concerning the first group of options :

- machines-file (workstations case) (mandatory) is the name of a file containing as many network hostname as the number of independent simulations the launcher will then have the possibility to launch
- nbproc (cluster case) (mandatory) is the number of processors the launcher will ask
- nbsimu (mandatory) is the number of independent simulations (nbsimu <= nbproc)
- startrank (mandatory) is the rank of the first independent simulation to launch (useful if the desired nbsimu is greater than the number of processors available at a time)
- time (mandatory) (cluster) is the total duration of the job on a cluster

and concerning the second group of options :

- -d <inputfile> (mandatory) is the input deck with the modifications mentioned above
- -s <format> (optional) is the format for the cross-sections. Default is NJOY.
- -c <pathfile> (optional) is the file containing the access paths for the nuclear data required by the code. Default file is the one configured with the distribution.
- -l <language> is used to indicate the language used when entering the keywords. The default value is “old".
- see chapter 1 for other options

First, a pretreatment step creates as many input decks and result directories as the number of independent simulations. Such a created input deck is characterized by its own random seed and by its own *resultdirectoryname*. Created input decks and result directories are identified with their simulation rank.
Then, independent simulations are launched in parallel. The results can be post-processed using tools described in section 40.5.

## 40.3 RESTART CALCULATION

It is possible to restart a calculation from an arbitrary depletion step. This is one of the reasons why *compos.jdd* and *compos_mendel.jdd* files are saved at each depletion step. The procedure to restart from the $N^{th}$ depletion step is the following:

1. it is recommended to save the output directory

2. in the output directory, replace the file *compos.jdd* by the file *compos_N.jdd*
3. in the output directory, replace the file *compos_mendel.jdd* by the file *compos_mendel_N.jdd*
4. use the script *Restart.py* to remove results corresponding to depletion steps higher than *N* in the ROOT output file and to produce the *restart_info* file.
5. replace the current ROOT output file and the *restart_info* files by the new *restart_info* and ROOT output file produced by the script *Restart.py*.
6. Launch the simulation

If the simulation stopped accidentally (simulation time exceeds the time requested to the cluster scheduler, ...), in most cases, it is possible to restart the calculation just by launching again the calculation. The simulation should restart from the current depletion step. The only exception concerns the calculations that have not stopped during the transport calculation (most of the time is spent in this step). It is recommended to restart from the previous step by resorting to the procedure detailed above.

## 40.4 USING THE SOURCE FROM THE PREVIOUS DEPLETION STEP

Since version 8.1, it is possible to store source particles in an ASCII or binary file and use these particles as a discrete source for another calculation. This functionality can be adopted to use the source of the last batch of the previous depletion step as a source for the current depletion step. The process has to be initialized by an extra calculation to produce the file for the first depletion step using the keyword `STORE_SOURCES_IN_FILE` in the `SIMULATION` block. Then, it is just necessary to have `STORE_SOURCES_IN_FILE` command in the `SIMULATION` block and also a source using this file. The source file is automatically updated at each time step.

## 40.5 RESULTS AND POST-PROCESSING

All the results of a depletion calculation are stored in an output directory whose name is specified by the `RESULT_DIRECTORY` keyword.

### 40.5.1 CONTENT OF THE OUTPUT DIRECTORY

The output directory of a depletion calculation is composed of the following files:

- the ROOT output file aggregating all the simulation results (see 40.5.2)
- the file *compos_mendel.jdd* containing the `COMPOSITION` block of the last depletion step. This file is updated at each depletion step.
- the files *compos_mendel_i.out* (where i is depletion step index) are copies of the file *compos_mendel.jdd* made at each depletion step.

- the file *compos.jdd* containing the `COMPOSITION` block of the last depletion step. This file is updated at each depletion step. It contains only nuclides that have not been removed from the transport calculation (see `ISOTOPIC_FILTERS` command).
- the files *compos_i.out* (where i is depletion step index) are copies of the file *compos.jdd*. This allows easily performing a static calculation at a specific time step.
- the files *reponses.jdd* and *scores.jdd* respectively contain the `RESPONSES` and `SCORE` blocks. They are automatically built from the depletion chain, the list of depleted compositions and the reaction rates calculation options.
- the output of the transport calculation (*t4output* file). By default, only the last depletion step file is preserved. This file can be saved at each depletion step by resorting to the `SAVE_T4LISTING` keyword.

### 40.5.2 CONTENT OF THE ROOT OUTPUT FILE

At each depletion step, results are stored in two ROOT tree structures (TTree class): a tree called *evoltree* containing all the depletion results and a tree called *evolflux* containing the multi-group flux of each depleted composition (if the option `WRITE_FLUX` is enabled).
The *evoltree* tree is composed of branches, each attached to a physical quantity:

- step_num: the depletion step index
- burnup: the mean burnup of the system in $MWd \cdot t_{HN}^{-1}$
- cumulirradtime: the cumulative irradiation time in s
- irradtime: the duration of the next depletion step in s
- kcoll: $k_{eff}$ (collision estimator)
- ktrack: $k_{eff}$ (track length estimator)
- kstep: $k_{eff}$ (generation estimator)
- beff_prompt: $\beta_{eff}$ computed by the "prompt" method, in $10^5$ pcm
- beff_nauchi: $\beta_{eff}$ computed by the "Nauchi" method, in $10^5$ pcm
- total_powereffect: the power cumulated on all the depletion regions expressed in W
- renormfact: the factor by which the reaction rates and fluxes are multiplied to obtain the normalization specified with the `RENORMALIZATION` command.
- compo_name: the name of the current depletion region
- compo_vol: the volume in $cm^3$ of the current depletion region
- local_burnup: the burnup of the current depletion region in $MWd/t_{HN}$

- therm_flux: the flux in the current depletion region in the first energy group of DEC_2G in neutrons $\cdot$ cm$^{-2} \cdot$ s$^{-1}$
- fast_flux: the flux in the current depletion region in the second energy group of DEC_2G in neutrons $\cdot$ cm$^{-2} \cdot$ s$^{-1}$
- powereffect: the power in the current depletion region in W
- isotope_name: the current nuclide name
- A: the mass number of the current nuclide
- punctual (pointwise in activation calculations; see Chapter 41): specifies whether the reaction rates of the current nuclide are computed directly by TRIPOLI-4 (1) or if they are computed from the multi-group flux and the GENDF cross-sections
- transport: specifies whether the current nuclide has been taken into account in the transport calculation (1)
- isotope_conc: concentration of the current nuclide in atoms per $10^{-24}$cm$^{-3}$
- mass: mass of the current nuclide in g
- activity: activity of the current nuclide in Bq per cm$^3$
- reaction_name: the name of the current reaction
- reactionrateg1: microscopic and volumic reaction rate of the current reaction. Fast energy group if the current reaction is fission, energy integrated for all other reactions.
- reactionrateg2: microscopic and volumic reaction rate of the current reaction. Thermal energy group if the current reaction is fission, zero for all other reactions.

If the WRITE_FLUX keyword is specified in the DEPLETION block, the *evolflux* tree containing the flux computed on the DEC_FLUX_MULTIG energy grid for each depleted composition. It has the following structure:

- stem_num: the depletion step index
- renormfact: the factor by which the reaction rates and fluxes are multiplied to obtain the normalization specified with the RENORMALIZATION command.
- compo_name: the current depleted composition name
- compo_vol: the volume in cm$^3$ of the current depleted composition
- goup_id: index of the current energy group of the DEC_FLUX_MULTIG energy grid
- energy1: lower bound of the current group in eV
- energy2: upper bound of the current group in eV
- flux_val: flux value in the current group in neutrons $\cdot$ cm$^{-2} \cdot$ s$^{-1}$

Observe that this tree requires a large disk space for large systems. In these cases, it is recommended to set WRITE_FLUX to 0.

### 40.5.3 POST-PROCESSING

The post-processing of a TTree structure is explained in great detail in the ROOT user's guide. It allows easily plotting physical quantities as a function of the time or burnup for one depletion calculation result, i.e. for one independent simulation. However, it is difficult to post-process several independent simulations. To this aim, post-processing tools, consisting mainly of two C++ classes, *BurnupResults* and *MeanBurnupResults*, are available in TRIPOLI-4 and may be called either from a ROOT C++ script or a Python script.

#### 40.5.3.1 Post-processing a unique ROOT output file using *BurnupResults* class

Building a *BurnupResults* object requires only the path to the ROOT output file to be post-processed. Once the object is built, the following methods allow obtaining the corresponding physical quantities at any depletion step:

For global physical quantities:

- GetKcoll(int step)
- GetKtrack(int step)
- GetStep(int step)
- GetBeffPrompt(int step)
- GetBeffNauchi(int step)
- GetBurnup(int step)
- GetTime(int step)
- GetRenorm(int step)
- GetTotalPower(int step)

For the local physical quantities (i.e. for each depletion region):

- GetPower(int step, string componame)
- GetLocalBurnup(int step, string componame)
- GetFastFlux(int step, string componame)
- GetThermFlux(int step, string componame)
- GetMass(int step, string componame, string isotope_name)
- GetConcentration(int step, string componame, string isotope_name)
- GetActivity(int step, string componame, string isotope_name)
- GetReactionRate(int step, string componame, string isotope_name, string reaction_name)

The following helper functions are also available:

- GetSteps() to obtain the number of depletion steps
- GetNbCompos() to obtain the number of depleted compositions
- GetDepletedCompositionName(int index) get the name of the $i^{th}$ depleted composition

It is also possible to directly plot the physical quantities listed above as a function of either burnup or time by resorting to the following methods:

- GetKcollGraph(int index = 0)
- GetKtrackGraph(int index = 0)
- GetStepGraph(int index = 0)
- GetBeffPromptGraph(int index = 0)
- GetBeffNauchiGraph(int index = 0)
- GetRenormGraph(int index = 0)
- GetTotalPowerGraph(int index = 0)
- GetPowerGraph(string componame, int index = 0)
- GetLocalBurnupGraph(string componame, int index = 0)
- GetFastFluxGraph(string componame, int index = 0)
- GetThermFluxGraph(string componame, int index = 0)
- GetMassGraph(string componame, string isotope_name, int index = 0)
- GetConcentrationGraph(string componame, string isotope_name, int index = 0)
- GetActivityGraph(string componame, string isotope_name, int index = 0)
- GetReactionRateGraph(string componame, string isotope_name, string reaction_name, int index = 0)

The last argument *index* specifies whether the physical quantity is plotted as a function of burnup ($index = 0$, default behavior) or time ($index = 1$). The returned object is a ROOT TGraph object that can be drawn by calling the *Draw* method. The listings 40.1 and 40.2 are respectively C++ and Python examples of post-processing scripts resorting to the *BurnupResults* class.

```
#include "PREFIX/InitPostscripts.C"

void OneSimulationAnalysis()
{
  // Load postscripts classes
  InitPostscripts();
  // Create an object BurnupResults from ROOT output file "evolution.root"
  BurnupResults b("./evolution.root");

  // Print some values at the last depletion step:
  // M(U235) in depleted composition FUEL1,
  int last_step = b.GetSteps();
  cout << b.GetMass(last_step,"FUEL_R1","U235") << endl;
  // Local burnup in depleted compositions FUEL1 and FUEL6
  cout << b.GetLocalBurnup(last_step,"FUEL_R1") << " " ;
  cout << b.GetLocalBurnup(last_step,"FUEL_R6") << endl;

  // Plot evolution of Kcoll vs. burnup
  b.GetKcollGraph()->Draw("al");
}
```

Listing 40.1: Example of C++ post-processing script using the *BurnupResults* class

```
import sys
sys.path.append("PREFIX")
from InitPostscripts import InitPostscripts
InitPostscripts()
from ROOT import DepletedComposition, BurnupResults

# Create an object BurnupResults from ROOT output file "evolution.root"
b = BurnupResults("./evolution.root")

# Print some values at the last depletion step:
# M(U235) in depleted composition FUEL1,
last_step = b.GetSteps()
print b.GetMass(last_step,"FUEL_R1","U235")
# Local burnup in depleted compositions FUEL1 and FUEL6
print b.GetLocalBurnup(last_step,"FUEL_R1"), b.GetLocalBurnup(last_step,"FUEL_R6")

# Plot evolution of Kcoll vs. burnup
b.GetKcollGraph().Draw("al")
# Necessary to prevent the canvas disappears
# (alternative: launch the script with python -i)
```

```
raw_input()
```

Listing 40.2: Example of Python post-processing script using the *BurnupResults* class

These examples of post-processing scripts can be found in the *postscripts* directory.

#### 40.5.3.2 Post-processing several ROOT output files using the *MeanBurnupResults* class

At each independent simulation is associated to a ROOT output file. The *MeanBurnupResults* class allows aggregating the results contained in each output file and computing means and their associated error bars on all the previously listed physical quantities.
Building a *MeanBurnupResults* object does not require any information. To post-process a ROOT output file it is just necessary to call the method *AddSimulationAndProcess* with the path to the ROOT output file. The interface of the *MeanBurnupResults* class is the same as *BurnupResults* except for an additional parameter (an integer) specifying the kind of value to be returned: the mean (0, default behavior), the standard deviation of the mean (1) or the relative standard deviation in % (2).
The listings 40.3 and 40.4 are respectively C++ and Python examples of scripts resorting to the *MeanBurnupResults* class to post-process 10 independent depletion calculations.

```
#include "PREFIX/InitPostscripts.C"

void IndependentSimulationsAnalysis()
{
  // Load postscripts classes
  InitPostscripts();

  const int nb_simus = 10;
  // Create an object MeanBurnupResults
  MeanBurnupResults m;
  // Add ROOT output files to the MeanBurnupResults object
  for (int i=0;i<nb_simus;i++) {
    TString filename = "evolution_";
    filename += (i+1);
    filename += ".root";
    m.AddSimulationAndProcess(filename);
  }

  // Print some mean values and their corresponding standard deviation at
  // the last depletion step:
  // M(U235) in depleted composition FUEL_R1,
  int last_step = m.GetSteps();
  cout << "M(U235) in FUEL_R1 = " << m.GetMass(last_step,"FUEL_R1","U235") ;
  cout << "(" << m.GetMass(last_step,"FUEL_R1","U235",1) << ")" << endl;
  // Local burnup in depleted compositions FUEL_R1 and FUEL_R6
  for (int i=0;i<6;i++) {
    TString med("FUEL_R");
    med += (i+1);
    cout << "LocalBurnup(" << med << ") = " << m.GetLocalBurnup(last_step,med);
    cout << "(" << m.GetLocalBurnup(last_step,med,1) << ")" << endl;
  }

  // Write kcoll results in an ASCII file
  ofstream file("kcoll.dat");
  for (int i=0;i<last_step;i++) {
    file << m.GetBurnup(i+1) << "\t" << m.GetKcoll(i+1);
    file << "\t" << m.GetKcoll(i+1,1) << endl;
  }
  file.close();

  // Plot evolution of Kcoll vs. burnup
  m.GetKcollGraph()->Draw("ale");
```

```
}
```

Listing 40.3: Example of C++ post-processing script using the *MeanBurnupResults* class

```
import sys
sys.path.append("PREFIX")
from InitPostscripts import InitPostscripts
InitPostscripts()
from ROOT import DepletedComposition, BurnupResults, MeanBurnupResults

nb_simus = 10

# Create an object MeanBurnupResults
m = MeanBurnupResults()
# Add ROOT output files to the MeanBurnupResults object
for i in range(nb_simus):
    m.AddSimulationAndProcess("evolution_" + str(i+1) + ".root")
    pass

# Print some mean values and their corresponding standard deviation
# at the last depletion step:
# M(U235) in depleted composition FUEL_R1,
last_step = m.GetSteps()
print "M(U235) in FUEL_R1 = %E(%E)"\
      %(m.GetMass(last_step,"FUEL_R1","U235"),m.GetMass(last_step,"FUEL_R1","U235",1))
# Local burnup in depleted compositions FUEL_R1 and FUEL_R6
for med in ["FUEL_R" + str(i+1) for i in range(6)]:
    print "LocalBurnup(%s) = %f(%f)"\
          %(med,m.GetLocalBurnup(last_step,med),m.GetLocalBurnup(last_step,med,1))


# Plot evolution of Kcoll vs. burnup
m.GetKcollGraph().Draw("ale")

# Write kcoll results in an ASCII file
f = open("kcoll.py.dat",'w')
for i in range(last_step):
  f.write("%i\t%f\t%f\n"%(m.GetBurnup(i+1),m.GetKcoll(i+1),m.GetKcoll(i+1,1)))
f.close()

# Necessary to prevent the canvas disappears
# (alternative: launch the script with python -i)
raw_input()
```

Listing 40.4: Example of Python post-processing script using the *MeanBurnupResults* class

These examples of post-processing scripts can be found in the *postscripts* directory.

#### 40.5.3.3 Producing Tripoli-4 composition files from ROOT outputs

As of TRIPOLI-4 v12, the ROOT output file contains all the necessary information to reconstruct all the TRIPOLI-4 compositions at any time step. The post-processing scripts in the *postscripts* directory make it possible to produce text files containing a list of material compositions in the TRIPOLI-4 format (see Chapter 10). For a single simulation result, one can use the *WriteT4CompositionFile()* method of the *BurnupResults* class:

```
import sys
sys.path.append("PREFIX")
# note: PREFIX must be replaced with the path to the 'postscripts' folder
# containing the post-processing scripts
from InitPostscripts import InitPostscripts
InitPostscripts()
from ROOT import BurnupResults

# Create a BurnupResults object from the 'evolution.root' output file
b = BurnupResults("evolution.root")

# Write a TRIPOLI-4 composition file called 'compos_1.t4' containing the
# compositions after the first depletion step
b.WriteT4CompositionFile('compos_1.t4', 1)

# Same thing for the compositions after the fifth depletion step
b.WriteT4CompositionFile('compos_5.t4', 5)
```

Listing 40.5: Using *BurnupResults* to produce a TRIPOLI-4 composition file from a ROOT output file

In the case of multiple independent simulations, one may also want to write the average material compositions. This can be done by calling the *WriteT4CompositionFile()* method on the *MeanBurnupResults* class:

```
import sys
sys.path.append("PREFIX")
# note: PREFIX must be replaced with the path to the 'postscripts' folder
# containing the post-processing scripts
from InitPostscripts import InitPostscripts
InitPostscripts()
from ROOT import MeanBurnupResults

# Create a MeanBurnupResults object
```

```
m = MeanBurnupResults()

# Add the 'evolution_1.root', 'evolution_2.root', ..., 'evolution_10.root'
# output files to the MeanBurnupResults object
nb_simus = 10
for i in range(nb_simus):
    m.AddSimulationAndProcess("evolution_" + str(i+1) + ".root")

# Write a TRIPOLI-4 composition file called 'compos_mean_1.t4' containing the
# average compositions after the first depletion step
m.WriteT4CompositionFile('compos_mean_1.t4', 1)

# Same thing for the compositions after the fifth depletion step
m.WriteT4CompositionFile('compos_mean_5.t4', 5)
```

Listing 40.6: Using *MeanBurnupResults* to produce a TRIPOLI-4 composition file from several independent ROOT output files

# Chapter 41

# ACTIVATION CALCULATION

The goal of activation calculations is typically to compute the dose delivered by photons due to the decay of matter that underwent neutron activation. TRIPOLI-4 implements the Rigorous 2-Step scheme (R2S), which consists of the following steps:

**Step 1:** neutrons are transported throughout the geometry. During this step, neutron fluxes and reaction rates are scored. The fluxes and reaction rates feed the source term of a system of Bateman equations for the concentrations of all the possible activation products. The Bateman equations are solved and the activities are determined at all the requested times. The neutron flux is allowed to vary in intensity over the activation time range, but it is assumed that the shape of the flux does not change (so that only one neutron transport calculation is necessary).

**Step 2:** For each activation time, the code computes the photon spectra emitted by the activation products. These spectra are used as the source of a photon-transport calculation.

Contrary to depletion calculations (see Chapter 40), the neutron flux, the reaction rates are not homogenized over the volumes of the TRIPOLI-4 geometry. The user is expected to define a mesh over the region of interest for activation and to specify the materials whose activation needs to be considered. Each mesh cell is then separately activated and represents a distinct radiation source for the second calculation step.

## 41.1 INPUT FOR AN ACTIVATION CALCULATION

An input deck for a depletion calculation is structured as follows :

- a TRIPOLI-4 static input file with some modifications explained below;
- an input file to specify the irradiation history, in the format specified in Section 40.1.4.

### 41.1.1 TRIPOLI-4 INPUT FILE

The format of the TRIPOLI-4 input file for the first calculation step is very similar to the format required for depletion calculations. Compared to a static TRIPOLI-4 input file, a few modifications are necessary.
First, the `RESPONSES` and `SCORE` blocks have to be replaced by `FILE` commands pointing to specific files:

```
FILE     directorypath/result/reponses.jdd
FILE     directorypath/result/scores.jdd
```

- *result* is the output directory. It is specified by the RESULT_DIRECTORY keyword in the DEPLETION block. By default it is called “result";
- The files *reponses.jdd* and *scores.jdd* contain respectively the RESPONSES and SCORE block. These files are automatically generated in the output directory from the list of depleted compositions, the depletion chain and reaction rates calculation options. No custom scores can be added to these files.

Second, the GRID_LIST block must contain at least three grids. The first one must be a one-group grid over the whole calculation energy range. The second one must be a two-group grid over the whole calculation energy range. Finally, the third grid must be the group structure associated with the multi-group cross sections provided in the GENDF files (specified by the GENDF keyword; see Chapter 40). The EAF-2010 GENDF file provided in the TRIPOLI-4 installation is associated with the 175-group VITAMIN-J structure. The GRID_LIST definition is provided in the TRIPOLI-4 installation, as well as in Figure 41.1.
Note that, contrary to depletion calculations, nothing special needs to be done about the composition files, because feedback from the compositions on neutron transport is assumed to be negligible.
Like in depletion calculations, the user needs to define a DEPLETION block. However, the syntax is slightly different, as shown below.

```
    DEPLETION
        MENDEL_LIBRARY  librarypath
        ACTIVATION
        EXTENDED_MESH
                Specification of the mesh for the activation calculation
        PARALLEL  <n_threads>
#               EDITION  <n_editions>
        DEPLETION_DIRECTORY  directorypath
        IRRADIATION_HISTORY   TIME  ihfilename
                Specification of the file describing irradiation history
        RENORMALIZATION
                 FIXED_SOURCE
        END_RENORMALIZATION
        DEPLETED_COMPOSITIONS
                 List of depleted compositions
#       HOMOGENIZE
#       SOBOL_POINTS  <n_points>
#       RANDOM_POINTS  <n_points>
        CHAIN
                Depletion chain / depleted compositions association
        POINTWISE_REACTION_RATES
                Selection of nuclides whose reaction rates are computed directly by TRIPOLI-4
        GENDF
                 GENDF cross-sections / depleted compositions association
        ESTIMATOR  <TYPE>
                Estimator to be used for computing flux and reaction rates (track length or collision
                estimators)
        RESULTS
                Files and directory results specifications
        END_RESULTS
    END_DEPLETION
```

Here:

- librarypath is the path to the Mendel library file (MendelLibrary_V0.1.BBL), which is installed along with TRIPOLI-4;
- n_cores is the number of CPU cores to use to parallelize the solution of the Bateman equations. Parallelization is handled by OpenMP;
- directorypath is the calculation directory; due to current limitations, this directory must be the current directory (. or the absolute path to the current directory);
- ihfilename is the name of the file containing irradiation history (see 40.1.4). The full path name is therefore *directorypath/ihfilename*;
- The unit for the irradiation history must be TIME, in days (contrast this with Section 40.1.4);
- The only allowed keyword for RENORMALIZATION is FIXED_SOURCE (see Section 40.1).

The extended mesh must be specified according to the format of Section 15.8. It may contain the optional NORMALIZE keyword, but none of the other ones (TRIANGLEZ, ENTROPY...). If NORMALIZE is given, then the score given in the standard TRIPOLI-4 output is the integral of the neutron flux over the activation mesh cells. Independently of the presence of the NORMALIZE keyword, the reaction rates for the activation step are correctly calculated using the neutron flux. Finally, the user must make sure that the mesh properly covers all the materials that need to be activated.
For the PARALLEL keyword, see the explanations in Section 41.2.3 below. For the syntax of the remaining options, see Chapter 40.

### 41.1.2 HOMOGENIZED CELLS

The list of activated materials is provided by the user with the keyword DEPLETED_COMPOSITIONS. The user can control the mapping between mesh cells and compositions with the HOMOGENIZE keyword. Two strategies are available:

- If the HOMOGENIZE keyword is specified, each mesh cell is assumed to be homogeneously filled with the material found at the centre[1] of the cell. The user must either ensure that cells are really homogeneous, or check that the error introduce by this assumption is negligible for their purpose.
- If the HOMOGENIZE keyword is not specified, TRIPOLI-4 treats the mesh cells as heterogeneous, but in order to do so it needs to draw a number of quasi-random or pseudo-random points in each cell to estimate the fraction of volume occupied by each composition. By default, TRIPOLI-4 uses $10^4$ points per cell. If further precision is required, the user may increase the number of points, at the expense of a larger initialization time, using the optional SOBOL_POINTS (for quasi-random sampling) or RANDOM_POINTS (for pseudo-random sampling) keywords.

The two homogenization options have different requirements in terms of mesh coarseness. If the mesh cells are homogenized, a fine mesh is required to accurately follow the concentration gradients of activation products over the geometry. Heterogeneous cells are less sensitive to the mesh fineness. On the other hand, the

[1] Note that the centre of spherical and cylindrical mesh cells is not defined as their centre of mass (which may lie outside the cell if the latter is not convex), but as the point corresponding to the midpoint of the range of spherical/cylindrical coordinates for the cell.

memory footprint for homogeneous cells is proportional to the number of mesh cells, while the footprint for heterogeneous cells is proportional to the number of mesh cells times the number of activated compositions, regardless of the number of compositions actually present in a single cell. Therefore, there is a trade-off associated with this option.

## 41.2 CALCULATION STEPS

As outlined above, the R2S calculation proceeds in two steps.

### 41.2.1 STEP 1: ACTIVATION

The first step includes the neutron-transport calculation and the solution of the Bateman equations for the concentrations of the activation products. The first calculation step produces several results, which are stored in the directory specified by the `RESULT_DIRECTORY` keyword:

- the ROOT output file aggregating all the simulation results (see Section 41.2.4 below)
- the files *reponses.jdd* and *scores.jdd* respectively contain the `RESPONSES` and `SCORE` blocks. They are automatically built from the depletion chain, the list of depleted compositions and the calculation options for the reaction rates.
- the output of the transport calculation (*t4output* file).

### 41.2.2 STEP 2: PHOTON TRANSPORT

The second calculation step samples photons according to the activities calculated in the first step and stored in the ROOT output file. These photons are used as the source of a photon transport calculation for which users are free to define their scores (fluxes, doses, etc.).
The TRIPOLI-4 input file for the second step is a normal input file, except for the definition of the source. The `SOURCE` block needs to be defined as follows:

```
SOURCES_LIST
    SOURCE
        PHOTON
        ACTIVATION
            <root_file>
            TIME_STEP <n_time_step>
            FILTER_BY_ISOTOPE
                Specification for the filtered isotopes
            ENERGY_THRESHOLD <e_threshold> // in MeV
            ACTIVITY_THRESHOLD <a_threshold> // in Bq/cm3
        END_ACTIVATION
    END_SOURCE
END_SOURCES_LIST
```

Here:

- `root_file` is the name of the ROOT file produced by the first calculation step;
- `n_time_step` is the index of the time step for the photon transport calculation. The photon source will be taken from the activities of the material compositions calculated at the given time step number. Note that the index is zero-based, but `n_time_step`=0 refers to the initial compositions, before neutron activation. Therefore, useful values of `n_time_step` range from 1 to the number of time steps in the irradiation file (Section 40.1.4);
- The (optional) `FILTER_BY_ISOTOPE` keyword can be used to limit the activation products that will be sampled. This may be useful, for instance, to ascertain the contribution of a particular isotope to the sought photon response. By default, all isotopes are accounted for (see below however for energy and activity cuts). Specifying `FILTER_BY_ISOTOPE`, followed by the number of isotopes, followed by the list of desired isotopes, will limit photon sampling to de-excitation photons of the isotopes included in the list.
- The (optional) keywords `ENERGY_THRESHOLD` and `ACTIVITY_THRESHOLD` may be used to cut de-excitation photons emitted below a given energy or activity threshold, respectively. By default, the energy threshold is 1 keV and the activity threshold is 0 Bq/cm$^3$, meaning that photons with energy below 1 keV will not be transported. Note: this means that their energy is neglected. It is not locally deposited.

If part of the source activity is cut by the `FILTER_BY_ISOTOPE`, `ENERGY_THRESHOLD` and `ACTIVITY_THRESHOLD` keywords, TRIPOLI-4 will automatically renormalize the source intensity to account for the missing activity. The output log will summarize the source normalization and the effect of the cuts with a message like the following:

```
integrated intensity = 5.630501e+11 Bq
(this value will be used as the source normalisation)
energy threshold = 1.000000e+00 keV
activity threshold = 0.000000e+00 Bq/cm3
neglected intensity (below energy and activity thresholds) = 2.916902e+09 Bq
ratio neglected/total = 5.153837e-01%
```

### 41.2.3 PARALLELIZATION OF THE ACTIVATION CALCULATION

Both steps of the R2S scheme may be run in parallel, using TRIPOLI-4's traditional parallelization techniques (Chapter 33).
Additionally, TRIPOLI-4 offers the possibility to parallelize the solution of the Bateman equations using OpenMP. The number of OpenMP threads can be controlled using the PARALLEL option. TRIPOLI-4 splits the activation calculation over the threads, assigning a certain number of activated cells to each OpenMP thread. When the activation calculations for a cell is completed, it is stored in a temporary memory buffer, waiting to be written to disk. Periodically (every 500 activated mesh cells by default), all the computed activation results are written to the ROOT output file; the buffer is cleared and a new batch of cell activation calculations starts.
Since I/O cannot be performed in parallel, the OpenMP threads must synchronize when the ROOT file is written. If the frequency of these rendez-vous points is too high, I/O will introduce some significant overhead and the efficiency of the parallel calculation will degrade. If the frequency is too low, the temporary buffer will need to be very large to accommodate all the calculated activation results, and the memory footprint might become unmanageable. The default interval (500 cells) is a good all-round compromise, but the user has the possibility to adjust the interval with the EDITION keyword.

### 41.2.4 CONTENT OF THE ROOT OUTPUT FILE

The ROOT file produced by the first calculation step is organized in two main TTree structures. The *evolflux* structure is similar to the one used in depletion calculations and described in Section 40.5.2. It is written only if the WRITE_FLUX keyword is specified in the DEPLETION block and it contains the flux computed on the DEC_FLUX_MULTIG energy grid for each depleted composition. It contains the following branches:

**step_num** : the activation time step index;

**compo_name** : the name of the activated composition;

**compo_vol** : the volume in $cm^3$ occupied by the current activated composition in the current mesh cell;

**renormfact** : the factor by which the reaction rates and fluxes are multiplied to obtain the normalization specified with the RENORMALIZATION command;

**cell_num** : the index of the cell;

**i, j, k** : the indices of the activated mesh cell along the three mesh axes;

**umin, vmin, wmin** : the lower bounds of the mesh cell coordinates, in the mesh coordinate system (cartesian, spherical or cylindrical);

**ulength, vlength, wlength** : the size of the mesh cell along the three directions of the mesh coordinate system (cartesian, spherical or cylindrical);

**group_id** : the index of the group;

**energy1** : the lower energy for the current group, in MeV;

**energy2** : the upped energy for the current group, in MeV;

**flux_val** : the value of the flux in the current group, in neutrons $\cdot\, \mathrm{cm}^{-2} \cdot \mathrm{s}^{-1}$;

This tree may require a great deal of disk space for large systems. In these cases, it is recommended to set `WRITE_FLUX` to 0.
The other relevant TTree structure is *gammaraystree*. This structure will always be present in activation calculations because it contains the necessary information for the second calculation step. It contains the following branches:

**step_num** : see *evolflux* above;

**irradtime** : cumulative amount of irradiation time, in seconds;

**coolingtime** : cumulative amount of cooling time, in seconds;

**compo_name** : see *evolflux* above;

**compo_vol** : see *evolflux* above;

**cell_num** : see *evolflux* above;

**i, j, k** : see *evolflux* above;

**umin, vmin, wmin** : see *evolflux* above;

**ulength, vlength, wlength** : see *evolflux* above;

**Egamma** : energy of the emitted photon line, in MeV;

**Igamma** : activity of the emitted photon line, in $\mathrm{Bq/cm}^3$;

**isotope_name** : name of the isotope emitting the current photon line;

**isotope_conc** : concentration of the isotope emitting the current photon line, in $10^{24}$ $\mathrm{cm}^{-3}$.

## 41.3 UNCERTAINTIES

Since R2S calculations are very similar to depletion calculations, one should apply the same caveats concerning the calculation of the statistical uncertainties on the calculation results. A single two-step calculation (neutron activation + photon transport) cannot possibly yield any estimate of the uncertainty. The only rigorous way to estimate the uncertainty is to observe a number of independent calculation replicas. One can then estimate the standard error from the dispersion of the sample. Assume $N$ independent R2S calculations (neutron activation + photon transport) were performed, and let $D_i$ be the photon dose (for example) resulting from the $i$th independent calculation. Then the average photon dose $\bar{D}$ and its standard error $s_{\bar{D}}$ are given by the traditional formulas

$$\bar{D} = \frac{1}{N}\sum_{i=1}^{N} D_i$$

$$s_{\bar{D}}^2 = \frac{1}{N(N-1)}\sum_{i=1}^{N} (D_i - \bar{D})^2$$

In practice, for the moment there are no pre- and post-processing tools that automate the tasks of creating the independent replicas, seeding them with different random numbers, running the two calculation steps, collecting the results and analyzing them. However, the launch scripts for depletion calculations can easily be repurposed for launching independent replicas for the first calculation step.

```
GRID_LIST       3
                DEC_1G          2
                                2.0000E+01      1.0000E-11
                DEC_2G          3
                                2.0000E+01      2.578840E-03    1.0000E-11
                DEC_FLUX_MULTIG                 176
1.0000000e-11   1.0000000e-07   4.1399000e-07   5.3158000e-07   6.8256000e-07   8.7642000e-07
1.1254000e-06   1.4450000e-06   1.8554000e-06   2.3824000e-06   3.0590000e-06   3.9279000e-06
5.0435000e-06   6.4760000e-06   8.3153000e-06   1.0677000e-05   1.3710000e-05   1.7603000e-05
2.2603000e-05   2.9023000e-05   3.7267000e-05   4.7851000e-05   6.1442000e-05   7.8893000e-05
1.0130000e-04   1.3007000e-04   1.6702000e-04   2.1445000e-04   2.7536000e-04   3.5358000e-04
4.5400000e-04   5.8295000e-04   7.4852000e-04   9.6112000e-04   1.2341000e-03   1.5846000e-03
2.0347000e-03   2.2487000e-03   2.4852000e-03   2.6126000e-03   2.7465000e-03   3.0354000e-03
3.3546000e-03   3.7074000e-03   4.3074000e-03   5.5308000e-03   7.1017000e-03   9.1188000e-03
1.0595000e-02   1.1709000e-02   1.5034000e-02   1.9305000e-02   2.1875000e-02   2.3579000e-02
2.4176000e-02   2.4788000e-02   2.6058000e-02   2.7000000e-02   2.8500000e-02   3.1828000e-02
3.4307000e-02   4.0868000e-02   4.6309000e-02   5.2475000e-02   5.6562000e-02   6.7379000e-02
7.2000000e-02   7.9500000e-02   8.2500000e-02   8.6517000e-02   9.8037000e-02   1.1109000e-01
1.1679000e-01   1.2277000e-01   1.2907000e-01   1.3569000e-01   1.4264000e-01   1.4996000e-01
1.5764000e-01   1.6573000e-01   1.7422000e-01   1.8316000e-01   1.9255000e-01   2.0242000e-01
2.1280000e-01   2.2371000e-01   2.3518000e-01   2.4724000e-01   2.7324000e-01   2.8725000e-01
2.9452000e-01   2.9720000e-01   2.9850000e-01   3.0197000e-01   3.3373000e-01   3.6883000e-01
3.8774000e-01   4.0762000e-01   4.5049000e-01   4.9787000e-01   5.2340000e-01   5.5023000e-01
5.7844000e-01   6.0810000e-01   6.3928000e-01   6.7206000e-01   7.0651000e-01   7.4274000e-01
7.8082000e-01   8.2085000e-01   8.6294000e-01   9.0718000e-01   9.6164000e-01   1.0026000e+00
1.1080000e+00   1.1648000e+00   1.2246000e+00   1.2873000e+00   1.3534000e+00   1.4227000e+00
1.4957000e+00   1.5724000e+00   1.6530000e+00   1.7377000e+00   1.8268000e+00   1.9205000e+00
2.0190000e+00   2.1225000e+00   2.2313000e+00   2.3069000e+00   2.3457000e+00   2.3653000e+00
2.3852000e+00   2.4660000e+00   2.5924000e+00   2.7253000e+00   2.8650000e+00   3.0119000e+00
3.1664000e+00   3.3287000e+00   3.6788000e+00   4.0657000e+00   4.4933000e+00   4.7237000e+00
4.9659000e+00   5.2205000e+00   5.4881000e+00   5.7695000e+00   6.0653000e+00   6.3763000e+00
6.5924000e+00   6.7032000e+00   7.0469000e+00   7.4082000e+00   7.7880000e+00   8.1873000e+00
8.6071000e+00   9.0484000e+00   9.5123000e+00   1.0000000e+01   1.0513000e+01   1.1052000e+01
1.1618000e+01   1.2214000e+01   1.2523000e+01   1.2840000e+01   1.3499000e+01   1.3840000e+01
1.4191000e+01   1.4550000e+01   1.4918000e+01   1.5683000e+01   1.6487000e+01   1.6905000e+01
1.7333000e+01   1.9640000e+01
END_GRID_LIST
```

Figure 41.1: An example of GRID_LIST block for R2S, with the 176 group boundaries of the VITAMIN-J structure (175 groups).

# Chapter 42

# TRIPOLI-4/GEANT4 COUPLING

GEANT4 is a particle-transport toolkit developed by several institutions around the world. The main application areas of GEANT4 are particle physics, space-radiation physics and medical physics. These domains often involve the transport of charged particles and high-energy particles; in this sense, GEANT4 and TRIPOLI-4 complement each other fairly well.
The coupling between TRIPOLI-4 and GEANT4 makes it possible for users to tap into GEANT4 physics library and use TRIPOLI-4 to solve transport problems which involve charged particles. Additionally, TRIPOLI-4 can use GEANT4 geometries just like another geometry format (similarly to what is done for ROOT geometries; see Chapter 36).
Note that GEANT4 is not a required prerequisite for normal TRIPOLI-4 execution. GEANT4 is only required for the execution of input decks using the TRIPOLI-4/GEANT4 coupling. In this case, you need to ensure that the GEANT4 environment is properly sourced before executing TRIPOLI-4.

## 42.1 REQUIREMENTS ON THE GEANT4 COMPILATION PHASE

The TRIPOLI-4/GEANT4 coupling requires GEANT4 to be compiled in single-threaded mode. In the GEANT4 version used by TRIPOLI-4 (v10.4), this requires setting the `GEANT4_BUILD_MULTITHREADED` CMake build flag to `OFF`.
If you has access to the TRIPOLI-4 sources, it is possible to recompile TRIPOLI-4 and couple it to a multi-threaded version of GEANT4. In this case, you must set the `GEANT4_BUILD_TLS_MODEL` CMake build option to `global-dynamic`. Failing to do so will result in mysterious run-time errors at the moment the GEANT4 shared libraries are loaded. For further details, please consult the GEANT4 documentation about the meaning of this option.

## 42.2 USING GEANT4 GEOMETRIES IN TRIPOLI-4

Since GEANT4 is not a particle-transport code, but rather a toolkit for particle transport, it does not offer a simple language to describe the problem geometry, like TRIPOLI-4 does. Instead, a GEANT4 geometry is defined by a set of one or more C++ source files. GEANT4 users need to compile these files to an executable format, together with a small main program and other simulation specifications.

Using GEANT4 geometries outside of GEANT4 is possible, but they still need to be compiled into a shared library linked to GEANT4. Also, the GEANT4 geometry description are implemented as a C++ class. Since the name of the geometry class and the arguments to the constructor are not known ahead of time, they cannot be coded in TRIPOLI-4.
The simplest way to make it possible to instantiate the GEANT4 geometry class from a shared library is to wrap the construction (and destruction) is a C function with a predefined name. The C wrapper must be compiled along with the wrapped GEANT4 geometry class.
Adding a wrapper to an existing GEANT4 can be done in two ways. The first, simpler way (Section 42.2.1) is to use a small open-source utility written for this purpose called `g42so`; the second way (Section 42.2.2) is to write the wrapper by hand.

### 42.2.1 WRAPPING A GEANT4 GEOMETRY WITH `G42SO`

One of the design principles of the TRIPOLI-4/GEANT4 coupling is that using an existing GEANT4 geometry in TRIPOLI-4 should require minimal user intervention. This task is simplified by `g42so`, a small tool written in Python 2. The `g42so` source code is shipped with the TRIPOLI-4® installation, but it can also be downloaded from Github (https://github.com/arekfu/g42so) or installed from PyPI (https://pypi.org/project/g42so/) via the Python package manager `pip`:

```
$ pip install g42so  # install from PyPI
$ pip install --user g42so  # install from PyPI into $HOME
$ pip install --user /path/to/g42so  # install from a local directory into $HOME
```

Please consult the Python documentation for instructions about using `pip`.
For the purpose of this example, assume that we are working with a GEANT4 geometry consisting of the following files:

- `include/DetectorConstruction.hh`;
- `src/DetectorConstruction.cc`;
- `include/CustomVolumeShape.hh`;
- `src/CustomVolumeShape.cc`.

In real life, these files would belong to a larger GEANT4 application, which would include a main executable, a primary-generator action (particle source) and other components. Also, we assume that the name of the geometry class is `DetectorConstruction` and, to make things a bit less trivial, that it uses a customized volume shape defined in `CustomVolumeShape.{cc,hh}`.
If `g42so` is installed, wrapping the geometry is easy:

```
$ g42so -I include --geant4-config /path/to/bin/geant4-config \
    src/DetectorConstruction.cc src/CustomVolumeShape.cc
```

The command above will produce a `libDetectorConstruction.so` file that can be used in the TRIPOLI-4 input file (see Section 42.2.3 below). The `-v` (or `--verbose`) option will show the compilation commands that `g42so` is executing.

```
/* This source files provides wrappers around the construction and the
 * destruction of your detector-construction class, which should derive from
 * G4VUserDetectorConstruction. You can expand the code as you like, for
 * instance by adding parameters to the class constructor or by calling your
 * detector-construction factories. Just make sure you expose the required
 * createG4Geometry and deleteG4Geometry functions.
 *
 * Once you have modified this file, save it and pass it to g42so as a source
 * file. Make sure you add the --custom-detector-wrapper CLI option, so that
 * g42so does not include its standard wrapper.
 */

#include "G4VUserDetectorConstruction.hh"
#include "G4GeometryManager.hh"

// This is where you should #include your custom headers
#include "MyDetectorConstruction.hh"

extern "C" {

  G4VUserDetectorConstruction *createG4Geometry() {
    // Replace "MyDetectorConstruction" with the name of your class
    G4VUserDetectorConstruction *aDetector = new MyDetectorConstruction();
    return aDetector;
  }

  void deleteG4Geometry(G4VUserDetectorConstruction *aGeom) {
    G4GeometryManager *manager = G4GeometryManager::GetInstance();
    if(manager->IsGeometryClosed())
      manager->OpenGeometry();
    delete aGeom;
  }

}
```

Listing 42.1: Template wrapper code for the instantiation of GEANT4 geometries generated by `g42so`.

### 42.2.2 BUILDING THE GEOMETRY WRAPPER BY HAND

Sometimes `g42so` cannot build the wrapper automatically. This happens, in particular, when the geometry class needs to be instantiated with some arguments. In such cases, `g42so` can still help you write your own wrapper.
First, running

```
$ g42so --dump-detector-wrapper >template_geom.cc
```

will write a template C++ wrapper to file `template_geom.cc`, shown in Listing 42.1. The template needs to be customized to match your class declaration. An example of customization is shown in Listing 42.2.
The customized wrapper can be subsequently compiled together with the geometry source files. This can be done using `g42so` or by invoking the compiler by hand. If you use `g42so`, you need to use the `--custom-detector-wrapper` option in order to tell `g42so` that it should not include its default wrapper.

```
#include "G4VUserDetectorConstruction.hh"
#include "G4GeometryManager.hh"

#include "DetectorConstruction.hh"  // note the custom header here

extern "C" {

  G4VUserDetectorConstruction *createG4Geometry() {
    G4VUserDetectorConstruction *aDetector = new DetectorConstruction("/path/to/data_file");
    return aDetector;
  }

  void deleteG4Geometry(G4VUserDetectorConstruction *aGeom) {
    G4GeometryManager *manager = G4GeometryManager::GetInstance();
    if(manager->IsGeometryClosed())
      manager->OpenGeometry();
    delete aGeom;
  }

}
```

Listing 42.2: An example of a customised geometry wrapper. Here we assume that the wrapped geometry class `DetectorConstruction` is declared in the `DetectorConstruction.hh` header file and that it must be instantiated with a string argument representing a path to a data file.

So, assuming that the customized wrapper was saved as `src/wrapper_geom_custom.cc`, the compilation command line with the custom wrapper should read

```
$ g42so -I include --geant4-config /path/to/bin/geant4-config --custom-detector-wrapper \
    src/DetectorConstruction.cc src/CustomVolumeShape.cc src/wrapper_geom_custom.cc
```

### 42.2.3 DECLARING THE GEANT4 GEOMETRY IN THE TRIPOLI-4 INPUT FILE

At this point you should have a compiled shared library object containing the GEANT4 geometry and the wrapper code. To use it in TRIPOLI-4, add a `G4_GEOMETRY` block, as follows:

**G4_GEOMETRY**
***NO_COMPOSITION***
**geometry_library.so**
**END_G4_GEOMETRY**

Here the `geometry_library.so` file is the shared library wrapping the GEANT4 geometry, as described above.

The NO_COMPOSITION keyword is optional. If it is not present, TRIPOLI-4 will attempt to build composition definitions based on the GEANT4 materials defined in the geometry. For each converted composition, TRIPOLI-4 will print a message translating the GEANT4 composition in a TRIPOLI-4 point-wise composition like the following:

```
Translated G4Material EAU as:
PUNCTUAL 300 EAU 2
H1 0.07676
O16 0.03338
```

You should check that these compositions are correct. In this case, no COMPOSITION or GEOMCOMP blocks are necessary.
The definition of the GEANT4 geometry may contain nuclides or material temperatures that are not available in the TRIPOLI-4 cross-section dictionary. In such cases, the NO_COMPOSITION option can be used to to turn off the automatic conversion. You must then provide a COMPOSITION block; the association between volumes and compositions is defined by the GEANT4 geometry, so no GEOMCOMP block is needed. Note that temperature interpolation (see Section 22.1) can also be used to work around mismatched temperatures in the GEANT4 compositions and in the TRIPOLI-4 cross-section dictionary.

## 42.3 VISUALIZING THE GEOMETRY WITH T4G

The TRIPOLI-4 visualizer T4G (Chapter 38) can be used to display GEANT4 geometries. For geometries that are not parametrized by UI commands, you just need to pass the name of the shared library to the T4G executable. Using our libDetectorConstruction.so example library from above, the command reads

```
$ static_visutripoli4 libDetectorConstruction.so
```

It is also possible to directly run T4G on the TRIPOLI-4 input file. This is the only way to display geometries parametrized by UI commands. The TRIPOLI-4 input file, however, must be given a .t4native extension. So, for example, if detector.t4 is a TRIPOLI-4 input file using a parametrized GEANT4 geometry, it can be viewed with T4G with the following commands:

```
$ ln -s detector.t4 detector.t4native
$ static_visutripoli4 detector.t4native
```

## 42.4 PHYSICS COUPLING

The TRIPOLI-4/GEANT4 coupling can be used to delegate the simulation of particles outside TRIPOLI-4's energy range to GEANT4. The principle is that any particle that is not treated by TRIPOLI-4 will be handed to GEANT4; this can happen for one of the following reasons:

1. TRIPOLI-4 cannot transport particles of the given type (for example: protons);
2. TRIPOLI-4 can transport the given particle type, but the type is not listed in the SIMULATION block under the PARTICLE keyword (see Section 22.2);
3. the particle energy is outside TRIPOLI-4's energy range, as determined by the ENERGY_INF and ENERGY_SUP parameters (see Section 22.2).

Note that GEANT4 can only transport particles that are actually produced by some process. Since TRIPOLI-4 will usually not produce particles outside its own range of applicability, the typical coupling scenario involves high-energy particles being transported by GEANT4 at the beginning of their life; neutrons, photons and other particles (depending on the options in the SIMULATION block) may then migrate to TRIPOLI-4 when they enter the low-energy range. The task of injecting high-energy particles into the simulation can be handled by wrapped GEANT4 sources (see Section 42.5 below).
Note also that the physics coupling between TRIPOLI-4 and GEANT4 is only possible in a GEANT4 geometry.

### 42.4.1 SYNTAX

The physics coupling with GEANT4 requires the presence of a G4_INTERFACE block in the input deck. The syntax is the following:

**G4_INTERFACE**
**PHYSICS**
***PHYSICS_LIST physics_list_name***
**END_G4_INTERFACE**

The PHYSICS keyword requests initialization of the GEANT4 physics. The PHYSICS_LIST keyword may be used to specify which physics list should be used by GEANT4 [41]. The physics list defaults to FTFP_INCLXX. Note that the G4_INTERFACE block must appear at the beginning of the input file, before G4_UI_COMMANDS and G4_GEOMETRY.

## 42.5 USING GEANT4 PARTICLE SOURCES IN TRIPOLI-4

High-energy source particles can be produced using GEANT4 sources. In the GEANT4 parlance, sources are called primary generator actions. Like for geometries, GEANT4 sources are described by implementing a special C++ class. The instantiation and the destruction of the primary-generator-action class need to be wrapped in a simple C function. The g42so utility can automatically generate the wrapper if the primary-generator-action class can be instantiated without any arguments; otherwise, the wrapper needs to be written by hand (although, again, g42so can help in that respect, too).
For the sake of illustration, we assume that the GEANT4 particle source consists of the following source files:

- include/PrimaryGeneratorAction.hh;
- src/PrimaryGeneratorAction.cc;
- include/PrimaryGeneratorActionMessenger.hh;
- src/PrimaryGeneratorActionMessenger.cc.

Here we assume that the primary generator action proper is defined in `PrimaryGeneratorAction.cc`, and that `PrimaryGeneratorActionMessenger.cc` defines a messenger class for the primary generator action (see Section 42.8 below).

### 42.5.1 WRAPPING A GEANT4 SOURCE WITH `G42SO`

Invoking `g42so` on the source (`*.cc`) files

```
$ g42so -I include --geant4-config /path/to/bin/geant4-config \
    src/PrimaryGeneratorAction.cc src/PrimaryGeneratorActionMessenger.cc
```

will produce a shared library called `libPrimaryGeneratorAction.so` containing the particle source and the wrapper.

### 42.5.2 BUILDING THE SOURCE WRAPPER BY HAND

If the constructor of the primary-generator-action class cannot be invoked without any arguments, it is necessary to build the wrapper by hand. The process is very similar to the processes for building the geometry wrapper (Section 42.2.2). First, one creates a wrapper template for primary generator actions with the command

```
$ g42so --dump-pga-wrapper >template_pga.cc
```

The wrapper template is shown in Listing 42.3. The template needs to be customized to match your class declaration. An example of customization is shown in Listing 42.4.
The customized wrapper can be subsequently compiled together with the primary-generator-action source files. This can be done using `g42so` or by invoking the compiler by hand. If you use `g42so`, you need to use the `--custom-pga-wrapper` option in order to tell `g42so` that it should not include its default wrapper. So, assuming that the customized wrapper was saved as `src/wrapper_pga_custom.cc`, the compilation command line with the custom wrapper should read

```
$ g42so -I include --geant4-config /path/to/bin/geant4-config --custom-pga-wrapper \
    src/PrimaryGeneratorAction.cc src/PrimaryGeneratorActionMessenger.cc \
    src/wrapper_pga_custom.cc
```

```
/* This source files provides wrappers around the construction and the
 * destruction of your primary-generator-action class, which should derive from
 * G4VUserPrimaryGeneratorAction. You can expand the code as you like, for
 * instance by adding parameters to the class constructor or by calling your
 * particle factories. Just make sure you expose the required createG4PGA and
 * deleteG4PGA functions.
 *
 * Once you have modified this file, save it and pass it to g42so as a source
 * file. Make sure you add the --custom-pga-wrapper CLI option, so that g42so
 * does not include its standard wrapper.
 */

#include "G4VUserPrimaryGeneratorAction.hh"

// This is where you should #include your custom headers
#include "MyPrimaryGeneratorAction.hh"

extern "C" {

  G4VUserPrimaryGeneratorAction *createG4PGA() {
    // Replace "MyPrimaryGeneratorAction" with the name of your class
    G4VUserPrimaryGeneratorAction *aPGA = new MyPrimaryGeneratorAction();
    return aPGA;
  }

  void deleteG4PGA(G4VUserPrimaryGeneratorAction *aPGA) {
    delete aPGA;
  }

}
```

Listing 42.3: Template wrapper code for the instantiation of GEANT4 primary generator actions (sources) generated by `g42so`.

```
#include "G4VUserPrimaryGeneratorAction.hh"
#include "PrimaryGeneratorAction.hh"

extern "C" {

  G4VUserPrimaryGeneratorAction *createG4PGA() {
    G4VUserPrimaryGeneratorAction *aPGA = new PrimaryGeneratorAction("/path/to/data_file");
    return aPGA;
  }

  void deleteG4PGA(G4VUserPrimaryGeneratorAction *aPGA) {
    delete aPGA;
  }

}
```

Listing 42.4: An example of a customised primary-generator-action wrapper. Here we assume that the wrapped primary-generator-action class `PrimaryGeneratorAction` is declared in the `PrimaryGeneratorAction.hh` header file and that it must be instantiated with a string argument representing a path to a data file.

### 42.5.3 DECLARING THE GEANT4 SOURCE IN THE TRIPOLI-4 INPUT FILE

At this point you should have a compiled shared library object containing the GEANT4 primary generator action and the wrapper code. To use it in TRIPOLI-4, add a `G4_SOURCE` to the `SOURCES_LIST` block, as follows:

```
SOURCE
    ALL_PARTICLE
    G4_SOURCE  pga_library.so
END_SOURCE
```

Here the `pga_library.so` file is the shared library wrapping the GEANT4 primary generator action, as described above.

## 42.6 WRAPPING SOURCES AND GEOMETRIES IN THE SAME SHARED LIBRARY

It is possible to use `g42so` to wrap a GEANT4 geometry and a primary generator action in the same shared library. To this end, simply run `g42so` on all the relevant source files at the same time, possibly including any hand-coded wrappers (in this case, remember to specify the relevant `--custom-*-wrapper` option).
Using the examples of Sections 42.2.2 and 42.5 above, a complete command line for `g42so` might read

```
$ g42so -I include --geant4-config /path/to/bin/geant4-config \
    src/DetectorConstruction.cc src/CustomVolumeShape.cc \
    src/PrimaryGeneratorAction.cc src/PrimaryGeneratorActionMessenger.cc \
    -o libG4sim.so
```

The generated `libG4sim.so` file can be used both in the `G4_GEOMETRY` and in the `G4_SOURCE` blocks.

## 42.7 SCORES

Particles treated by GEANT4 and recognized by TRIPOLI-4 (neutrons, photons, electrons and positrons) are normally taken into account for the purpose of the scores defined in the TRIPOLI-4 input deck. This is true even outside TRIPOLI-4's usual energy range.
However, since there is no special syntax for defining scores on particles in the GEANT4 domain, it is currently not possible to define scores for particles that TRIPOLI-4 does not recognize.

## 42.8 SUPPORT FOR GEANT4 UI COMMANDS

Some GEANT4 applications are parametrized using GEANT4's UI commands. UI commands typically have the form "`/myApp/setDetectorX 3 cm`"; they can be used to change some simulation parameters (including

the geometry) at run time, without recompiling the application. Executing a GEANT4 UI commands is only possible if the GEANT4 kernel is the correct state; for instance, commands that modify the geometry can typically be executed only in `PREINIT` state.
The TRIPOLI-4/GEANT4 coupling supports the execution of arbitrary UI commands in `PREINIT` state (before the call to the `/run/initialize` UI command) and `IDLE` state (after `/run/initialize` but before `/run/beamOn`). The commands must be specified in a `G4_UI_COMMANDS` block, as follows:

**G4_UI_COMMANDS**
***PREINIT***
***/some/ui/command params***
***/some/ui/command params***
...
***FIN_PREINIT***
***IDLE***
***/some/ui/command params***
***/some/ui/command params***
...
***FIN_IDLE***
**END_G4_UI_COMMANDS**

Any number of commands may be included in each kernel state.
Some important remarks:

- the `/run/initialize` and `/run/beamOn` commands must not be included;
- the `G4_UI_COMMANDS` block must appear before `G4_GEOMETRY` if the geometry is parametrized by UI commands.

## 42.9 TROUBLESHOOTING

When TRIPOLI-4 loads the shared libraries containing the user's GEANT4 geometry or source, errors like the following may appear:

```
/.../bin/dynamic_tripoli4-root-enabled: symbol lookup error:
./libPrimaryGeneratorAction.so: undefined symbol:
_ZN31PrimaryGeneratorActionMessengerC1EP22PrimaryGeneratorAction
```

The long string at the end is a mangled C++ identifier. It is the internal representation the name of a class, a method or a function that could not be found when TRIPOLI-4 loaded the `libDetectorConstruction.so`

shared library. Tools called demanglers[1] can transform the internal representation into a human-readable string. The symbol above demangles to

```
PrimaryGeneratorActionMessenger::PrimaryGeneratorActionMessenger(PrimaryGeneratorAction*)
```

Typically, errors about missing symbols are caused by the omission of some necessary source files at the in the arguments to `g42so`. In this particular case, the demangled string indicates that we probably forgot to include the `PrimaryGeneratorActionMessenger.cc` source file when compiling the shared library. Indeed, the error message above was generated by omitting this source file.
Ideally, this kind of omissions should be detected when the shared library is compiled. Future versions of `g42so` might support this feature.

[1] See `c++filt` or `https://demangler.com`, for instance.

# Chapter 43

# TRIPOLI-4/MORET COUPLING

The MORET code is developed by the IRSN and simulates the transport of neutrons in three dimensions by the means of the Monte Carlo method. TRIPOLI-4 and its visualisation tool T4G can use the MORET code to process geometrical calls and boundary conditions. The integration of MORET into TRIPOLI-4 was designed so that minimal modifications are required in data files. The geometry part of the MORET code is linked on the fly in the TRIPOLI-4 code in order to process all geometry calls in place of the TRIPOLI-4 geometrical functions. The geometry functions of MORET must thus be made available to TRIPOLI-4 by means of a dynamic library. To inform TRIPOLI-4 of the location of the MORET library on the system, the user must fill the `LD_LIBRARY_PATH` appropriately.

We will give the keywords used to declare a block to be processed by the MORET code. However, users will need to refer to the MORET documentation for the syntax inside the blocks.

## 43.1 USING A MORET GEOMETRY

To invoke MORET for the geometry, the keyword `MORET_GEOMETRY` must be placed before the geometry declaration block inside the data file. Upon reading that keyword, TRIPOLI or T4G will call the MORET geometry reader. The keyword `ENDG` must be used to end the geometry.

The syntax block is then

```
MORET_GEOMETRY
        file(name)
ENDG
```

## 43.2 USING MORET BOUNDARY CONDITIONS

As with a native TRIPOLI-4 geometry, the MORET geometry allows for boundary conditions to be applied. The command used for this is as follows:

```
MORET_LIMIT
        FACE F_<iface1>:V_<ivolu1>:M_<imodule1> <condition_type1>
        FACE F_<iface2>:V_<ivolu2>:M_<imodule2> <condition_type2>
        ...
ENDB
```

Only reflection boundary conditions and the default leakage boundary conditions are available in the version 12 of TRIPOLI-4. The condition_type is 1 for the reflection.

# Chapter 44

# SCHEDULER

This optional block opens a new way to use TRIPOLI-4. Up to version 10, it was only possible to invoke the code in the standard way, only one initialisation and then one simulation.
It allows to process parts (initialisation, simulation) of calculations one or more times. It allows to have multiple SIMULATION blocks (with names) that can be invoked as tasks when needed.

The general syntax for a scheduler block is as follows :

```
SCHEDULER
        INITIALIZE                          SIMUi
        SIMULATION                          SIMUi
        LOAD_SOURCES_FROM_KIJ_RESULTS
        CLEAN_ALL_RESULTS
END_SCHEDULER
```

Internally, the calculation has been split into three parts:

1. the reading of the input deck;
2. initialisation of internal structures of a simulation;
3. simulation (transporting particles).

The reading of the input deck is always invoked when the code is launched. If the scheduler block is absent, the calculation runs as usual going on with the initialisation and simulation. If the scheduler block is present, then the calculation will be defined by the task included into this block.

**WARNING:** this feature opens a whole category of calculation schemes which are not currently possible which TRIPOLI-4. It is recommended to stay within the calculations schemes described in this document.

This capability is being made available in mono-processor mode and in parallel mode.

## 44.1 KIJ ACCELERATION OF CRITICAL CALCULATIONS

The scheduler was initially created to implement KIJ acceleration inside a criticality simulation. This acceleration is indeed a multi-step calculation:

- first, a standard very short simulation is done with KIJ matrix calculation;
- then, the KIJ matrix is diagonalised and the source of the next calculation is initialised with the fundamental solution;
- then the real simulation is started with new scores.

**WARNING:** this feature is theoretically longer in mono-processor mode than when using no acceleration. It is thus recommended not to use this feature in mono-processor mode. Only the parallelism can accelerate source convergence when using KIJ two steps calculation.
In the context of this development, some junctions between tasks have been developed:

- the capability to clear all scores between tasks;
- the capability to initialise sources from previous results of a KIJ calculation.

The user prepares two critical simulations that he will name `SIMU1` and `SIMU2`.
The first simulation should contain very few batches. The keyword `BATCH_PER_SIMULATOR` can be easy to use in this context. 3 to 5 batches per simulator are good number to start with when using for example 50 processors in parallel. This simulation should include a `KIJ_MATRIX` keyword. The keyword must be completed by a full KIJ block specifying the mesh to be used for scoring fissions. The first simulation must be initialised and run.
Then the KIJ matrix must be diagonalised and the fundamental vector must be transferred in the sources. The keyword `LOAD_SOURCES_FROM_KIJ_RESULTS` must be invoked here. At this step, it is important to note that the sources must have the exact same mesh as the KIJ matrix. New keywords (syntaxic sugars) have been introduced in the sources definition to help the user get the same mesh (`UPTO_LIN` and `REPEAT`).
The second simulation `SIMU` is a normal simulation. It was important here to be able to deactivate the KIJ matrix calculation in the second simulation as the acceleration gain of the method may easily be lost with a huge score like KIJ matrices. The user disables the KIJ scoring in the second simulation by not including the keyword `KIJ_MATRIX` in the second simulation.
An example of a scheduler block is as follows:

```
SCHEDULER
  INITIALIZE SIMU1
  SIMULATION SIMU1
  LOAD_SOURCES_FROM_KIJ_RESULTS
  CLEAN_ALL_RESULTS
  SIMULATION SIMU2
END_SCHEDULER
```

## 44.2 SPECIALISED TASKS

Some tasks have been created to make junctions between initialisation and simulations. Other tasks can be implemented to create more interesting sequences.

### 44.2.1 CLEANING OF RESULTS

This task cleans all collected results by a previous simulation. The statistics are reset to scores of 0. and 0 batches. The scores currently remain initialized and can be accumulated again in a second simulation without initialisation.

### 44.2.2 INITIALISING SOURCES FROM KIJ MATRIX FUNDAMENTAL

This task should be invoked when a previous simulation has been done and the sources of the next simulation are to be initialised with the result of the fundamental calculated from the KIJ matrix.

**Important:** The KIJ matrix must have been calculated using a `EXTENDED_MESH` (fissile volumes will not do). The sources will be declared using cartesian tabulated mesh (`CARTESIAN`, `TABULATED` keywords) in which all matrix elements are provided (`F_UVW` format) and will have exactly the same structure than the KIJ matrix (same position, same number of cells in every dimensions).

# Chapter 45

# GENERATOR OF STOCHASTIC GEOMETRIES (CASTOR)

CASTOR (Construction and Analysis of STOchastic Realizations) is a code written in C++ and developed at CEA. The executable can be found at the following path: `CODE/<OpSys>/tools/Castor/bin/castor`, where `<OpSys>` denotes the operating system.

## 45.1 INTRODUCTION

CASTOR is devoted to the Monte Carlo sampling of three types of stochastic geometries [49]:

1. three-dimensional isotropic and homogeneous Poisson tessellations ($n$-ary mixtures);
2. three-dimensional Poisson-Box tessellations ($n$-ary mixtures);
3. three-dimensional spherical inclusions (non-overlapping spheres with low packing-fraction) in a background matrix (binary mixtures).

The spheres of the spherical inclusions can be either mono-dispersed (constant radius) or poly-dispersed (radius sampled according to a probability law).

Currently, the sampled random geometry can be exported to three differents types of outputs:

1. ASCII files corresponding to different blocks to be inserted into a datafile for the Monte Carlo transport code TRIPOLI-4. The code can generate between two and three blocks and one connectivity map (according to the options chosen by the user):
   - `GEOMETRY` block: describes the whole geometry with a combinatorial or surface-based description (see also Chapter 7);
   - `GEOMCOMP` block: associates the compositions (identified by a label/color) and volumes (see also Chapter 11);
   - `BOUNDARY_CONDITION` block (optional): describes the boundary conditions applied to the geometry (see also Chapter 9);

- a connectivity map (optional): contains the list of neighbors for each volume and is included in the GEOMETRY block (see also Subsection 7.14.1).

A full dataset for TRIPOLI-4 can be then completed by the user via the keyword FILE (see Subsection 6.2).

2. An ASCII file corresponding to an input file for the TRIPOLI-4 visualizer T4G (see Chapter 38), in order to visualize the geometry directly.
3. A file with VtK format to visualize the geometry in three dimensions with a software such as ParaView. The export to VtK of one given geometry is based on the tetrahedral decomposition of polyhedra composing the tessellation or on the discretization of spheres.

For very large geometries, we recommend to add the following options to the command line calling the TRIPOLI-4 executable (in order to speed-up the initialization of the geometry):

```
-a -T notest -T decresponse -T decresponse
```

As a verification of generated geometries, the code can also compute ensemble-averaged stastical properties over the geometries, namely the stereological properties (chord length moments and distributions). The code computes the average chord lengths:

- across the volumes composing the geometry (average uncolored chord length $CL$)
- across compositions of label $i$ (average colored chord length $CL_i$)

for a given number of test lines sampled with $\mu$-randomness property and exports corresponding histograms on pre-defined grids.

## 45.2 CONSTRUCTION OF RANDOM GEOMETRIES

### 45.2.1 TYPE OF GEOMETRIES (MANDATORY)

The type of stochastic geometries is the single mandatory argument. It should be provided by this command:

```
./castor --poisson3d
```

for Poisson tessellations,

```
./castor --poissonbox3d
```

for Poisson-Box tessellations,

```
./castor --inclusions3d
```

for spherical inclusions.

### 45.2.2 NUMBER OF REALIZATIONS

The number of geometries to generate is given by the option `-N` or `--number`. This number is set to 5 by default.

Example:

```
./castor --poisson3d -N 1000
```

to generate 1000 realizations.

### 45.2.3 CONTAINER

The shape of the container containing the geometry is given by the option `-C` or `--container`. The container is always centered at the origin $(0,0,0)$. Several types of containers can be used:

- a **parallepipedic box** by using:

  ```
  ./castor --poisson3d -C box,10.,25.,6.
  ```

  for a box of side $L_x = 10$, $L_y = 25$ and $L_z = 6$;

- a **cube** by using:

  ```
  ./castor --poisson3d -C box,20.
  ```

  for a cube of side $L = 20$;

- a **sphere** by using:

  ```
  ./castor --poisson3d -C sphere,20.
  ```

  for a sphere of radius $R = 20$;

- a **cylinder parallel to $x$-axis** by using:

  ```
  ./castor --poisson3d -C xcylinder,5.,20.
  ```

  for a cylinder of radius $R = 5$ and of length $L_x = 20$;

- a **cylinder parallel to $y$-axis** by using:

  ```
  ./castor --poisson3d -C ycylinder,5.,20.
  ```

  for a cylinder of radius $R = 5$ and of length $L_y = 20$;

- a **cylinder parallel to $z$-axis** by using:

  ```
  ./castor --poisson3d -C zcylinder,5.,20.
  ```

  for a cylinder of radius $R = 5$ and of length $L_z = 20$;

By default, the container is a squared box of side 10.

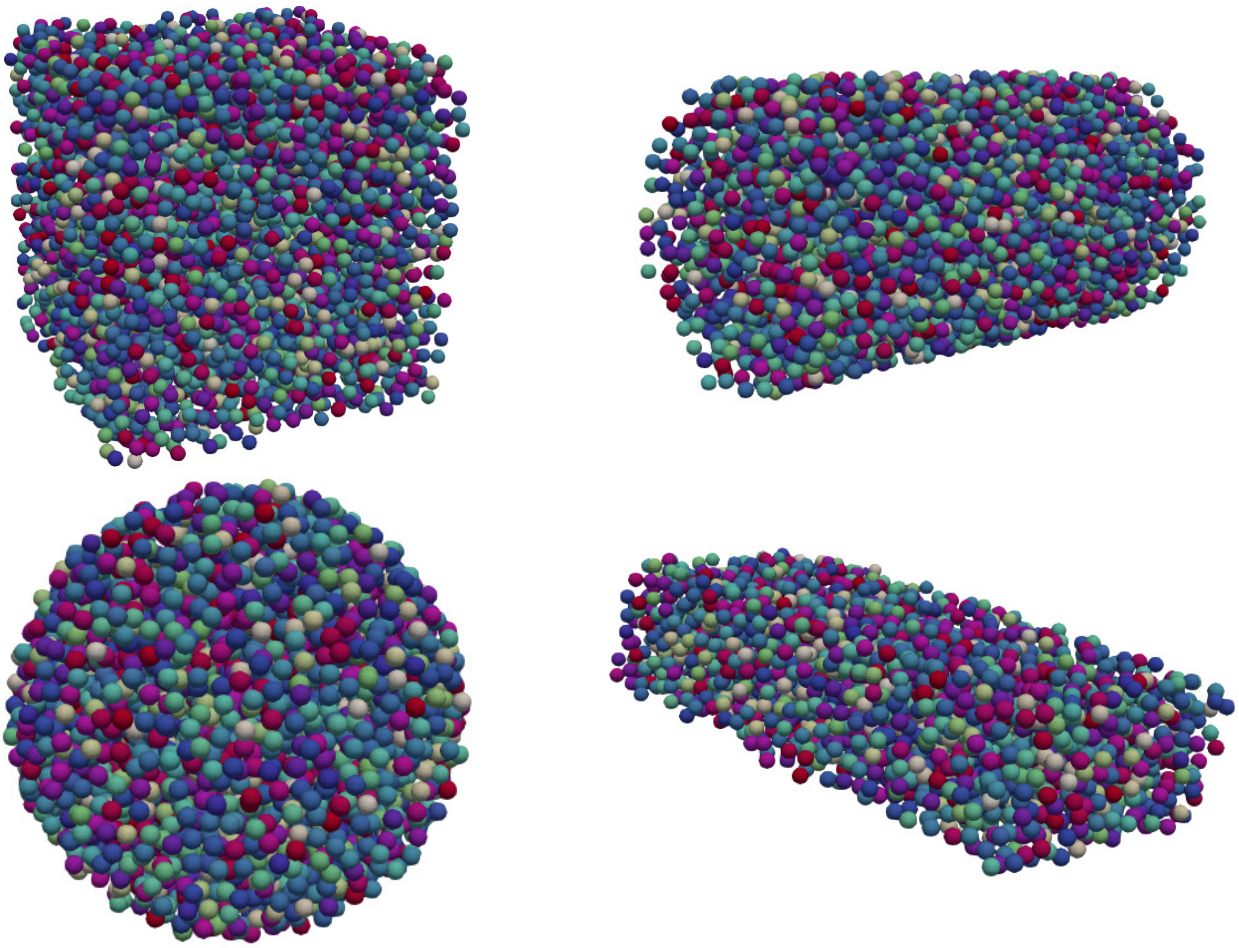


Figure 45.1: Different containers for spherical inclusions: cube, parallelepipedic box, $x$-cylinder, sphere.

### 45.2.4 COLORING PROBABILITIES (VOLUME RATIOS)

For Poisson and Poisson-Box tessellations, a step of coloring is performed after the decomposition of the domain with random hyperplanes: each volume of the tessellation is assigned a label $i$ with a probability $p_i$ ($1 \leq i \leq n$), to obtain a $n$-ary mixture ($n$ being a arbitray integer). In an infinite domain, this probability $p_i$ would correspond to the volume ratio for composition $i$.

For spherical inclusions, no random coloring process is used, since we consider the binary mixture matrix/-spheres: in this case, the geometry is decomposed into a background matrix (filled with composition 0) and a set of non-overlapping spheres (filled with composition 1). The target volume ratio (corresponding to $p_1$) for spheres is also denoted by the packing fraction.

In both cases, the user can provide the target volume ratios for each composition by using the option `-P` or `--probabilities`.

Examples:

- tessellation (binary mixture):

  ```
  ./castor --poissonbox3d -P 0.7,0.3
  ```

  where the label 0 is assigned with probability $p_0 = 7$ and the label 1 assigned with probability $p_1 = 0.3$;

- tessellation (ternary mixture):

  ```
  ./castor --poisson3d -P 0.4,0.5,0.1
  ```

  where the label 0 is assigned with probability $p_0 = 0.4$, the label 1 assigned with probability $p_1 = 0.5$ and the label 2 assigned with probability $p_2 = 0.1$;

- spherical inclusions (by construction, binary mixture):

  ```
  ./castor --inclusions3d -P 0.9,0.1
  ```

  for a target packing fraction equal to 0.1.

By default, the geometry is a binary mixture with $p_0 = 0.8$ and $p_1 = 0.2$.

Please note that the maximal packing fraction in CASTOR is 0.37 because of the underlying algorithm (Random Sequential Addition).

### 45.2.5 GEOMETRICAL SPECIFICATIONS (TESSELLATION DENSITY OR RADIUS DISTRIBUTION)

#### 45.2.5.1 Poisson and Poisson-Box geometries

For Poisson and Poisson-Box tessellations, the geometries are generated for a given (positive) free parameter corresponding to the tessellation density, denoted ρ (see illustration in Figure 45.2). In infinite geometries, the average chord length is equal to:

- $\frac{1}{\rho}$ in Poisson tessellations;
- $\frac{2}{3\rho}$ in Poisson-Box tessellations.

The user can provide this parameter using `-R` or `--rho`.

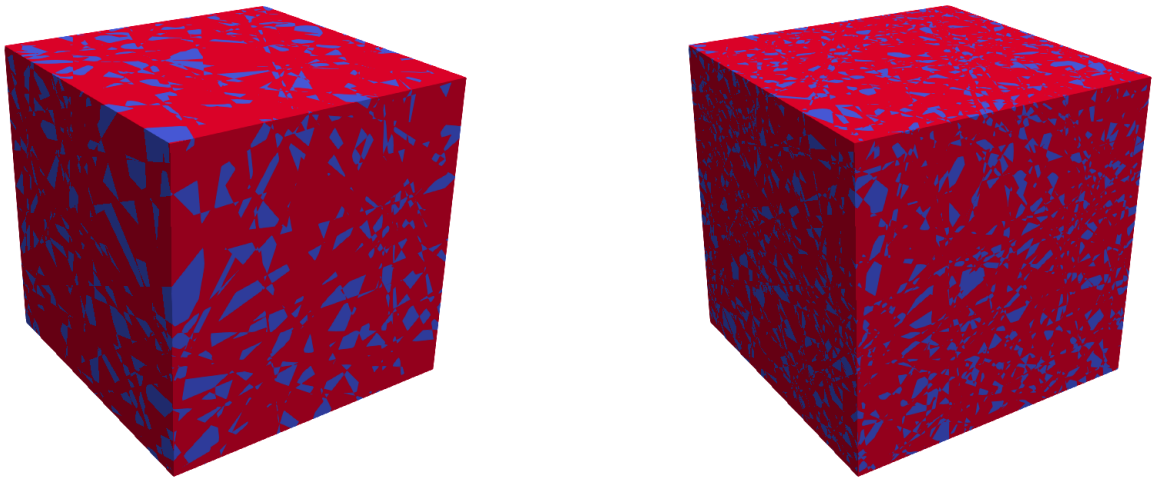


Figure 45.2: Example of realization of a Poisson tessellation in a cube of side $L = 10$, with coloring probabilities $p_0 = 0.8$ (red) and $p_1 = 0.2$ (blue), with a tessellation density $\rho = 5.0$ (left) and $\rho = 10.0$ (right).

Examples:

- The tessellation density ρ is set to 5 (theoretical chord length equal to 0.2).

```
./castor --poisson3d -R 5.
```

- The tessellation density ρ is set to 2 (theoretical chord length equal to 0.333333).

```
./castor --poissonbox3d -R 2.
```

The default density is equal to 2.

#### 45.2.5.2 Spherical inclusions

For spherical inclusions, the geometries are generated for a given radius distribution. The user can provide this parameter using `-D` or `--distribution`. So far, three types of distributions are available:

- radius following a **constant** distribution by using:

  ```
  ./castor --inclusions3d -D constant,0.1
  ```

  for a radius $r = 0.1$;

- radius following a **uniform** distribution by using:

  ```
  ./castor --inclusions3d -D uniform,0.05,0.3
  ```

  for a uniform distribution between $r_{min} = 0.05$ and $r_{max} = 0.3$;

- radius following an **exponential** distribution by using:

  ```
  ./castor --inclusions3d -D exponential,0.2
  ```

  for exponential distribution with mean $r = 0.2$.

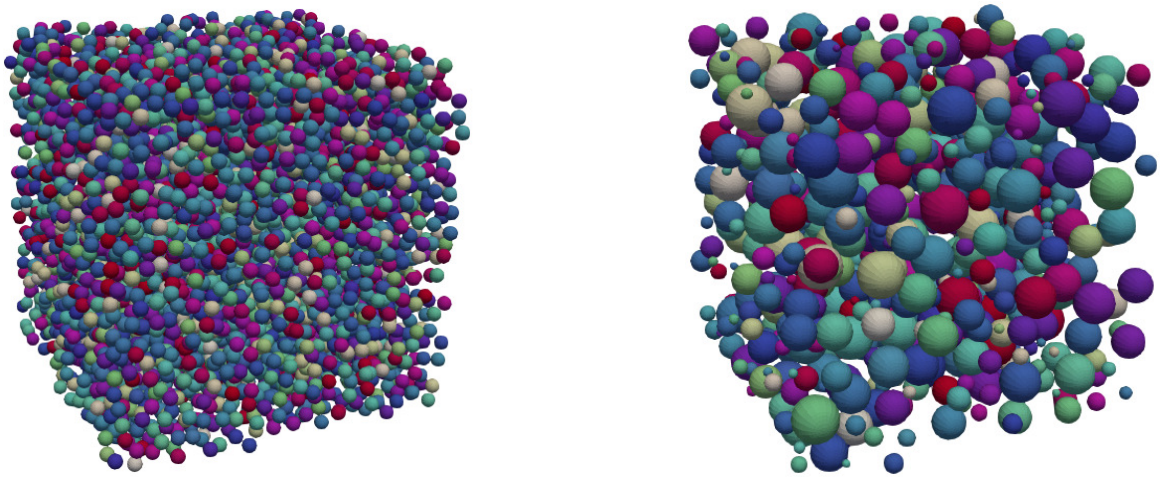


Figure 45.3: Example of realization of spherical inclusions in a cube of side $L = 10$, with a packing fraction $\xi = 0.2$, with a constant distribution $r = 0.2$ (left) and a uniform distribution between $r_{min} = 0.1$ and $r_{max} = 0.6$ (right).

The default distribution is a constant distribution with radius $r = 0.5$.

### 45.2.6 CONSTRUCTION ALGORITHM

For Poisson tessellations, several algorithms can be used to build the geometry. The naive algorithm being time-consuming, other algorithms have been proposed to speed-up the construction. The following options (that exclude each other) can be chosen:

- naive algorithm (algorithm A) by using:

```
./castor --poisson3d --naive
```

- algorithm B (ascending hierrachy), by using:

```
./castor --poisson3d --opt
```

- Edelbrunner's algorithm (algorithm C) by using:

```
./castor --poisson3d --edelsbrunner
```

- mesh-based algorithm D by using:

```
./castor --poisson3d --meshing -M 10,20,30
```

  for a mesh $10 \times 20 \times 30$.

- hybrid algorithm E (hydrid algorithm between algorithms B and D) by using:

```
./castor --poisson3d --hybrid -M 10,20,30
```

  for a mesh $10 \times 20 \times 30$.

By default, the algorithm B is used.

Geometries produced by these different algorithms are assumed to be identical for uncolored geometrical properties. Nevertheless, please note that the indexation of volumes can be different since volumes can be created in a different order as a function of the algorithm. Thus, coloring properties can differ from one algorithm to another.

The construction of spherical inclusions (based on a Random Sequential Addition algorithm) can be accelerated by a mesh-based method (bounding box). Therefore, the user can provide a mesh as follows:

```
./castor --inclusions3d -M 10,15,20
```

for a mesh $10 \times 15 \times 20$. By default, the mesh-based algorithm is not applied.

There is no option concerning the algorithm for Poisson-Box tessellations.

## 45.3 EXPORT INPUT FILES FOR TRIPOLI-4 AND T4G

### 45.3.1 ACTIVATION AND TARGET REPOSITORY

The option `-T` or `--tripoli4` activates the generation of inputs files for TRIPOLI-4.
The option `-t` or `--tpath` specifies a repository where outputs will be exported.

Example:

```
./castor --inclusions3d -T -t repo_t4
```

By default, output files will be exported into the current directory.

For the block GEOMETRY, each output file is called geometry_i (where *i* corresponds to the index of the realization). For the block GEOMCOMP, each output file is called geomcomposition_i. For the block BOUNDARY_CONDITION, each output file is called limit_i. The output file for the connectivity map is called map_i.

### 45.3.2 START RANK AND END RANK

The start rank and end rank for the production of TRIPOLI-4 output files can be chosen by the user with options --tstartrank and --tendrank.

Example: TRIPOLI-4 input files are produced between the 21st and the 30th realization:

```
./castor --inclusions3d -N 100 -T -t repo_t4 --tstartrank 21 --tendrank 30
```

By default, all generated geometries are exported to TRIPOLI-4 files if -T is activated, i.e. the start rank is equal to 1 and the end rank is equal to *N*, where *N* is the number of generated geometries.

### 45.3.3 NAME OF MIXTURE COMPOSITIONS

The specification of the name of each composition (such as they appear within the block GEOMCOMP) is done by using the option -c or --compositions.
The list provided with this option must have the same size as the list provided with the option -P (volume ratios).
Examples:

- Composition with 35% FUEL, 55% WATER and 10% CLADDING:

  ```
  ./castor --poisson3d -T -P 0.35,0.55,0.1 -c FUEL,WATER,CLADDING
  ```

- Composition with 10% FUEL and 90% WATER:

  ```
  ./castor --inclusions3d -T -P 0.1,0.9 -c FUEL,WATER
  ```

The list is set to WATER,FUEL by default, i.e. the composition 0 is named WATER and the composition 1 is named FUEL.

### 45.3.4 TRANSLATION

As explained in Subsection 45.2.3, the geometries are always centered at the origin. However, it is possible to apply a translation to the center for the export to TRIPOLI-4. In order to move the center to the position

$(P_x, P_y, P_z)$, the user can use the commande `-p` or `--translation`.

Example: geometry centered at $(0.5, -4.5, 32.3)$ within the block `GEOMETRY`:

```
./castor --poisson3d -T -p 0.5,-4.5,32.3
```

By default, no translation is applied to the center of the geometry.

### 45.3.5 BOUNDARY CONDITIONS

The user can add boundary conditions (see also Chapter 9) all around the container by two possible options :

- the option `-r` for reflections everywhere;
- the option `-b` for a flexible set of boundary conditions.

Please note that:

1. the options `-b` and `-r` exclude each other;
2. if none of these options is used, no file is produced for the block `BOUNDARY_CONDITION`.

First, the option `-r` or `--reflection` adds reflective boundaries conditions to the whole container.
Examples:

```
./castor --poisson3d -C sphere,10. -T -r
```

```
./castor --poisson3d -C box,10. -T -r
```

Then, the option `-b` or `--boundaries` enables to apply specific boundary conditions on each boundary, with TRIPOLI-4 keywords (consistent with the language selected as an option for TRIPOLI-4).

For a sphere, a list of one single boundary conditions has to be provided.
Examples:

- apply leakage on the sphere:

  ```
  ./castor --poisson3d -C sphere,10. -T -b LEAKAGE
  ```

- apply reflection on the sphere:

  ```
  ./castor --poisson3d -C sphere,10. -T -b REFLECTION
  ```

For a cylinder, a list of 3 boundary conditions has to be provided (lower surface, upper surface, lateral surface).
Examples:

- apply leakage on $x = x_{\min}$, leakage on $x = x_{\max}$ and reflection on the lateral surface:

  ```
  ./castor --poisson3d -C xcylinder,20.,5. -T -b LEAKAGE,LEAKAGE,REFLECTION
  ```

- apply leakage on $y = y_{min}$, reflection on $y = y_{max}$ and leakage on the lateral surface:

```
./castor --poisson3d -C ycylinder,20.,5. -T -b LEAKAGE,REFLECTION,LEAKAGE
```

- apply reflection on $z = z_{min}$, reflection on $z = z_{max}$ and leakage on the lateral surface:

```
./castor --poisson3d -C zcylinder,20.,5. -T -b REFLECTION,REFLECTION,LEAKAGE
```

For a box, a list of 6 boundary conditions has to provided ($x_{min}$, $x_{max}$, $y_{min}$, $y_{max}$, $z_{min}$, $z_{max}$).
Examples:

- apply reflection on $x = x_{min}$, leakage on $x = x_{max}$, leakage on $y = y_{min}$, reflection on $y = y_{max}$, leakage on $z = z_{min}$, leakage on $z = z_{max}$ :

```
./castor --poisson3d -C box,10. -T
         -b REFLECTION,LEAKAGE,LEAKAGE,REFLECTION,LEAKAGE,LEAKAGE
```

- apply leakage on $x = x_{min}$, leakage on $x = x_{max}$, leakage on $y = y_{min}$, reflection on $y = y_{max}$, leakage on $z = z_{min}$, leakage on $z = z_{max}$ :

```
./castor --poisson3d -C box,10. -T
         -b LEAKAGE,LEAKAGE,LEAKAGE,REFLECTION,LEAKAGE,LEAKAGE
```

### 45.3.6 LANGUAGE

The outputs exported for TRIPOLI-4 contain specific keywords that should be consistent with the language selected as an option for TRIPOLI-4. This is why it can be necessary to use the option `-l` or `--language`. Three languages are possible: `english`, `french` or `old`.

Examples:

```
./castor --poisson3d -T -l english
./castor --poisson3d -T -l french
./castor --poisson3d -T -l old
```

By default, the language is set to `old`.

### 45.3.7 EXPORT FOR THE VISUALIZER T4G

The export of input files (concatenation of `GEOMETRY` and `GEOMCOMP` blocks) for the visualizer T4G is optional. The user can activate the production of such files with the option `-g` or `--t4g`.

Example:

```
./castor --poisson3d -T -g
```

Each output file is called `t4g_i` (where *i* corresponds to the index of the realization).

### 45.3.8 CONNECTIVITY MAP

A connectivity map is produced by default. The user can deactivate the connectivity map in the `GEOMETRY` block with the option '-n' or '–no-map'.

Example:

```
./castor --poisson3d -T -n
```

### 45.3.9 MESH (SPHERICAL INCLUSIONS ONLY)

Finally, for the specific case of spherical inclusions, the option `-m` or `--mesh-t4` enables to:

- reset a new mesh for the export to TRIPOLI-4 or T4G if a previous mesh has already been used for the construction;
- add a mesh for the export to TRIPOLI-4 or T4G if no mesh has been used for the construction (see Figure 45.4).

The option `-d` or `--no-mesh-t4` enables to remove the mesh that has been used for the construction (for the export to TRIPOLI-4/T4G). If none of these two options (`-m` or `-d`) is added, the input files for TRIPOLI-4/T4G are obtained with the mesh used during the construction.

Examples:

- a mesh $20 \times 20 \times 20$ is used to speed-up the construction and a mesh $50 \times 50 \times 50$ is used to speed-up the transport in TRIPOLI-4:

  ```
  ./castor --poisson3d -T -M 20,20,20 -m 50,50,50
  ```

- no mesh is used to speed-up the construction and a mesh $50 \times 50 \times 50$ is used to speed-up the transport in TRIPOLI-4:

  ```
  ./castor --poisson3d -T -m 50,50,50
  ```

- a mesh $20 \times 20 \times 20$ is used to speed-up the construction and no mesh is used for the transport in TRIPOLI-4:

  ```
  ./castor --poisson3d -T -M 20,20,20 --no-mesh-t4
  ```

- a single mesh $20 \times 20 \times 20$ is used to speed-up both the construction and the transport in TRIPOLI-4:

  ```
  ./castor --poisson3d -T -M 20,20,20
  ```

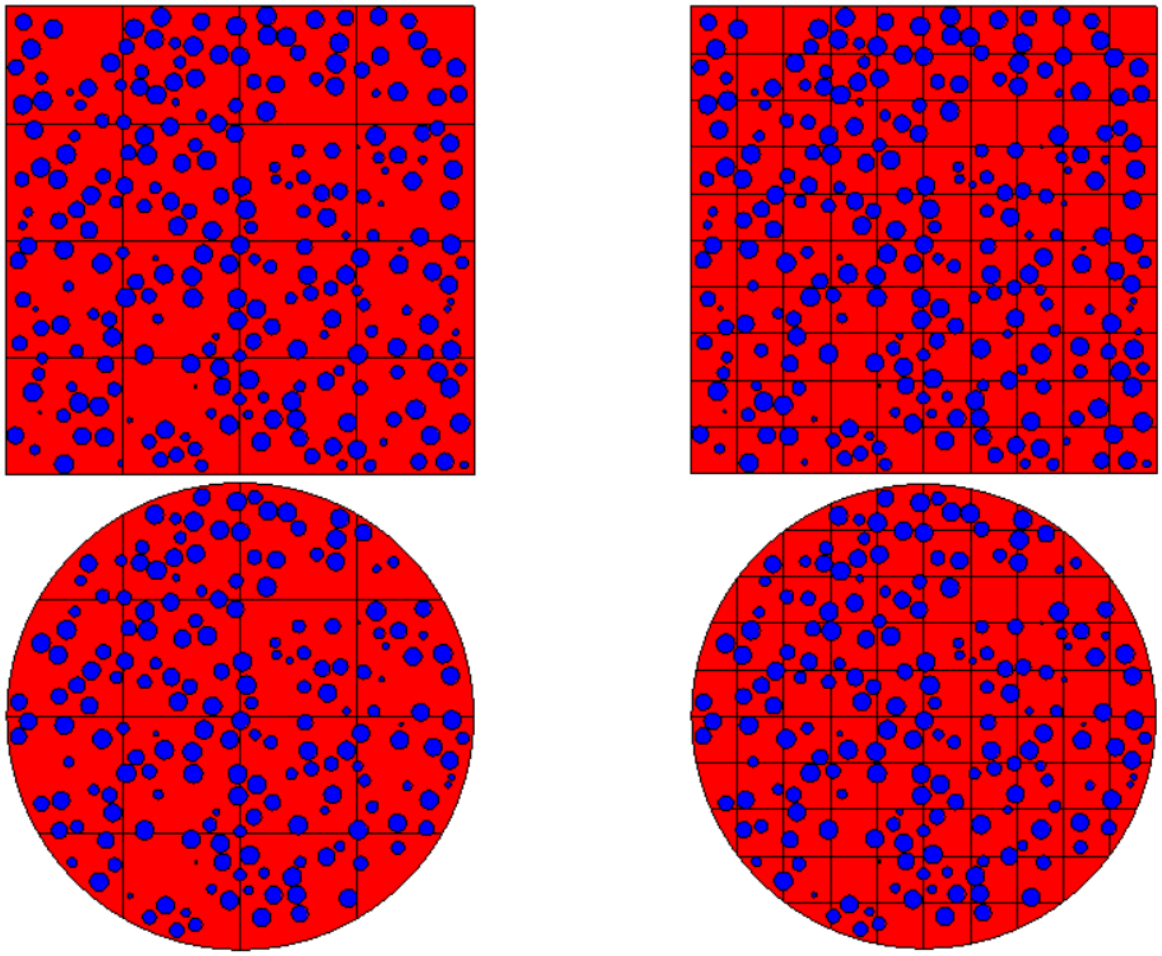


Figure 45.4: Examples of mesh in the visualizer T4G for spherical inclusions: $4 \times 4 \times 4$ (left) and $10 \times 10 \times 10$ (right), for a box container (top) or a spherical container (bottom).

## 45.4 EXPORT INPUT FILES FOR VTK

### 45.4.1 ACTIVATION AND TARGET REPOSITORY

The option `-V` or `--vtk` activates the generation of inputs files for VtK.

The option `-v` or `--vpath` specifies a repository where outputs will be exported.
Example:

```
./castor --inclusions3d -V -v repo_vtk
```

By default, output files will be exported into the current directory.
Each output file is named `paraview_i.vtk` where *i* corresponds to the index of the realization.

It is important to note that:

- for spherical inclusions, only spheres are visible on the visualization, but not the container itself;
- for Poisson or Poisson-Box tessellations, the spherical/cylindrical container is applied *a posteriori* by adding a wrapping surface (only for the export for TRIPOLI-4). Indeed, the (vertex-based) construction of the geometry is based on the smallest bounding box containing the spherical/cylindricalcontainer. Since the representation of the geometry in VtK is also vertex-based, the export for VtK enables to show the full geometry within the parallelepedic box and not within the container used for TRIPOLI-4.

### 45.4.2 START RANK AND END RANK

The start rank and end rank of generation of these output files can be chosen by the user with options `--vstartrank` and `--vendrank` (or `-i`).
Examples:

- VtK files are produced between the 21$^{st}$ and the 30$^{th}$ realization:

  ```
  ./castor --inclusions3d -N 100 -V -v repo_vtk --vstartrank 21 --vendrank 30
  ```

- Only 5 first VtK files will be produced:

  ```
  ./castor --inclusions3d -N 100 -V -v repo_vtk -i 5
  ```

Please note that VtK files can be quite heavy, given the geometrical parameters, whence the possibility to export only a selection of geometries into VtK format. By default, all the generated geometries are exported to VtK files if `-V` is activated.

### 45.4.3 VISUALIZATION OPTION

By default, the geometry is visualized with a given color for each composition. However, the user can choose to visualize "uncolored" geometries with the option `-u` or `--randomize-color` (see illustration in

Figure 45.5). That requires to randomize the labels for the visualization, in order to remove correlations between labels and indices of volumes.

Example:

```
./castor --poisson3d -N 100 -V -v repo_vtk -u
```

**Warning:** this option leads to an offset of the random seed of the generator.

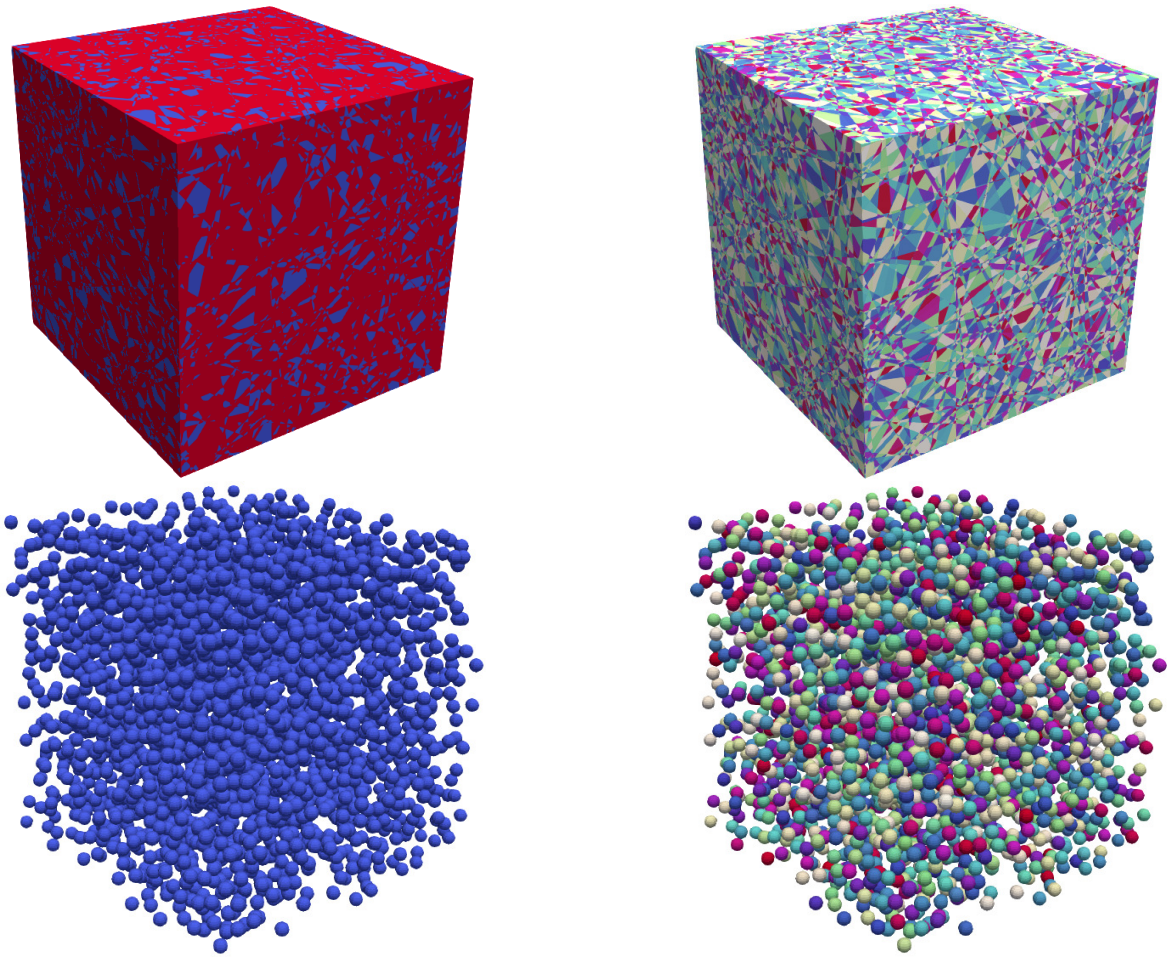


Figure 45.5: Visualization with ParaView without (left) and with (right) option `-u` activated.

### 45.4.4 RESOLUTION OF THE SPHERES (SPHERICAL INCLUSIONS ONLY)

Finally, a specific option `-k` or `--resolution` can be used for spherical inclusions only. This option enables to choose the number of discretized angles for the decomposition of any spherical inclusion (the minimal value being 4).

Example:

```
./castor --inclusions3d -V -v repo_vtk -k 20
```

By default, this option is equal to 10.

Nevertheless, if the exported file is too heavy (which depends directly on the number of spheres), it could be necessary to reduce this parameter of resolution. Conversely, it could be useful to increase the resolution in order to improve the visualization (see Figure 45.6.

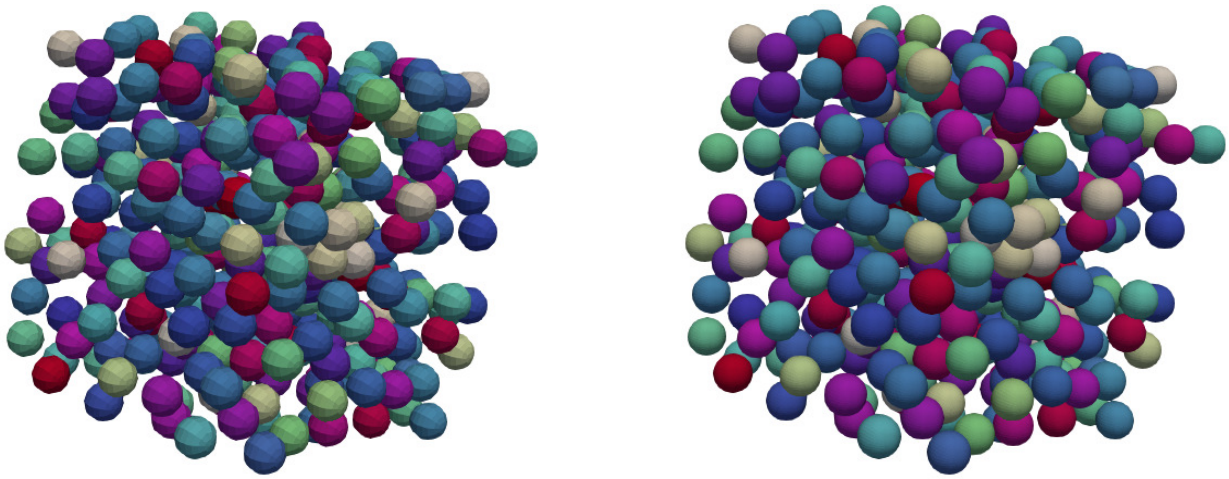


Figure 45.6: Visualization with ParaView with `-r 10` (left) and with `-r 40` (right).

## 45.5 STATISTICAL PROPERTIES

### 45.5.1 ACTIVATION AND TARGET REPOSITORY

The option `-S` or `--statistics` activates the calculation of statistical properties related to generated geometries (i.e. chord length first-order moments and distributions).

The option `-s` or `--spath` specifies a repository where outputs will be exported.

Example:

```
./castor --inclusions3d -S -s repo_stats
```

By default, output files will be exported into the current directory.

This option does not work for spherical/cylindrical containers.

At the end of the calculation, the following ensemble-averaged quantities are displayed (in the console):

- the average chord length through volumes and the associated standard deviation;
- the average chord length through each composition $i$ and the associated standard deviation.

The corresponding histograms are exported to the target repository. They are named as follows:

- `histogram_chord_length.txt` for the uncolored chord length;
- `histogram_chord_length_color_i.txt` for the chord length through composition $i$.

### 45.5.2 NUMBER OF TEST LINES

The number of test lines used to compute the chord length per realization can be chosen with the option `-j` or `--nb-lines`.

Example for 100 test lines:

```
./castor --poissonbox3d -S -j 100
```

The number of test lines is equal to 1 by default.

### 45.5.3 OPTIONS FOR PRE-DEFINED GRIDS

The user can define grids used to tally the chord length distribution, with the options `--minima`, `--maxima` and `--nb-bins`. For a $n$-ary mixture, the user has to provide a list of size $n+1$ for each of these options.

In the following example:

```
./castor --poissonbox3d -S --minima 0.,1.,4.5 --maxima 40.,80.,100.
         --nb-bins 1500,500,200
```

- the grid used for the scoring of the uncolored chord length distribution ranges from 0 to 40 with 1500 bins;
- the grid used for the scoring of the colored chord length distribution through composition 0 ranges from 1 to 80 with 500 bins;
- the grid used for the scoring of colored chord length distribution through composition 1 ranges from 4.5 to 100 with 200 bins.

By default, all grids range from 0 to $L_{\max}$ with 1000 bins, where $L_{\max}$ corresponds to the maximal chord length in the container.

# Appendix A

# EXAMPLE OF GEOMETRY

## A.1 INTRODUCTION

This appendix presents a case with geometry developed using the available lattice functionality: a hexagonal lattice with modification of the cells, included in a parallelepipedic lattice cell, in order to illustrate calculations relating to a lattice of lattices. The construction is described step-by-step to show the different stages involved.
Two other examples of relatively complex geometry are described in this appendix:

- a homogenizer:
  - surface-based geometry,
  - combinatorial geometry.
- an eighth of a transport cask cylinder:
  - surface-based geometry,
  - combinatorial geometry.

## A.2 A HEXAGONAL AND PARALLELEPIPEDIC LATTICE, A LATTICE OF LATTICES

The geometry is shown on the following page.

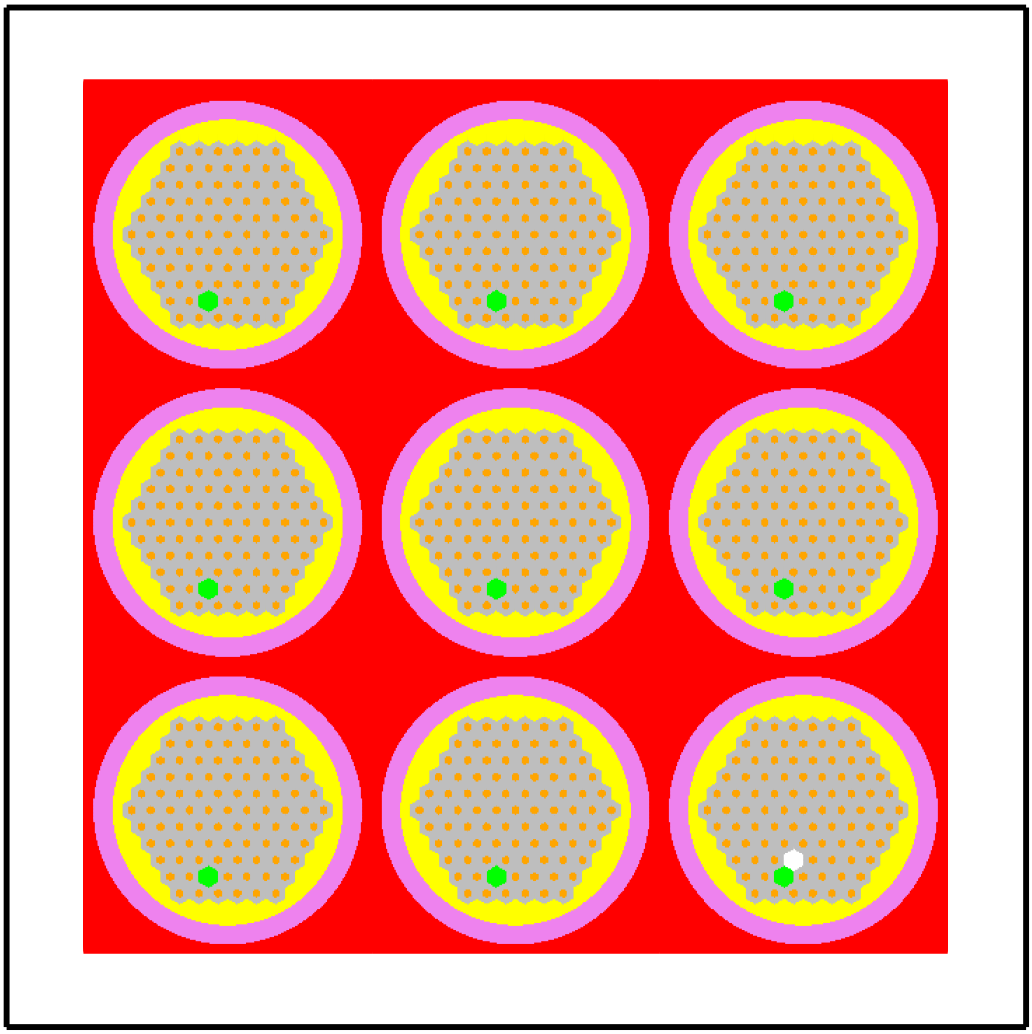
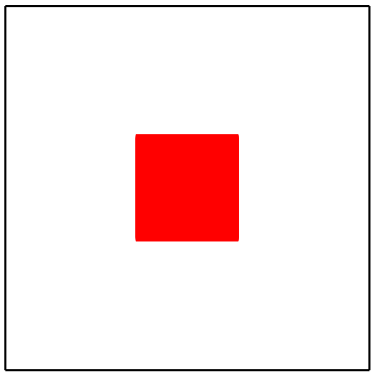


Figure A.1: Geometry of Case 2

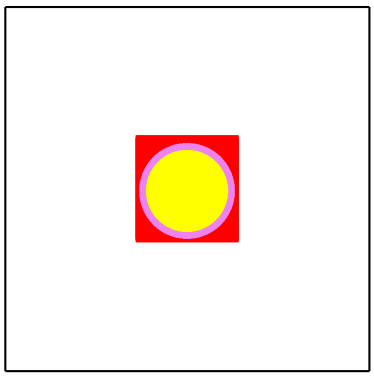
The first step consists in constructing the basic parallelepipedic cell:

```
TYPE 100 BOX 15 15 3
// Basic cell of the
parallelepipedic lattice
VOLU 100 COMBI 100 0 0 0 ENDV
```



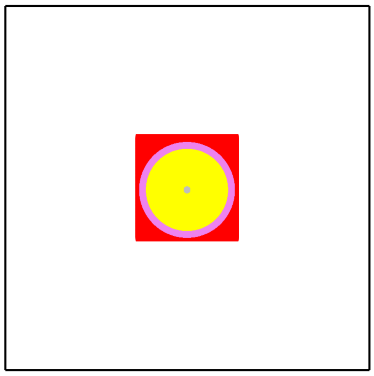
Then two concentric cylinders are created

```
TYPE 101 CYLZ 7 1
TYPE 102 CYLZ 6 1
// Cylinders
VOLU 101 COMBI 101 0 0 0 SMASH 1
100 ENDV
VOLU 102 COMBI 102 0 0 0 SMASH 1
101 ENDV
```



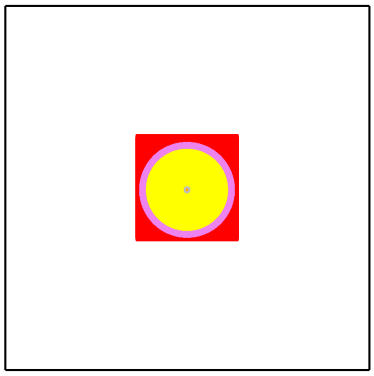
Then the hexagonal cell is created

```
TYPE 1 HEXAZ 1 1 0
// Hexagonal cell
VOLU 1 COMBI 1 0 0 0 SMASH 1 102
ENDV
```



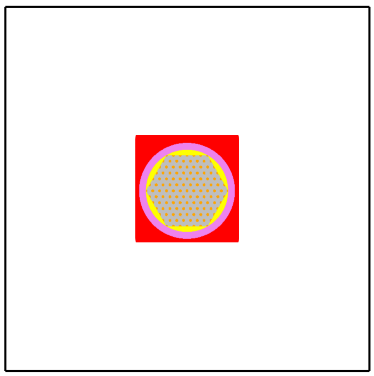
This cell itself contains a hexagonal prism

```
TYPE 2 HEXAZ 0.4 0.4 0
// Volume included in the
hexagonal cell
VOLU 2 COMBI 2 0 0 0 SMASH 1 1
ENDV
```



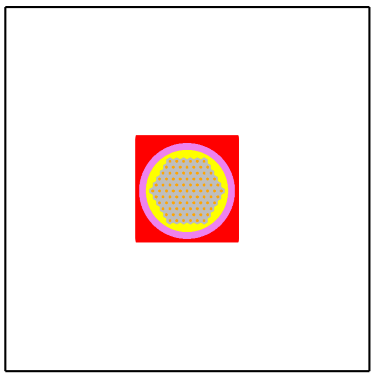
It is now possible to create the hexagonal lattice. Since it is to be centered on (0,0,0), it is sufficient to declare that the basic cell is the cell (6,6,1)

```
VOLU 3 RESH VOLU 1 6 1
BASIS 6 6 1
ENDV
```



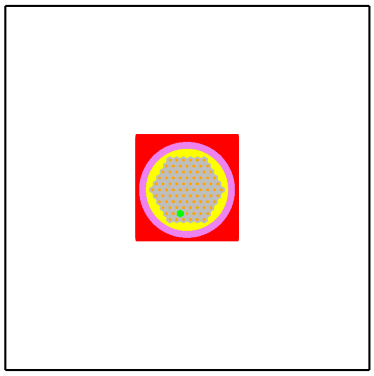
It is only the complete cells that are actually wanted; therefore, option `MATCH` needs to be used. It should be noted that the lattice cannot override volume 102 (incompatible with option `MATCH`).

```
VOLU 3 RESH VOLU 1 6 1
BASIS 6 6 1
MATCH
ENDV
```



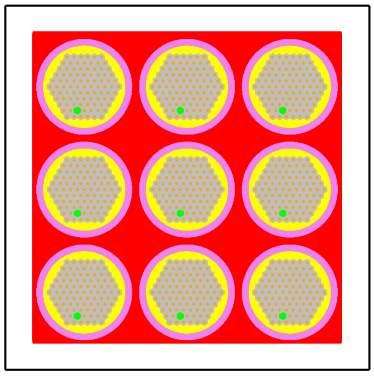
It is necessary to individualize the cell (3,2,1) using option `CELL`. It has to be declared that the cell overrides 102, as we want to create a lattice of lattices.

```
VOLU 4 CELL 3 2 1
LATTICE 3 REPLACE 1
SMASH 1 102
ENDV
```



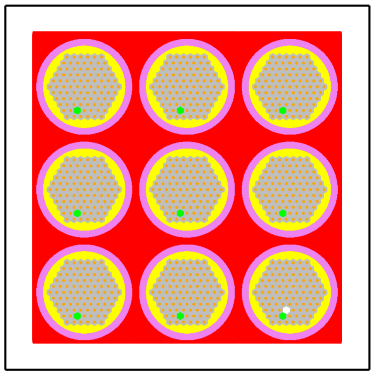
The lattice of lattices is now created. The basic cell is the central cell (2,2,1).

```
VOLU 1000 RESC
VOLU 100 3 3 1
BASIS 2 2 1
ENDV
```



Cell (4,3,1) is then excluded from the hexagonal lattice in cell (3,1,1) of the square lattice using the option LOWER_CELL.

```
VOLU 40 LOWER_CELL DEPTH 2
LATTICE 3 VOLU 1 CELL 4 3 1
LATTICE 1000 VOLU 100 CELL 3 1 1
ENDV
```



Display and creation of a captionless encapsulated PostScript file is obtained by:

```
GRAPH
-25 -25 0
1 0 0
0 1 0
50 50
4
```

# A.3 HOMOGENIZER

## A.3.1 DEFINING THE COMBINATORIAL GEOMETRY OF A HOMOGENIZER

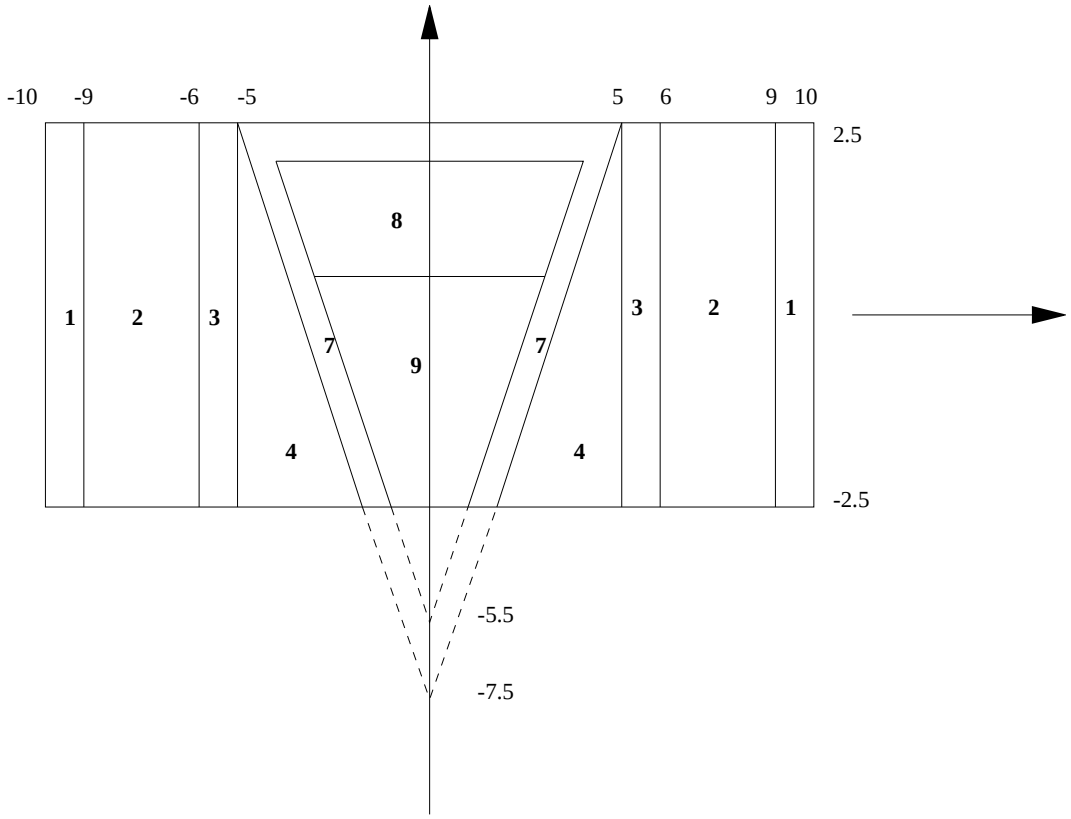



Figure A.2: Homogenizer described using a combinatorial geometry

```
TITLE TEST-GEOM

TYPE 1 CYLZ 10 2.5
TYPE 2 CYLZ 9 2.5
TYPE 3 CYLZ 6 2.5
TYPE 4 CYLZ 5 2.5
TYPE 5 BOX 10 10 10
TYPE 6 CONE 26.565 0 0 1

// Cylinders

VOLU 1 COMBI 1 0 0 0 ENDV
VOLU 2 COMBI 2 0 0 0 SMASH 1 1 ENDV
VOLU 3 COMBI 3 0 0 0 SMASH 2 1 2 ENDV
VOLU 4 COMBI 4 0 0 0 SMASH 3 1 2 3 ENDV

VOLU 5 COMBI 5 0 0 -7.5 FICTIVE ENDV
VOLU 6 COMBI 5 0 0 -7.  FICTIVE ENDV

// Cones

VOLU 7 COMBI 6 0 0 -7.5 10
SMASH 4 1 2 3 4 VMINUS 1 5
ENDV
VOLU 8 COMBI 6 0 0 -5.5 7.5
SMASH 5 1 2 3 4 7 VMINUS 1 6
ENDV
VOLU 9 COMBI 6 0 0 -5.5 6
SMASH 6 1 2 3 4 7 8 VMINUS 1 6
ENDV

GRAPH
-15 1 -3
1 0 0
0 0 1
30 4
0

ENDGEOM
```

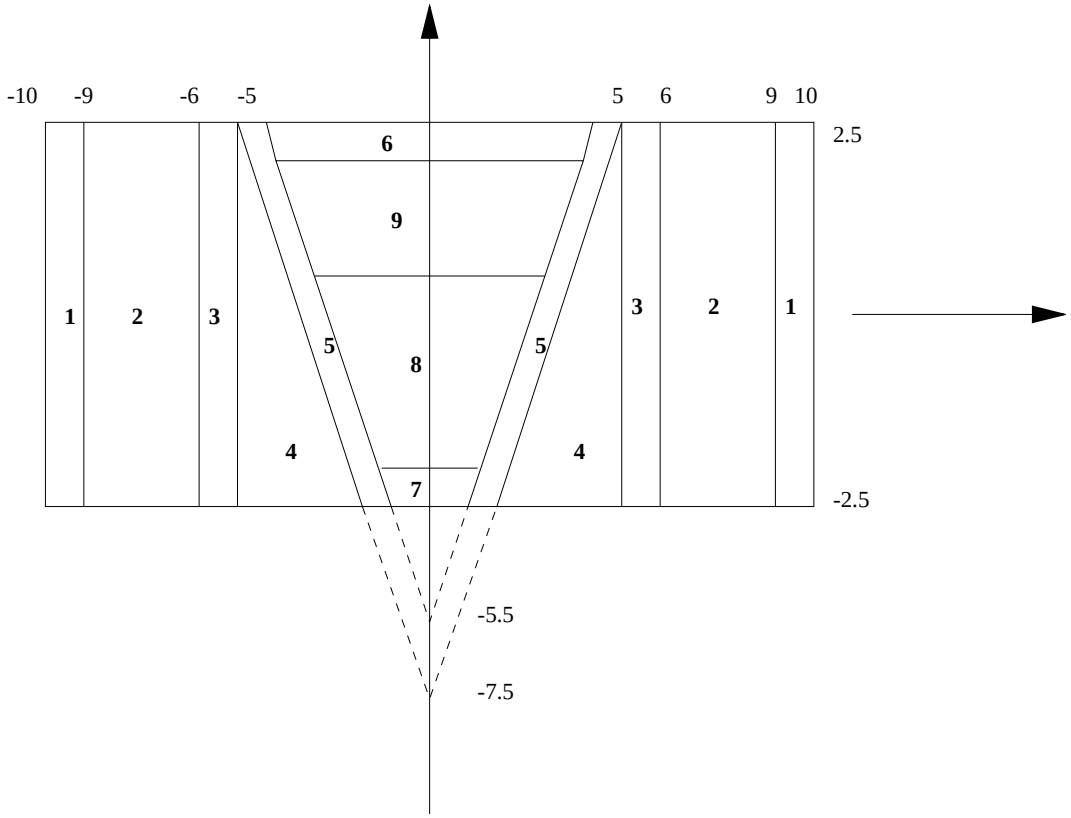



Figure A.3: Homogenizer described using a surface-based geometry

### A.3.2 DEFINING THE SURFACE-BASED GEOMETRY OF A HOMOGENIZER

```
GEOMETRY
TITLE TEST-GEOM

/*
* homogenizer
*/

SURF 1 CYLZ 0 0 10
SURF 2 PLANEZ 2.5
SURF 3 PLANEZ -2.5
SURF 4 CYLZ 0 0 9
SURF 5 CYLZ 0 0 6
SURF 6 CYLZ 0 0 5
SURF 7 PLANEZ -2
SURF 8 PLANEZ 0.5
```

```
SURF 9 PLANEZ 2
SURF 10 QUAD 1 1 -0.25 0 0 0 0 0 -3.75 -14.0625
SURF 11 QUAD 1 1 -0.25 0 0 0 0 0 -2.75 -7.5625

VOLU 1 EQUA MINUS 2 1 2 PLUS 2 3 4
ENDV
VOLU 2 EQUA MINUS 2 2 4 PLUS 2 3 5
ENDV
VOLU 3 EQUA MINUS 2 2 5 PLUS 2 3 6
ENDV
VOLU 4 EQUA MINUS 2 2 6 PLUS 2 3 10
ENDV
VOLU 5 EQUA MINUS 2 10 2 PLUS 2 3 11
ENDV
VOLU 6 EQUA MINUS 2 11 2 PLUS 1 9
ENDV
VOLU 7 EQUA MINUS 2 11 7 PLUS 1 3
ENDV
VOLU 8 EQUA MINUS 2 11 8 PLUS 1 7
ENDV
VOLU 9 EQUA MINUS 2 11 9 PLUS 1 8
ENDV
ENDGEOM
```

## A.4 TRANSPORT CASK

The cask exhibits symmetry, and it is, therefore, possible to only describe one eighth of it.
Note: Although the current geometry makes it possible to construct 1/8 of a cask using combinatorial geometry, it is not yet possible to associate reflective boundary conditions with it. This is because the surfaces of the volume thus created result from truncation by operators.

### A.4.1 DEFINING THE COMBINATORIAL GEOMETRY OF A TRANSPORT CASK

```
GEOMETRY
TITLE TEST-GEOM

COMMENT
transport cask
COMMENT

TYPE 1 BOX 1000 1000 1000
TYPE 2 CYLZ 12 2.5
```

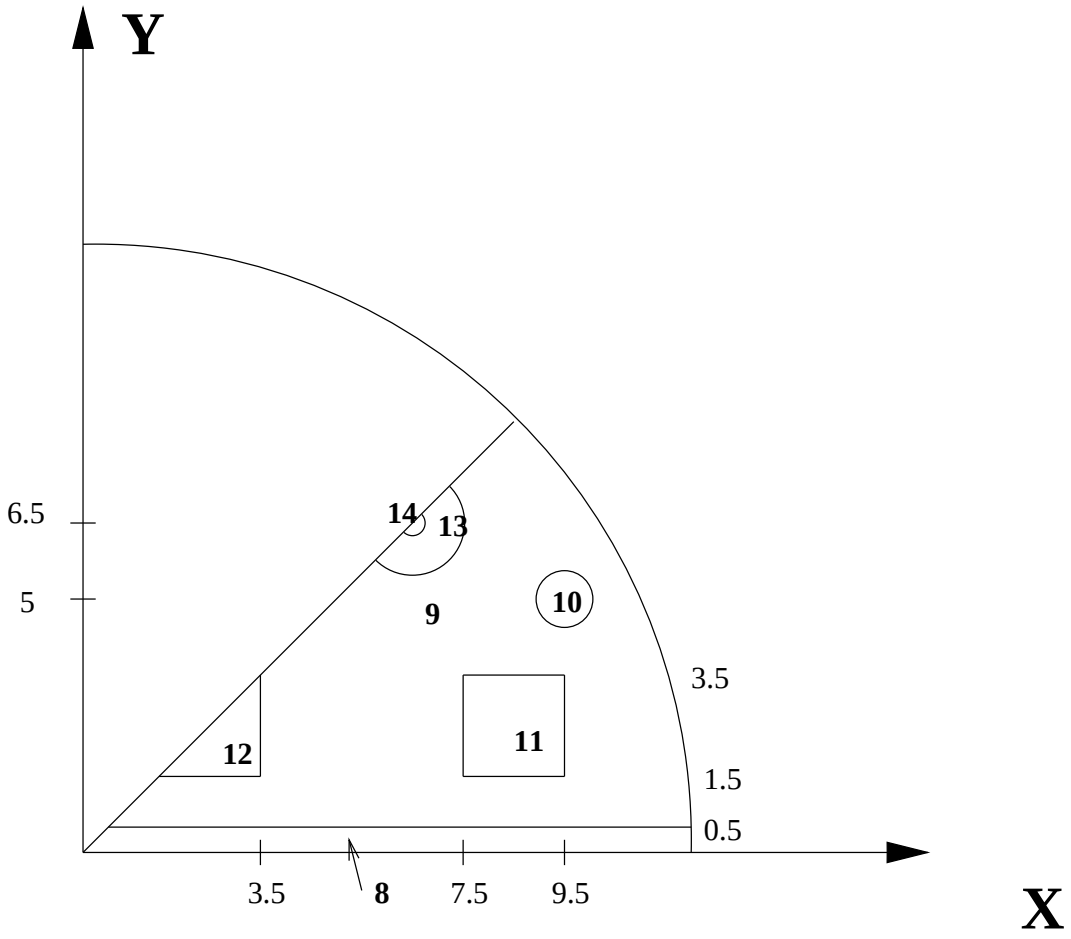



Figure A.4: Cask described using a combinatorial geometry

```
TYPE 3 BOX 24 12 5
TYPE 4 CYLZ 1 2.5
TYPE 5 CYLZ 0.5 2.5
TYPE 6 CYLZ 0.3 2.5
TYPE 7 BOX 2 2 5
TYPE 8 BOX 12 1.4 5
TYPE 9 BOX 12 24 5
TYPE 10 BOX 12 12 5

VOLU 1 COMBI 1 0 0 0
ENDV
VOLU 2 COMBI 3 0 -6 0
FICTIVE
ENDV
```

```
VOLU 3 COMBI 9 -6 0 0
FICTIVE
ENDV
VOLU 4 COMBI 2 0 0 0
VMINUS 2 2 3
FICTIVE
ENDV
VOLU 5 COMBI 8 6 0.25 0
FICTIVE
ENDV
VOLU 6 COMBI 10 6 6 0
FICTIVE
ENDV
VOLU 7 ROTATION VOLU 6 0 0 1 45 0 0 0
FICTIVE
ENDV
VOLU 8 COMBI 8 6 0.25 0
VMINUS 1 7 INTE 2 4 5 SMASH 1 1
ENDV
VOLU 9 COMBI 2 0 0 0
VMINUS 4 2 3 7 8 SMASH 1 1
ENDV
VOLU 10 COMBI 5 9.5 5 0
SMASH 2 1 9
ENDV
VOLU 11 COMBI 7 8.5 2.5 0
SMASH 2 1 9
ENDV
VOLU 12 COMBI 7 2.5 2.5 0
VMINUS 1 7 SMASH 2 1 9
ENDV
VOLU 13 COMBI 4 6.5 6.5 0
VMINUS 1 7 SMASH 2 1 9
ENDV
VOLU 14 COMBI 6 6.5 6.5 0
VMINUS 1 7 SMASH 3 1 9 13
ENDV
ENDGEOM
```

### A.4.2 DEFINING THE SURFACE-BASED GEOMETRY OF A TRANSPORT CASK

```
GEOMETRY
```

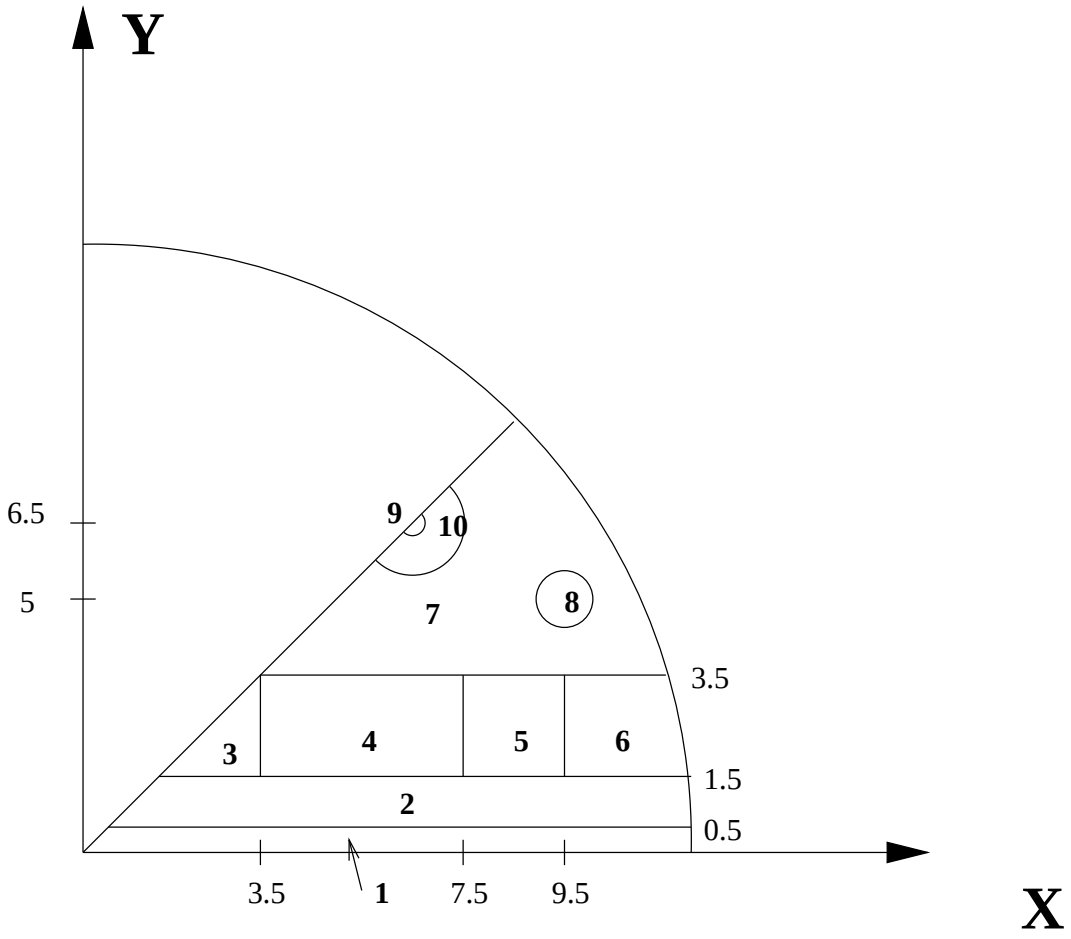



Figure A.5: Cask in surface-based form

```
TITLE TEST-GEOM
// transport cask

SURF 1 CYLZ 0 0 12
SURF 2 PLANY 0
SURF 3 PLAN 1 -1 0 0
SURF 4 PLANY 0.5
SURF 5 PLANY 1.5
SURF 6 PLANY 3.5
SURF 7 PLANX 9.5
SURF 8 PLANX 7.5
SURF 9 PLANX 3.5
SURF 10 CYLZ 9.5 5 0.5
SURF 11 CYLZ 6.5 6.5 1
```

```
SURF 12 CYLZ 6.5 6.5 0.3
SURF 13 PLANEZ -5
SURF 14 PLANEZ 5

VOLU 1 EQUA MINUS 3 1 4 14 PLUS 3 2 3 13
ENDV
VOLU 2 EQUA MINUS 3 1 5 14 PLUS 3 3 4 13
ENDV
VOLU 3 EQUA MINUS 2 9 14 PLUS 3 3 5 13
ENDV
VOLU 4 EQUA MINUS 3 6 8 14 PLUS 3 5 9 13
ENDV
VOLU 5 EQUA MINUS 3 7 6 14 PLUS 3 5 8 13
ENDV
VOLU 6 EQUA MINUS 3 1 6 14 PLUS 3 7 5 13
ENDV VOLU 7 EQUA MINUS 2 1 14 PLUS 5 6 3 10 11 13
ENDV
VOLU 8 EQUA MINUS 2 10 14 PLUS 1 13
ENDV
VOLU 9 EQUA MINUS 2 12 14 PLUS 2 3 13
ENDV
VOLU 10 EQUA MINUS 2 14 11 PLUS 3 3 12 13
ENDV
ENDGEOM
```

# Appendix B

# DATA ENTRY SEQUENCE

A) **GEOMETRY**

B) **TITLE** `title` (mandatory)

C) If required to specify

- a **TYPE**, go to C-1
- a surface **SURF**, go to C-2
- a volume **VOLU**, go to C-3

If all descriptions have been given, go to E (or D then E) for a 2D-view, otherwise directly to G.

C-1) **TYPE** `numtype` (Defines the type of number `numtype`) In order to obtain:

- a **BOX** go to C-1.a,
- a **SPHERE** go to C-1.b,
- a cylinder with axis X, Y or Z: **CYLX**, **CYLY** or **CYLZ** go to C-1.c,
- a cylinder with any axis: **CYL**, go to C-1.d,
- a **CONE**, go to C-1.e,
- a hexagonal prism **HEXAX**, **HEXAY** or **HEXAZ**, go to C-1.f,
- a **PARALLELEPIPEDE**, go to C-1.g,
- a **TORUS**, go to C-1.h.

Then return to C.

C-1.a) **BOX** `sidex sidey sidez`.
C-1.b) **SPHERE** `radius`.
C-1.c) **CYL[X,Y,Z]** `half-height_radius`.
C-1.d) **CYL** `half-height_radius ux uy uz` with `(ux,uy,uz)` an axis director.
C-1.e) **CONE** `apex_angle ux uy uz` with `(ux,uy,uz)` apex director.
C-1.f) **HEXA[X,Y,Z]** `inscribed_diameter height angle`.

C-1.g) **PARALLELEPIPEDE** side(1) side(2) side(3)
axis1(x) axis1(y) axis1(z)
axis2(x) axis2(y) axis2(z)
axis3(x) axis3(y) axis3(z).

C-1.h) **TORUS** ux uy uz major-radius minor-radius.

C-2) **SURF** numsurf (Defines the surface of number numsurf). In order to obtain:

- a plane parallel to the planes of reference frame **PLANE[X,Y,Z]**, go to C-2.a,
- any plane **PLANE**, go to C-2.b,
- a **SPHERE**, go to C-2.c,
- a cylinder of axis $X$, $Y$ or $Z$: **CYL[X,Y,Z]**, go to C-2.d,
- a cone of axis $X$, $Y$ or $Z$: **CONE[X,Y,Z]**, go to C-2.e,
- any quadratic **QUAD**, go to C-2.f.

Then return to C.

C-2.a) **PLANE[X,Y,Z]** side where side is the side of the plane (if, for example, **PLANEX**: X=side).

C-2.b) **PLANE** a b c d plane of equation $ax+by+cz+d=0$.

C-2.c) **SPHERE** x y z radius with $(x,y,z)$ the coordinates of the center and radius the radius of the sphere.

C-2.d) **CYL[X,Y,Z]** coordinate1 coordinate2 radius
where coordinate1 and coordinate2 are the coordinates of a point on the axis
(coordinate1,coordinate2) = $(y,z)$ for **CYLX**, $(x,z)$ for **CYLY**, $(x,y)$ for **CYLZ**.

C-2.e) **CONE**[X,Y,Z] xc yc zc half-angle.

C-2.f) **QUAD** a b c d e f g h i j quadratic equation $ax^2+by^2+cz^2+dxy+eyz+fxz+gx+hy+iz+j=0$.

C-3) **VOLU** numvolu (Defines a volume of number numvolu) In order to obtain:

- a combinatorial geometry, go to C-3.a,
- a surface-based geometry, go to C-3.b,
- a rotation, go to C-3.c,
- a lattice, go to C-3.d,
- a particular cell, go to C-3.e.

C-3.a) **COMBI** numtype (Defines a volume on the basis of the type of number numtype)

C-3.a.1) If the type is not based on a cone
centrex centrey centrez (coordinates of center of volume created).
Go to C-3.f.

C-3.a.2) If the type is based on a cone
sommetx sommety sommetz height (coordinates of the cone apex + height of cone).
Go to C-3.f.

C-3.a.3) To center the volume relative to another one, add **V_ORIGIN** numvol.

C-3.a.4) To centre the volume relative to a cell of a lattice, add
**M_ORIGIN** ni nj nk **LATTICE** numlattice.

C-3.b) **EQUA** (Defines a volume using equations)

C-3.b.1) **PLUS** nbplus (with nbplus the number of surfaces with respect to which the volume is considered positive, if nbplus=0 do not write this line and go to C-3.b.2) surf1...surfn (list of the surfaces by number). Go to C-3.b.2.

C-3.b.2) **MINUS** nbmoins (with nbmoins the number of surfaces with respect to which the volume is considered negative, if nbmoins=0 do not write this line and go to C-3.f) surf1...surfn (list of the surfaces by number).
Go to C-3.f.

C-3.c) **ROTATION** (Defines a rotation)
**VOLU** numvolu ux uy uz θ Ωx Ωy Ωz.
Rotation of the volume of number numvolu of an angle θ around the axis described by the point (Ωx,Ωy,Ωz) and the vector (ux,uy,uz).
Go to C-3.f.

C-3.d) If the cell is parallelepipedic, go to C-3.d.1, if hexagonal go to C-3.d.2.

C-3.d.1) **RESC VOLU** numvolu-ele nx ny nz. Repetition of the volume of number numvolu-ele nx (ny and nz respectively) times along axis *X* (*Y* and *Z*, respectively).

C-3.d.2) **RESH VOLU** numvolu-ele n1 n3. Repetition of the volume n1 (n2=n1 and n3, respectively) times along axis 1 (2 and 3, respectively).
Add **MATCH** if only complete cells are required.

C-3.d.3) Add **EXCEPT** nbcells ni(1) nj(1) nk(1) ni(2) nj(2) nk(2) ... to delete certain cells.

C-3.d.4) Add **KEEP** nbcells ni(1) nj(1) nk(1) ni(2) nj(2) nk(2) ... to only keep certain cells.

C-3.e) In order to redescribe a cell, go to C-3.e.1, for a sub-cell (lattice of lattices), go to C-3.e.2

C-3.e.1) **CELL** ni nj nk **LATTICE** numlattice **REPLACE** numvol

C-3.e.2) **LOWER_CELL DEPTH** depth
**LATTICE** numlattice(the-smallest) **VOLU** numvol **CELL** ni nj nk
...
**LATTICE** numlattice(the-greatest) **VOLU** numvol **CELL** ni nj nk

C-3.f) Choice of an operator

C-3.f.1) If the volume being defined overrides the volumes already described (the non-zero intersection case), continue; otherwise, go directly to C-3.f.2.
**SMASH** nbvolu (overrides nbvolu volumes)
volu1...volun (list of the volumes by number)

C-3.f.2) If the volume being defined is overridden by volumes that were previously defined (the non-zero intersection case), continue; otherwise, go directly to C-3.f.3.

**VMINUS** nbvolu (truncation of intersections with nbvolu volumes)
volu1...volun (list of the volumes by number)

C-3.f.3) If the volume being defined only retains its intersection with previously defined (virtual) volumes, continue; otherwise, go directly to C-3.f.4.
**INTE** nbvolu (intersection with a set of nbvolu volumes)
volu1...volun (list of the volumes by number)

C-3.f.4) If the volume being defined is increased by the union with previously defined (virtual) volumes, continue; otherwise, go directly to C-3.f.5
**UNION** nbvolu (union with a set of nbvolu volumes)
volu1...volun (list of the volumes by number)

C-3.f.5) If the volume is virtual, add
**FICTIVE**;
otherwise, go directly to C-3.f.6.

C-3.f.6) If the volume is to be excluded in a lattice add
**VOL_EXCEPT**;
otherwise, go directly to C-3.f.7.

C-3.f.7) End of volume definition
**ENDV**.
If another structure is defined, go to C (for another volume, directly to C-3); otherwise, go to D.

D) In order to define the colors, continue; otherwise, go directly to E.
**COLOR** nb-colors
namecolor(1) nbvol(1) list-volumes(1)...
namecolor(n) nbvol(n) list-volumes(n)

E) **GRAPH**
Px Py Pz (origin of the 2D-cut)
Ux Uy Uz Vx Vy Vz (vectors defining the 2D-view)
LU LV (size of the window)
Flag (to generate a Postscript file, 1 (with captions) or 2 (without), 3 (with captions) or 4 (without) for encapsulated PostScript, otherwise 0)

F) **POSITION** x y z (point position request)

G) **ENDGEOM**

A) **VOLSURF**

B) To define surfaces, go to B-1); otherwise, go directly to B-2).

B-1) **SURFACE** `nb(surfaces)`
numvol$_1$(1) numvol$_2$(1) surface(1)...
numvol$_1$(n) numvol$_2$(n) surface(n)
(a surface is described by two volumes)

B-2) **VOLUME** `nb(volumes)`
numvol(1) volume(1)...
numvol(n) volume(n)

C) **END_VOLSURF**

A) **BOUNDARY_CONDITION**

B) `number_of_conditions`

C) `number_of_volume(i)` then one of the following options:

C-1) **REFLECTION** `surface(j)` (Reflection on surface j of volume i)

C-2) **TRANSLATION** `surface(j)` (Translation from surface j of i)

C-3) **COSINUS** `surface(j)` (Emission of uniform cosine from surface j of i) Return to C for another condition.

D) **END_BOUNDARY_CONDITION**

A) **COMPOSITION**

B) number_of_compositions

C) A composition is entered in one of the following ways

C-1) ) **POINT_WISE**

Electronic properties, optional, only to overwrite Tripoli automatic choice in electron/positron transport mode:

C-1.a ) NONPOLARIZABLE (non polarizable medium, insulator)

C-1.b ) CONDUCTOR (electric conductor medium)

C-1.c ) GASNTP (gaseous medium at normal temperature and pressure conditions, 1 atm-0°C)

temperature (pointwise cross-sections, ENDF format)
composition_name nb(isotopes)
isotope(1) nb_atoms(1)... (isotope(i) should be included in the library)
isotope(n) nb_atoms(n)

C-2) **MULTIGROUP_HOMOGENEOUS** temperature (**APOLLO2** cross-sections)
composition_name **APOLLO2** burnup
apollo2_cross-section_file_name

C-3) **MULTIGROUP_HOMOGENEOUS** temperature (TRIPOLI-4®cross-sections)
composition_name T4HOMOT4 0
TRIPOLI_cross-section_file_name

C-4) **FILE**

Electronic properties, optional, only to overwrite Tripoli automatic choice in electron/positron transport mode

C-4.a ) NONPOLARIZABLE (non polarizable medium, insulator)

C-4.b ) CONDUCTOR (electric conductor medium)

C-4.c ) GASNTP (gaseous medium at normal temperature and pressure conditions, 1 atm-0°C)

```
temperature
composition_name file_name
```

C-5) Return to C for another composition. If all compositions have been defined, go to D.

D) **END_COMPOSITION**

A) **GEOMCOMP**

B) Define a combination for a composition:
`composition_name` (name of a composition as set in **COMPOSITION**)
`volume_number_of_composition`
`volu1... volun` (list of the volumes by number)
Return to B for a new composition, go to C to end.

C) **END_GEOMCOMP**

A) **SOURCES_LIST** in order to impose the total norm to go to B; otherwise, go to C.

B) **NORM** `norm`

C) `nbsources` Go to D once for each source.

D) **SOURCE**

D-1) **NORM** `norm`. Only in order to normalize the source; otherwise, pass.

D-2) **COEFF** `coeff`. Only in order to multiply the norm by `coeff`; otherwise, pass.

D-3) **NEUTRON** or **PHOTON**, depending on the type of particle.

D-4) Point source, go to D-4.a; otherwise, to D-4.b

D-4.a) **POINT** `x y z` Source at $(x,y,z)$

D-4.b) **FACTORIZED FRAME** Choose one of the following:

- **CARTESIAN**
- **SPHERE**
- **CYLINDER**

then enter

```
Ox Oy Oz
ix iy iz
(jx jy jz) for CARTESIAN only
kx ky kz
```

**VOLUME** `nbvolumes volume1 ... volumen`
**GEOMETRIC_DISTRIBUTION**
In order to obtain a tabular source, go to C-4.b.2.1; for an analytical source go to C-4.b.2.2.

D-4.b.1.1) **TABULATED**
**TYPE** choose
`F_U_V_W` or `F_U_VW` or `F_UVW`
`VAR_U` then choose `X` or `Y` or `Z`
`nb(u)`
`u[1] ... u[nb(u)]`
`VAR_V` then choose `X` or `Y` or `Z`
`nb(v)`
`v1 ... v[nb(v)]`
`VAR_W` then choose `X` or `Y` or `Z`
`nb(w)`
`w[1] ... w[nb(w)]`
Depending on the **TYPE** choice:

```
F_U value[1] ... value[nb(u)-1]
F_V value[1] ... value[nb(v)-1]
F_W value[1] ... value[nb(w)-1]
```

or
F_U value[1] ... value[nb(u)-1]
F_VW value[1] ... value{[nb(v)-1]*[nb(w)-1]}
or
F_UVW value[1] ... value{[nb(u)-1]*[nb(v)-1]*[nb(w)-1]}

D-4.b.1.2) **ANALYTICAL**
**FUNCTION**=...;
**DOMAIN**=...;

D-5) **ANGULAR_DISTRIBUTION** Choose one of the following options:

D-5.a) **MONO_DIR** $\Omega_x$ $\Omega_y$ $\Omega_z$

D-5.b) **ISOTROPIC**

D-5.c) **INTERPOLATED** nbpoints nbinter
<**MODE_INTERPOLATION**>(1) numlim sup(1)
....
<**MODE_INTERPOLATION**>(nbinter) numlim sup(nbinter)
$\mu$(0) value(0)
....
$\mu$(nbpoints-1) value(nbpoints-1)

D-5.d) **ANALYTICAL**
**FUNCTION**=...;
**DOMAIN**=...;

D-6) **ENERGETIC_DISTRIBUTION SPECTRUM** Choose one of the following options:

D-6.a) **MONOKINETIC** E0

D-6.b) **BAND** Esup Einf

D-6.c) **WATT_SPECTRUM** E0
**PARAM** Esup Einf a b (optional)

D-6.d) **MAXWELL_SPECTRUM** $\theta$ Esup Einf

D-6.e) **INTERPOLATED** nbpoints nbinter
<**MODE_INTERPOLATION**>(1) numlim sup(1)
....
<**MODE_INTERPOLATION**>(nbinter) numlim sup(nbinter)
E(0) value(0)
....
E(nbpoints-1) value(nbpoints-1)

D-6.f) **ANALYTICAL**
**FUNCTION**=...;
**DOMAIN**=...;

D-6.g) **RAY_SPECTRUM** nblines
E(0) I(0)

```
...
E(nblines-1) I(nblines-1)
```

D-7) **TIME_DISTRIBUTION** Choose one of the following options:

D-7.a) **DIRAC** `t0`

D-7.b) **UNIFORM** `tinf tsup`

D-7.c) **INTERPOLATED** `nbpoints nbinter`
<**MODE_INTERPOLATION**>>`(1) numlim sup(1)`
`....`
<**MODE_INTERPOLATION**>>`(nbinter) numlim sup(nbinter)`
`t(0) value(0)`
`....`
`t(nbpoints-1) value(nbpoints-1)`

D-7.d) **ANALYTICAL**
**FUNCTION**=...;
**DOMAIN**=...;

D-8) **EXTERNAL**

D-8.a) `source.so` file name of the source function dynamically linked

D-8.b) **NO_GEOMSOURCE_WARNING** optional, if present, doesn't stop calculation if sampling efficiency too low (< 10%)

D-8.c) **SOBOL_POINTS** `nbsampling` optional, nb of sampling for testing the source function (default=1000)

D-8.d) **VOLUME** `nbvolumes volume1 ... volumen` optional

D-8.e) **PARAM** `nbparam param1 ... paramn` optional, numerical parameters transmitted to source function

D-9) **END_SOURCES**

Go back to D until all the sources are defined, then go to E.

E) **END_SOURCES_LIST**

A) **RESPONSES**

B) `nbresponses`

C) Choose one of the following options and return to C to define another response. Go to D to end.

C-1) **FLUX** `particle_type` (**NEUTRON** or **PHOTON**)

C-2) **CURRENT** `particle_type` (**NEUTRON** or **PHOTON** or **ELECTRON** or **POSITRON**)

C-3) **REACTION** `particle_type` (**NEUTRON** or **PHOTON**)

C-4) **DEPOSITED_ENERGY** `particle_type` (**NEUTRON** or **PHOTON** or **ELECTRON** or **POSITRON**)

C-5) **DEPOSITED_SPECTRUM** `particle_type` (**PHOTON** or **PHOTON ELECTRON POSITRON**)

C-6) **DEPOSITED_SPECTRUM_WEIGHTED_ENERGY** `particle_type` (**PHOTON** or **PHOTON ELECTRON POSITRON**) If there is a single nucleus, go to C-3.a, if there are many go to C-3.b.

C-3.a) **NUCLEUS**, go to C-3.c

C-3.b) **NUCLEI** `nbnuclei`, go to C-3.c

C-3.c) `name_of_nucleus`

- If the nucleus belongs to a composition, go to C-3.d
- If the nucleus does not belong to a composition, but is in the library, go to C-3.e
- If the nucleus is to be described, go to C-3.f

In the **NUCLEI** case, C-3.c is done once for each nucleus.

C-3.d) **COMPOSITION** `composition_name`
**INTERACTION** `nbinteractions`
`interaction1 ...interactionn`

C-3.e) **FICTIVE** `temperature` (integer)
**INTERACTION** `nbinteractions`
`interaction1 ...interactionn`

C-3.f) (**COEFF** `coeff`) **DATA**
`nbpoints nbinter` (number of points, interpolation intervals)
<**MODE_INTERPOLATION**>`(1) upper_limit(1)`
`....`
<**MODE_INTERPOLATION**>`(nbinter) upper_limit(nbinter)`
`Energy(0) cross-section(0)`
`....`
`Energy(nbpoints-1) cross-section(nbpoints-)1`

D) **END_RESPONSES**

A) **GRID_LIST**

B) nbgrids

C) name_of_grid
nbbounds
bounds1 ...
boundsi **IN_LIN** p boundsi+p+1 ...
boundsj **IN_LOG** q boundsj+q+1 ...
boundsn (list of bounds)
Return to C for a new grid; otherwise, go to D to end.

D) **END_GRID_LIST**

A) **SCORE**

B) nbscores

C) Define a score
associated_response_number
Choose one of the following options (track length estimator: **TRACK**, collision estimator: **COLLISION**, surface-based estimator: **SURF** or point estimator: **FLUX_PT**) and return to C to describe another score. Go to D to end.

C-1) **TRACK** associated_energy_grid_name If a time grid is required (angular) go to C-1.a (C-1.b); otherwise, go directly to B-1.b.

C-1.a) **TIME GRID** associated_time_grid_name
Go to C-1.b.

C-1.b) **ANGULAR GRID** associated_angle_grid_name
Go to C-1.c.

C-1.c) **GRID** associated_energy_grid_name
Go to C-1.c.2 by default, or C-1.c.1 if specific properties are required.

B-1.c.1) **COMPLETE** or **BOOTSTRAP**
**COMPLETE** enables all results to be preserved (by default the average value is preserved), and **BOOTSTRAP** enables calculation of the confidence interval. Go to C-1.c.2.

C-1.c.2) For a list of volumes (C-1.c.2.1), for a sum of volumes (C-1.c.2.2), for a list of cells (C-1.c.2.3), for a sum of cells (C-1.c.2.4).

C-1.c.2.1) **VOLUME LIST** nbvolu
volume1...volumen

C-1.c.2.2) **VOLUME SUM** nbvolu
volume1...volumen

C-1.c.2.1) **CELL LIST** nbcells
numvol1 numlattice1 cell1(i) cell1(j) cell1(k)...
numvoln numlatticen celln(i) celln(j) celln(k)

C-1.c.2.2) **CELL SUM** nbcells
numvol1 numlattice1 cell1(i) cell1(j) cell1(k)...
numvoln numlatticen celln(i) celln(j) celln(k)

C-2) **COLLISION** associated_energy_grid_name
If a time (and/or angle) grid is required, go to C-2.a; otherwise, go directly to C-2.b.

C-2.a) **TIME GRID** associated_time_grid_name
and/or **ANGULAR GRID** associated_angle_grid_name
Go to C-2.b.

C-2.b) **GRID** associated_energy_grid_name
For a list of volumes (C-2.b.1), for a sum of volumes (C-2.b.2), for a list of cells (C-2.c.3), for a sum of cells (C-2.c.4)

C-2.b.1) **VOLUME LIST** nbvolumes
volume1...volumen

C-2.b.2) **VOLUME SUM** nbvolu
volume1...volumen

C-2.c.3) **CELL LIST** nbcells
numvol1 numlattice1 cell1(i) cell1(j) cell1(k)...
numvoln numlatticen celln(i) celln(j) celln(k)

C-2.c.4) **CELL SUM** nbcells
numvol1 numlattice1 cell1(i) cell1(j) cell1(k)...
numvoln numlatticen celln(i) celln(j) celln(k)

C-3) **SURF** associated_energy_grid_name
If time (and/or angle) grid is required, go to B-3.a; otherwise, go directly to B-3.b.

C-3.a) **TIME GRID** associated_time_grid_name and/or **ANGULAR GRID** associated_angle_grid_name
Go to B-3.b.

C-3.b) **GRID** associated_energy_grid_name
For a list go to C-3.b.1, for a sum C-3.b.2 (surface i is defined as the boundary between volumei1 and volumei2 where volumei2=-1 for an escape surface)

C-3.b.1) **FRONTIER LIST** nbsurfaces
volume11 volume12
...
volumen1 volumen2

C-3.b.2) **FRONTIER SUM** nbsurfaces
volume11 volume12
...
volumen1 volumen2
(surface i is defined as the boundary between volumei1 and volumei2 where volumei2=-1 for an escape surface)

C-4) **FLUX_PT** associated_energy_grid_name
If a time (and/or angle) grid is required, go to B-4.a; otherwise, go directly to B-4.b.

C-4.a) **TIME GRID** associated_time_grid_name
and/or **ANGULAR GRID** associated_angle_grid_name
Go to C-4.b.

C-4.b) **GRID** associated_energy_grid_name
**POINT LIST** nbpoints
x1 y1 z1
...
xn yn zn

D) **END_SCORE**

A) **VARIANCE_REDUCTION** (to be performed twice for a combined neutron-gamma problem)

B) **NEUTRON** or **PHOTON**

C) **GRID** nbbounds
energy1 ... energyn

D) For a point detector, go to D-1, for a surface-based detector, go to D-2.

D-1) **DETECT** nbdetectors
D1x D1y D1z (coordinates of detector)
$\beta_1$ (associated coefficient)
...
Dnx Dny Dnz
$\beta_n$
If mesh, go to E.

D-2) **DETECT_RHO** 1 $\rho(D)$ $\beta_D$

E) **MESH**

E-1) Nx' Ny' Nz' (number of cells)
dx' dy' dz' (sizes of cells)
3

E-2) **FRAME** (associated with mesh)

E-2.a) **CARTESIAN**
O'x O'y O'z
ix iy iz
jx jy jz
kx ky kz

E-2.b) **CYLINDER**
O'x O'y O'z
ix iy iz
kx ky kz

In order to refine the mesh by 8 in a zone, go to E-3; otherwise, go to E-4.

E-3) **DIVISION WINDOW**
X'min Y'min Z'min
X'max Y'max Z'max (description of window)

E-4) **END_MESH**

F) **INIPOND**
For automatic biasing, go to F-1, for manual biasing, go to F-2.

F-1) **AUTO**, go to F-3.

F-2) **DATA** `nomcompo(1)`
`k(1) ... k(nbgroups)`
`...`
`nomcompo(n)`
`k(1) ... k(nbgroups)`

F-3) **OMEGA**
Case of surface-based detector biasing. If the direction is calculated by gradient, go to F-3.1; if the direction is set by the user, go to F-3.2.

F-3.1) **GRAD_I**

F-3.2) **SET_OMEGA**
**CYLINDER**
$u_x$ $u_y$ $u_z$
$c_x$ $c_y$ $c_z$

Go to F-4.

F-4) **END_INIPOND**

G) Optional case of energy biasing in $\exp(-\alpha(i)E)$:
**ALFA** $\alpha(1)\ldots\alpha(n)$.
There is one value for each energy group.

H) Optional case of time biasing in $exp(-\lambda t)$:
**TIME** $\lambda$

I) **GRAPH**
`Ox Oy Oz` (origin of the 2D-view in geometry reference frame)
`Lgraf(1) Lgraf(2)` (size of window)
`X` or `Y` or `Z` (direction normal to the 2D-view in mesh reference frame)

J) **END_VARIANCE_REDUCTION**

A) **HOMOGENIZE**

B) **NBMAT** nbcompo

B-1) nomcompo
**VOLUME LIST** nbvolu
volu1 ... volun
name(grid)
Continue to B-1 until all the compositions have been defined.

B-2) **FILE** name(cross-sections_file)

C) **DIFFUSION** 1

D) **END_HOMOGENIZE**

A) **SIMULATION**

B) Mandatory data
**BATCH** nbbatches
**SIZE** number_of_particles_per_batch
**PARTICLE** number_of_types_of_particles
If there are neutrons, go to B-1; otherwise, go directly to B-2.

B-1) **NEUTRON**
If there are photons go to B-2; otherwise, go directly to C.

B-2) **PHOTON**
If there are also electrons and positrons go to B-3; otherwise, go directly to C.

B-3) **ELECTRON POSITRON**

C) Optional data
**ENERGY_INF NEUTRON** Einf(neutron)
**ENERGY_SUP NEUTRON** Esup(neutron)
**ENERGY_INF PHOTON** Einf(photon)
**ENERGY_SUP PHOTON** Esup(photon)
**ENERGY_INF ELECTRON** Einf(electron)
**ENERGY_INF POSITRON** Einf(positron)
**ENERGY_SUP ELECTRON** Esup(electron)
**ENERGY_SUP POSITRON** Esup(positron)
**EDITION** frequency
For a criticality problem, go to C-1; for a fixed source sub-criticality problem, go to C-2. For a shielding problem, leave blank, go to D.

C-1) ) **CRITICALITY**
If it is wished to delete batches
**DISCARD** number_of_batches_to_be_discarded

C-2) **FIXED_SOURCES_CRITICALITY**

D) Other optional data
**TRACK_INTEGRATION** 1 in order to use the track integration option
**MONITORING** 0 in order to eliminate monitoring
**COLLISION** 1 Einf(coll) in order to bias collision above Einf(coll)
**KEEP_RESULT** in order to keep the results to continue the calculation
**RANDOM** in order to give an initial random number
**XML_EXPORT** in order to write results in xml file
**XML_NOCOMPLETE** in order to reduce xml output to EDITION

D) Optional data concerning electromagnetic shower
**ELECTRON_INELASTIC_CUTOFF** energy
**ELECTRON_BREMS_CUTOFF** energy
**NO_ELECTRON_MULTIPLE_SCATTERING**
**ELECTRON_TOTAL_STOPPING_POWER**

F) **END_SIMULATION**

A) **CALCULATION** If a volume calculation is required, go to B; otherwise, go to C.

B) Calculation of volumes
(**SET**) **VOLUME** (**SET**is used to initialize the volumes for the value calculated)
**FRAME**
Choice of coordinated system (U,V,W) associated with origin O′ and vectors i, j and k.

B-1) **CARTESIAN**
O′x O′y O′z
ix iy iz
jx jy jz
kx ky kz

B-2) **CYLINDER**
O′x O′y O′z
ix iy iz
kx ky kz

B-3) **SPHERE**
O′x O′y O′z
ix iy iz
kx ky kz

**WINDOW**
Umin, Vmin, Wmin
Umax, Vmax, Wmax
**PRECISION** precision
**END_VOLUME**
If a density calculation is required go to C; otherwise, go to D.

C) Calculation of densities
**DENSITY**
If the densities of all the compositions are required go to C-1; otherwise, go to C-2.

C-1) **ALL** Go to C-3.

C-2) **LIST** nbcompo name_composition1...name_compositionn Go to C-3.

C-3) ) **END_DENSITY**

D) **END_CALCULATION**

A) **PEPIN**

B) **GRID** name(grid)
In order to rebuild the materials with the natural composition, go to E.

C) **ISOTOPES** (to not rebuild the materials)

D) **SCORE_MODE** mode (with mode equating to **TRACK** or **COLLISION**)

E) **HISTORY** nb(steps)
time(init) <UNIT>
time(1) <UNIT> value(1)
...
time(final) <UNIT> value(final)

F) **VOLUME** nbvolu
volu1...volun

G) **END_PEPIN**

A) **STORAGE**

B) `nb(instances_of_storage)`

C) `name(file)`
**NEUTRON** or **PHOTON**
`Emax Emin`
**FRONTIER** `volu1 volu2`
Do C once for each request, then go to D.

D) **END_STORAGE**

A) **PERTURBATION_LIST**

B) nb(perturbations)

C) **PERTURBATION**
Go to C-1) for a concentration perturbation, go to C-2 for a density perturbation (only one factor must be given in C-3).

C-1) **CONCENTRATION**. Go to C-3.

C-2) **DENSITY**. Go to C-3.

C-3) nb(compositions)
**VOLUME LIST** nb(volu) volu(1)...volu(N)
**FACT** fact(1)...fact(N)
**SCORE LIST** nb(scores) numscore(1)...numscore(n)
Go to C-5 for a differential mode; otherwise, go directly to C-6.

C-4) **DETERMINISTIC NUCLEI** nb(nuclei)

C-4.a) number of composition of nucleus1 name of composition
**VOLUME LIST** nb_volu
numvol1 ... numvoln
**NUCLEUS** nucleus name **DATA** filename

Do C-4.a once for every disturbed nucleus

C-5) **MODE DELTA**. Go to C-6.

C-6) **END_PERTURBATION**

Do C once for every perturbation request, then go to D.

F) **END_PERTURBATION_LIST**

A) **OUTPUT**

B) **LANGUAGE** (optional, English by default).

C) For XVGR outputs go to C-1).

C-1) **XVGR**
**ENERGY_INF** Einf
**ENERGY_SUP** Esup
**TYPE** interpolation_type (where interpolation_type equates to LIN_LIN, LIN_LOG, LOG_LIN, LOG_LOG).
**END_XVGR**

F) **END_OUTPUT**

A) **VOLENERCUT**

B) type_of_particle

C) **ALL** `default_energy_cut` set a default energy threshold for all volumes (optional)

D) `numvol_i energy_cut_i` set energy_cut_i to numvol_i.
Do D for each volume requested.
Go to B for every particle requested.

F) **END_VOLENERCUT**

# Appendix C

# TRANSLATION KEYWORDS

Note: The keywords in the different languages are not listed in strict alphabetical order. The first keywords relate to geometry, followed by the general commands and finally the keywords for the PEPIN option.

| English (english) | French (french) | Default (old) |
|---|---|---|
| GEOMETRY: | | |
| BASIS | BASE | BASE |
| BLACK | NOIR | BLACK |
| BLUE | BLEU | BLUE |
| BOX | BOITE | BOITE |
| BROWN | BRUN | BROWN |
| CELL | MAILLE | MAILLE |
| COLOR | COULEUR | COLOR |
| COMBI | COMBI | COMBI |
| COMMENT | COMMENT | COMMENT |
| CONE | CONE | CONE |
| CONEX | CONEX | CONEX |
| CONEY | CONEY | CONEY |
| CONEZ | CONEZ | CONEZ |
| CORAL | CORAL | CORAL |
| CYAN | CYAN | CYAN |
| CYL | CYL | CYL |
| CYLX | CYLX | CYLX |
| CYLY | CYLY | CYLY |
| CYLZ | CYLZ | CYLZ |
| DEPTH | PROF | PROF |
| ENDGEOM | FINGEOM | FINGEOM |
| ENDV | FINV | FINV |
| EQUA | EQUA | EQUA |
| EXCEPT | EXCEPT | EXCEPT |
| FICTIVE | FICTIF | FICTIF |
| GOLD | OR | GOLD |
| GRAPH | GRAPHE | GRAF |
| GRAY | GRIS | GRAY |
| GREEN | VERT | GREEN |
| GREY | GREY | GREY |
| HEXAX | HEXAX | HEXAX |
| HEXAY | HEXAY | HEXAY |
| HEXAZ | HEXAZ | HEXAZ |
| INTE | INTE | INTE |
| KEEP | GARDE | GARDE |
| LATTICE | RESEAU | RESEAU |

| English (english) | French (french) | Default (old) |
|---|---|---|
| LOWER_CELL | SOUS_MAILLE | SOUS_MAILLE |
| MAGENTA | MAGENTA | MAGENTA |
| MATCH | MATCH | MATCH |
| MINUS | MOINS | MOINS |
| M_ORIGIN | M_ORIGIN | M_ORIGIN |
| NAVY | NAVY | NAVY |
| ORANGE | ORANGE | ORANGE |
| PARALLELEPIPEDE | PARALLELEPIPEDE | PARALLELEPIPEDE |
| PINK | ROSE | PINK |
| PLANE | PLAN | PLAN |
| PLANEX | PLANX | PLANX |
| PLANEY | PLANY | PLANY |
| PLANEZ | PLANZ | PLANZ |
| PLUM | PLUM | PLUM |
| PLUS | PLUS | PLUS |
| POSITION | POSITION | POSITION |
| QUAD | QUAD | QUAD |
| RED | ROUGE | RED |
| REPLACE | REMPLACE | REMPLACE |
| RESC | RESC | RESC |
| RESH | RESH | RESH |
| ROTATION | ROTATION | ROTATION |
| SALMON | SAUMON | SALMON |
| SMASH | ECRASE | ECRASE |
| SPHE | SPHE | SPHE |
| SURF | SURF | SURF |
| TITLE | TITRE | TITRE |
| TORUS | TORE | TORE |
| TORUSX | TOREX | TORUSX |
| TORUSY | TOREY | TORUSY |
| TORUSZ | TOREZ | TORUSZ |
| TYPE | TYPE | TYPE |
| UNION | UNION | UNION |
| VIOLET | VIOLET | VIOLET |
| VMINUS | VMOINS | VMOINS |
| VOLU | VOLU | VOLU |
| VOL_EXCEPT | VOL_EXCEPT | VOL_EXCEPT |
| V_ORIGIN | V_ORIGIN | V_ORIGIN |
| WHITE | BLANC | WHITE |
| YELLOW | JAUNE | YELLOW |
| TRIPOLI-4 GENERAL COMMANDS : | | |
| 1D_SCAN | BALAYAGE_1D | 1D_SCAN |
| 2D_ANGULAR | ANGULAIRE_2D | 2D_ANGULAR |
| 2D_GRID | 2D_DECOUPAGE | 2D_DECOUPAGE |
| 2D_TABULATED | 2D_TABULE | 2D_TABULATED |
| ACTIVATION | ACTIVATION | ACTIVATION |
| ACTIVITY_THRESHOLD | ACTIVITE_SEUIL | ACTIVITY_THRESHOLD |
| ABORT_ON_LEAKAGE | ARRET_SUR_FUITES | ABORT_ON_LEAKAGE |
| ABSORBED_ENERGY | ENERGIE_ABSORBEE | ABSORBED_ENERGY |
| ABSORPTION | ABSORPTION | ABSORPTION |
| ADJOINT | ADJOINT | ADJOINT |
| ADJOINT_CRITICALITY | CRITICITE_ADJOINTE | ADJOINT_CRITICALITY |
| ADJUSTABLE_BORON | BORE_AJUSTABLE | ADJUSTABLE_BORON |
| AFTER_COLLISIONS | APRES_COLLISIONS | AFTER_COLLISIONS |
| AFTER_COLLISIONS_AND_AT_SURFACES | APRES_COLLISIONS_ET_AUX_SURFACES | AFTER_COLLISIONS_AND_AT_SURFACES |
| ALBEDO | ALBEDO | ALBEDO |
| ALBEDO_SEARCH | RECHERCHE_ALBEDO | ALBEDO_SEARCH |
| ALFA | ALFA | ALFA |
| ALL | TOUS | ALL |

| English (english) | French (french) | Default (old) |
|---|---|---|
| ALL | TOUS | TOUS |
| ALL_COMPLETE | TOUS_COMPLET | ALL_COMPLETE |
| ALL_COLLIDED_FLUX_PT_POINTS | COLLISIONS_POUR_TOUT_FLUX_PT_POINTS | ALL_COLLIDED_FLUX_PT_POINTS |
| ALL_FISSILE_VOLUMES_INSIDE_MESH | TOUS_VOLUMES_FISSILES_DANS_MAILLAGE | ALL_FISSILE_VOLUMES_INSIDE_MESH |
| ALL_PARTICLE | TOUTES_PARTICULES | ALL_PARTICLE |
| ALL_VOLUMES | TOUSVOLUMES | ALL_VOLUMES |
| ALL_VOLUMES_INSIDE_MESH | TOUS_VOLUMES_DANS_MAILLAGE | ALL_VOLUMES_INSIDE_MESH |
| ALPHA | ALPHA | ALPHA |
| AMS | AMS | AMS |
| ANALOG | ANALOGUE | ANALOG |
| ANALYTICAL | ANALYTIQUE | ANALYTICAL |
| ANGULAR | ANGULAIRE | ANGULAR |
| ANGULAR_DISTRIBUTION | DISTRIBUTION_ANGULAIRE | ANGULAR_DISTRIBUTION |
| APOLLO2 | APOLLO2 | APOLLO2 |
| ASCII | ASCII | ASCII |
| ASCIIG | ASCIIG | ASCIIG |
| ATOMIC_WEIGHT_WINDOW | FENETRE_MASSE_ATOMIQUE | ATOMIC_WEIGHT_WINDOW |
| AUTO | AUTO | AUTO |
| AUTO_COUPLING | AUTO_COUPLAGE | AUTO_COUPLING |
| BAND | BANDE | BANDE |
| BATCH | BATCH | BATCH |
| BEFORE_AND_AFTER_COLLISIONS | AVANT_ET_APRES_COLLISIONS | BEFORE_AND_AFTER_COLLISIONS |
| BEGIN_OF_BATCH | BEGIN_OF_BATCH | BEGIN_OF_BATCH |
| BIBLIO | BIBLIO | BIBLIO |
| BOLTZMANN | BOLTZMANN | BOLTZMANN |
| BOOTSTRAP | BOOTSTRAP | BOOTSTRAP |
| BOUNDARY_CONDITION | LIMITE | LIMIT |
| BREMSSTRAHLUNG_MODEL | MODELE_BREMSSTRAHLUNG | BREMSSTRAHLUNG_MODEL |
| CALCULATION | CALCUL | CALCUL |
| CAPTURE | CAPTURE | CAPTURE |
| CARTESIAN | CARTESIEN | CARTESIAN |
| CELL | MAILLE | MAILLE |
| CENTRAL_SPLITTING | CENTRAL_SPLITTING | CENTRAL_SPLITTING |
| CHECK_GEO | VERIF_GEO | CHECK_GEO |
| CHECK_HOLES | VERIF_TROUS | CHECK_HOLES |
| CHECK_TRACKING | VERIF_POURSUITE | CHECK_TRACKING |
| CHI_FISSION | CHI_FISSION | CHI_FISSION |
| COEFF | COEFF | COEFF |
| COEFF_NB_ISOTOPES | COEFF_NB_ISOTOPES | COEFF_NB_ISOTOPES |
| COLLIDED | DIFFUSES | COLLIDED |
| COLLISION | COLLISION | COLLISION |
| COLLISION_SITES | LIEUX_COLLISION | COLLISION_LOCUS |
| COLLISION_TAPE | BANDE_COLLISION | BANDE_COLLISION |
| COMMENT | COMMENTAIRE | COMMENT |
| COMPLETE | COMPLET | COMPLETE |
| COMPOSITION | COMPOSITION | COMPOSITION |
| CONCENTRATION | CONCENTRATION | CONCENTRATION |
| CONDUCTOR | CONDUCTEUR | CONDUCTOR |
| CONTROL_RUSSIAN_ROULETTE | CONTROLE_ROULETTE_RUSSE | CONTROL_RUSSIAN_ROULETTE |
| COSINUS | COSINUS | COSINUS |
| COVARIANCE | COVARIANCE | COVARIANCE |
| CPP_ROOT_GEOM | CPP_ROOT_GEOM | CPP_ROOT_GEOM |
| CREATION | CREATION | CREATION |
| CRITICAL_BORON_SEARCH | RECHERCHE_BORE_CRITIQUE | CRITICAL_BORON_SEARCH |
| CRITICALITY | CRITICITE | CRITIC |
| CROSS_SPLIT | SPLIT_FRONTIERE | CROSS_SPLIT |
| CS_RATIO | CS_RATIO | CS_RATIO |
| CU_RATIO | CU_RATIO | CU_RATIO |
| CURRENT | COURANT | COURANT |
| CYLINDER | CYLINDRE | CYLINDER |

| English (english) | French (french) | Default (old) |
|---|---|---|
| DATA | DATA | DATA |
| DBRC | DBRC | DBRC |
| DELAYED | RETARDE | DELAYED |
| DELAYED_NUSIGF | NUSIGF_RETARDE | DELAYED_NUSIGF |
| DELTA | DELTA | DELTA |
| DENSITY | DENSITE | DENSITY |
| DEPLETED_COMPOSITIONS | COMPOSITIONS_EVOLUANTES | DEPLETED_COMPOSITIONS |
| DEPOSITED_CHARGE | CHARGE_DEPOSEE | DEPOSITED_CHARGE |
| DEPOSITED_ENERGY | ENERGIE_DEPOSEE | DEPOSITED_ENERGY |
| DEPOSITED_SPECTRUM | SPECTRE_DEPOSE | DEPOSITED_SPECTRUM |
| DEPOSITED_SPECTRUM_WEIGHTED_ENERGY | SPECTRE_DEPOSE_PONDERE_PAR_ENERGIE | DEPOSITED_SPECTRUM_WEIGHTED_ENERGY |
| DETECTOR | DETECTEUR | DETECT |
| DETECT_RHO | DETECT_RHO | DETECT_RHO |
| DETERMINISTIC | DETERMINISTE | DETERMINISTE |
| DEUTERON | DEUTERON | DEUTERON |
| DIRAC | DIRAC | DIRAC |
| DISCARD | SUPPRIME | DISCARD |
| DISCRETE_MU | MU_DISCRET | DISCRETE_MU |
| DISCRETE_PHI | PHI_DISCRET | DISCRETE_PHI |
| DISCRETE_PROBABILITY | PROBABILITE_DISCRETE | DISCRETE_PROBABILITY |
| DISPLACEMENT_ENERGY | ENERGIE_DEPLACEMENT | DISPLACEMENT_ENERGY |
| DIVEN_DISPERSION_FACTOR | FACTEUR_DE_DISPERSION_DE_DIVEN | DIVEN_DISPERSION_FACTOR |
| DPA | DPA | DPA |
| DYNAMIC | DYNAMIQUE | DYNAMIC |
| EDITION | EDITION | EDITION |
| ELECTRON | ELECTRON | ELECTRON |
| ELECTRON_BREMS_CUTOFF | ELECTRON_COUPURE_BREMS | ELECTRON_COUPURE_BREMS |
| ELECTRON_INELASTIC_CUTOFF | ELECTRON_COUPURE_INELASTIQUE | ELECTRON_COUPURE_INELASTIQUE |
| ELECTRON_MIN_STEP_LENGTH | ELECTRON_LONGUEUR_MIN_PAS | ELECTRON_MIN_STEP_LENGTH |
| ELECTRON_PHOTON_BALANCE | BILAN_ELECTRON_PHOTON | ELECTRON_PHOTON_BALANCE |
| ELECTRON_REPARTITION | ELECTRON_REPARTITION | ELECTRON_REPARTITION |
| ELECTRON_TOTAL_STOPPING_POWER | ELECTRON_POWER_ARRET_TOTAL | ELECTRON_TOTAL_STOPPING_POWER |
| ELEMENT | ELEMENT | ELEMENT |
| END | FIN | FIN |
| END_1D_SCAN | FIN_BALAYAGE_1D | END_1D_SCAN |
| END_AMS | FIN_AMS | FIN_AMS |
| END_AUTO_COUPLING | FIN_AUTO_COUPLING | FIN_AUTO_COUPLING |
| END_BOUNDARY_CONDITION | FIN_LIMITE | FIN_LIMIT |
| END_CALCULATION | FIN_CALCUL | FIN_CALCULATION |
| END_CELL | FIN_MAILLE | FIN_MAILLE |
| END_CHECK_GEO | FIN_VERIF_GEO | FIN_CHECK_GEO |
| END_CHECK_TRACKING | FIN_VERIF_POURSUITE | FIN_CHECK_TRACKING |
| END_COLLISION_TAPE | FIN_BANDE_COLLISION | FIN_BANDE_COLLISION |
| END_COMPOSITION | FIN_COMPOSITION | FIN_COMPOSITION |
| END_CREATION | FIN_CREATION | FIN_CREATION |
| END_DENSITY | FIN_DENSITE | FIN_DENSITY |
| END_DISPLACEMENT_ENERGY | FIN_ENERGIE_DEPLACEMENT | END_DISPLACEMENT_ENERGY |
| END_ELEMENT | FIN_ELEMENT | FIN_ELEMENT |
| END_EXPLOITATION | FIN_EXPLOITATION | FIN_EXPLOITATION |
| END_G4_GEOMETRY | FIN_G4_GEOMETRIE | FIN_G4_GEOMETRY |
| END_G4_INTERFACE | FIN_G4_INTERFACE | FIN_G4_INTERFACE |
| END_G4_UI_COMMANDS | FIN_G4_COMMANDES_UI | FIN_G4_UI_COMMANDS |
| END_GEOMCOMP | FIN_GEOMCOMP | FIN_GEOMCOMP |
| END_GREEN_FUNCTION_TAPE | FIN_BANDE_FONCTION_GREEN | FIN_GREEN_FUNCTION_BAND |
| END_GRID_LIST | FIN_LISTE_DECOUPAGE | FIN_LIST_DECOUPAGE |
| END_HOMOGENIZE | FIN_HOMOGENEISATION | FIN_HOMOGENIZE |
| END_IDLE | FIN_IDLE | FIN_IDLE |
| END_IFP_SCORES | FIN_IFP_SCORES | FIN_IFP_SCORES |
| END_IFP_PERTURBATION_LIST | FIN_LISTE_PERTURBATIONS_IFP | FIN_IFP_PERTURBATION_LIST |
| END_IFP_SENSITIVITY_LIST | FIN_LISTE_SENSIBILITES_IFP | FIN_IFP_SENSITIVITY_LIST |

| English (english) | French (french) | Default (old) |
|---|---|---|
| END_IMPORTANCE | FIN_IMPORTANCE | FIN_IMPORTANCE |
| END_INIPOND | FIN_INIPOND | FIN_INIPOND |
| END_MASS | FIN_MASSE | FIN_MASS |
| END_MATRIX | FIN_MATRICE | FIN_MATRIX |
| END_LATEX | FIN_LATEX | FIN_LATEX |
| END_MESH | FIN_MAILLAGE | FIN_MAILLAGE |
| END_NUCLEI | FIN_NOYAUX | FIN_NUCLEI |
| END_MESH_VARIABLE | FIN_MAILLAGE_VARIABLE | FIN_MAILLAGE_VARIABLE |
| END_OF_BATCH | END_OF_BATCH | END_OF_BATCH |
| END_OPTIMIZE | FIN_OPTIMISATION | FIN_OPTIMIZE |
| END_OUTPUT | FIN_OUTPUT | FIN_OUTPUT |
| END_PATH | FIN_PARCOURS | END_PATH |
| END_PERTURBATION | FIN_PERTURBATION | FIN_PERTURBATION |
| END_PERTURBATION_LIST | FIN_LISTE_PERTURBATION | FIN_LIST_PERTURBATION |
| END_PREINIT | FIN_PREINIT | FIN_PREINIT |
| END_REACTION_RATES_SPHERICAL_HARMONICS | FIN_REACTION_RATES_SPHERICAL_HARMONICS | FIN_REACTION_RATES_SPHERICAL_HARMONICS |
| END_RESPONSES | FIN_REPONSES | FIN_REPONSES |
| END_ROOT_GEOMETRY | FIN_ROOT_GEOMETRIE | FIN_ROOT_GEOMETRY |
| END_SCAN_GEOMCOMP | FIN_BALAYAGE_GEOMCOMP | FIN_SCAN_GEOMCOMP |
| END_SCAN_GEOM_1D | FIN_BALAYAGE_GEOM_1D | FIN_SCAN_GEOM_1D |
| END_SCORE | FIN_SCORE | FIN_SCORE |
| END_SECTION | FIN_SECTION | FIN_SECTION |
| END_SIMULATION | FIN_SIMULATION | FIN_SIMULATION |
| END_SOURCES | FIN_SOURCES | FIN_SOURCES |
| END_SOURCES_LIST | FIN_LISTE_SOURCES | FIN_LIST_SOURCES |
| END_STORAGE | FIN_STOCKAGE | FIN_STOCKAGE |
| END_SUPER_HISTORY_PERTURBATION_LIST | FIN_LISTE_PERTURBATIONS_SUPER_HISTORY | FIN_SUPER_HISTORY_PERTURBATION_LIST |
| END_SUPER_HISTORY_SENSITIVITY_LIST | FIN_LISTE_SENSIBILITES_SUPER_HISTORY | FIN_SUPER_HISTORY_SENSITIVITY_LIST |
| END_TAYLOR_PERTURBATION | FIN_TAYLOR_PERTURBATION | FIN_TAYLOR_PERTURBATION |
| END_TRACKS | FIN_TRACKS | FIN_TRACKS |
| END_UNITS | FIN_UNITES | FIN_UNITS |
| END_VARIANCE_REDUCTION | FIN_PONDERATION | FIN_PONDERATION |
| END_VOLENERCUT | FIN_COUPURE_ENER_PAR_VOL | FIN_VOLENERCUT |
| END_VOLSURF | FIN_VOLSURF | FIN_VOLSURF |
| END_VOLUME | FIN_VOLUME | FIN_VOLUME |
| END_WIELANDT_PERTURBATION_LIST | FIN_LISTE_PERTURBATIONS_WIELANDT | FIN_WIELANDT_PERTURBATION_LIST |
| END_WIELANDT_SENSITIVITY_LIST | FIN_LISTE_SENSIBILITES_WIELANDT | FIN_WIELANDT_SENSITIVITY_LIST |
| END_XVGR | FIN_XVGR | FIN_XVGR |
| ENERGETIC_DISTRIBUTION | DISTRIBUTION_ENERGETIQUE | ENERGETIC_DISTRIBUTION |
| ENERGY | ENERGIE | ENERGY |
| ENERGY_INF | ENERGIE_INF | ENERGY_INF |
| ENERGY_SUP | ENERGIE_SUP | ENERGY_SUP |
| ENERGY_THRESHOLD | ENERGIE_SEUIL | ENERGY_THRESHOLD |
| ENGLISH | ANGLAIS | ENGLISH |
| ENTROPY | ENTROPIE | ENTROPY |
| EPITHERMAL_BROADENING | ELARGISSEMENT_EPITHERMIQUE | EPITHERMAL_BROADENING |
| EV | EV | EV |
| EXCLUSION | EXCLUSION | EXCLUSION |
| EXPLOITATION | EXPLOITATION | EXPLOITATION |
| EXTENDED_MESH | MAILLAGE_ETENDU | EXTENDED_MESH |
| EXTERNAL | EXTERNE | EXTERNAL |
| FACE | FACE | FACE |
| FACT | FACT | FACT |
| FACTORIZED | FACTORISE | FACTORIZED |
| FAST | RAPIDE | FAST |
| FICTIVE | FICTIF | FICTIF |
| FIFRELIN | FIFRELIN | FIFRELIN |
| FILE | FICHIER | FILE |
| FILE_NAME | FILE_NAME | FILE_NAME |
| FILTER | FILTRE | FILTER |

| English (english) | French (french) | Default (old) |
|---|---|---|
| FILTER_AND | FILTRE_ET | FILTER_AND |
| FILTER_BY_COMPOSITION | FILTRE_PAR_COMPOSITION | FILTER_BY_COMPOSITION |
| FILTER_BY_ISOTOPE | FILTRE_PAR_ISOTOPE | FILTER_BY_ISOTOPE |
| FILTER_OR | FILTRE_OU | FILTER_OR |
| FILTER_SEQ | FILTRE_SEQ | FILTER_SEQ |
| FILTER_TARGET | FILTRE_CIBLE | FILTER_TARGET |
| FIRST_ORDER | PREMIER_ORDRE | FIRST_ORDER |
| FILTER_TARGET | FILTRE_CIBLE | FILTER_TARGET |
| FISSION_ENERGY_SUP | ENERGIE_SUP_FISSION | FISSION_ENERGY_SUP |
| FISSION_FROM_EVALUATION | FISSION_DE_L_EVALUATION | FISSION_FROM_EVALUATION |
| FISSION_MODEL | MODELE_DE_FISSION | FISSION_MODEL |
| FISSION_N_SUM_SPECTRUM | SPECTRE_SOMME_N_FISSION | FISSION_N_SUM_SPECTRUM |
| FISSION_SPECTRUM | SPECTRE_DE_FISSION | FISSION_SPECTRUM |
| FISSION_ZA_DISTRIBUTION | FISSION_ZA_DISTRIBUTION | FISSION_ZA_DISTRIBUTION |
| FIXED_SOURCES_CRITICALITY | SOURCE_FIXE_CRITICITE | FIXED_SOURCES_CRITICALITY |
| FIXED_SOURCE_EMISSION | EMISSION_SOURCE_FIXE | FIXED_SOURCE_EMISSION |
| FLUX | FLUX | FLUX |
| FLUX_PT | FLUX_PT | FLUX_PT |
| FORCED_UNIT_BASE_INTERPOL | INTERPOL_UNIT_BASE_FORCEE | FORCED_UNIT_BASE_INTERPOL |
| FORMAT | FORMAT | FORMAT |
| FRAME | REPERE | FRAME |
| FRAME | REPERE | REPERE |
| FRENCH | FRANCAIS | FRENCH |
| FREQUENCY | FREQUENCE | FREQUENCY |
| FROM | PAR | FROM |
| FROM_FISSION | DE_FISSION | FROM_FISSION |
| FRONTIER | FRONTIERE | FRONTIER |
| G4_GEOMETRY | G4_GEOMETRIE | G4_GEOMETRY |
| G4_INTERFACE | G4_INTERFACE | G4_INTERFACE |
| G4_SOURCE | G4_SOURCE | G4_SOURCE |
| G4_UI_COMMANDS | G4_COMMANDES_UI | G4_UI_COMMANDS |
| GASNTP | GAZTPN | GASNTP |
| GAS_PRODUCTION | PRODUCTION_GAZ | GAS_PRODUCTION |
| GAUSS_SPECTRUM | GAUSS_SPECTRE | GAUSS_SPECTRUM |
| GEOMETRIC_DISTRIBUTION | DISTRIBUTION_GEOMETRIQUE | GEOMETRIC_DISTRIBUTION |
| GEOMETRY | GEOMETRIE | GEOMETRIE |
| GLOBAL_SCORES | SCORES_GLOBAUX | GLOBAL_SCORES |
| GLOBAL_VALUE | VALEUR_GLOBALE | GLOBAL_VALUE |
| GPT_SUPER_HISTORY_SENSITIVITY_LIST | GPT_SUPER_HISTORY_SENSITIVITY_LIST | GPT_SUPER_HISTORY_SENSITIVITY_LIST) |
| GPT_NUMERATOR | GPT_NUMERATOR | GPT_NUMERATOR) |
| GPT_DENOMINATOR | GPT_DENOMINATOR | GPT_DENOMINATOR) |
| GRAD_I | GRAD_I | GRAD_I |
| GRAPH | GRAPHE | GRAF |
| GRAPH_CONTOURS | GRAPHE_CONTOURS | GRAF_CONTOURS |
| GRAPH_NO_CONTOUR | GRAPHE_SANS_CONTOURS | GRAF_NO_CONTOUR |
| GREEN_FUNCTION_TAPE | BANDE_FONCTION_GREEN | GREEN_FUNCTION_BAND |
| GRID | DECOUPAGE | DECOUPAGE |
| GRID_LIST | LISTE_DECOUPAGE | LIST_DECOUPAGE |
| GROUP_PROBABILITY | GROUPE_PROBABILITE | GROUP_PROBABILITY |
| HEIGHT | HAUTEUR | HEIGHT |
| HELIUM | HELIUM | HELIUM |
| HISTO | HISTO | HISTO |
| HOLE_COUNT | COMPTE_TROU | HOLE_COUNT |
| HOLE_MAP | CARTE_TROU | HOLE_MAP |
| HOLE_VOLU_MAP | CARTE_VOLU_TROU | HOLE_VOLU_MAP |
| HOMOGENEIZE | HOMOGENEISATION | HOMOGENIZE |
| IDLE | IDLE | IDLE |
| IFP_SCORES | IFP_SCORES | IFP_SCORES |
| IFP_ADJOINT_FLUX | IFP_ADJOINT_FLUX | IFP_ADJOINT_FLUX |
| IFP_ADJOINT_REACTION_RATE | IFP_ADJOINT_REACTION_RATE | IFP_ADJOINT_REACTION_RATE |

| English (english) | French (french) | Default (old) |
|---|---|---|
| IFP_CYCLE_LENGTH | LONGUEUR_CYCLE_IFP | IFP_CYCLE_LENGTH |
| IFP_KINETIC_PARAMETERS | PARAMETRES_CINETIQUES_IFP | IFP_KINETIC_PARAMETERS |
| IFP_CONVERGENCE_STATISTICS | STATISTIQUES_CONVERGENCE_IFP | IFP_CONVERGENCE_STATISTICS |
| IFP_PERTURBATION_LIST | LISTE_PERTURBATIONS_IFP | IFP_PERTURBATION_LIST |
| IFP_SENSITIVITY_LIST | LISTE_SENSIBILITES_IFP | IFP_SENSITIVITY_LIST |
| IMPORTANCE | IMPORTANCE | IMPORTANCE |
| INCIDENT_SPECTRUM | SPECTRE_INCIDENT | INCIDENT_SPECTRUM |
| INDUCED_BY_INTERACTION | INDUIT_PAR_INTERACTION | INDUCED_BY_INTERACTION |
| INELASTIC_THRESHOLD_CORRECTION | CORRECTION_SEUIL_INELASTIQUE | INELASTIC_THRESHOLD_CORRECTION |
| INFO | INFO | INFO |
| INIPOND | INIPOND | INIPOND |
| INITIAL_GUESS | VALEUR_INITIALE | INITIAL_GUESS |
| INTENSITY | INTENSITE | INTENSITY |
| INTERACTION | INTERACTION | INTERACTION |
| INTERPOLATED | INTERPOLE | INTERPOLATED |
| INTERPOLATION | INTERPOLATION | INTERPOLATION |
| ISOTROPIC | ISOTROPIQUE | ISOTROPIC |
| KEEP_RESULT | GARDE_RESULTAT | KEEP_RESULT |
| KIJ_MATRIX | KIJ_MATRICE | KIJ_MATRIX |
| KIJ_GREEN_MATRIX | KIJ_GREEN_MATRICE | KIJ_GREEN_MATRIX |
| KINCHIN_PEASE_MODEL | KINCHIN_PEASE_MODEL | KINCHIN_PEASE_MODEL |
| KINEMATIC_LIMITS | LIMITES_CINEMATIQUES | KINEMATIC_LIMITS |
| KTH_COLLISION_SOURCE | SOURCE_DE_KIEME_COLLISION | KTH_COLLISION_SOURCE |
| LANGUAGE | LANGUE | LANGUE |
| LAMBDA_HOLE | LAMBDA_TROU | LAMBDA_HOLE |
| LATEX | LATEX | LATEX |
| LEAKAGE | FUITE | LEAKAGE |
| LETHARGY | LETHARGIE | LETHARGY |
| LIMIT_SIZE | TAILLE_LIMITE | LIMIT_SIZE |
| LINE | DROITE | LINE |
| LINEAR | LINEAIRE | LINEAR |
| LIST | LISTE | LIST |
| LLNL_FISSION | FISSION_LLNL | LLNL_FISSION |
| LOCAL_ENERGY_ABSORPTION | ABSORPTION_ENERGIE_LOCALE | LOCAL_ENERGY_ABSORPTION |
| LOCAL_ENERGY_DEPOSITION | DEPOT_ENERGIE_LOCAL | LOCAL_ENERGY_DEPOSITION |
| LOWER_LIMIT | LIMITE_INFERIEURE | LOWER_LIMIT |
| MACROSCOPIC_REACTION_RATE | TAUX_DE_REACTION_MACROSCOPIQUE | MACROSCOPIC_REACTION_RATE |
| MAP | CARTE | MAP |
| MASS | MASSE | MASS |
| MATERIAL | MATERIAU | MATERIAL |
| MATRIX | MATRICE | MATRIX |
| MAXP_FOR_COLLISION_SITES | MAXP_POUR_LIEUX_COLLISION | MAXP_FOR_COLLISION_LOCUS |
| MAXWELL_SPECTRUM | SPECTRE_MAXWELL | MAXWELL_SPECTRE |
| MAX_POINTS | POINTS_MAX | MAX_POINTS |
| MEAN | MOYENNE | MEAN |
| MENDEL_LIBRARY | BIBLIOTHEQUE_MENDEL | MENDEL_LIBRARY |
| MESH | MAILLAGE | MAILLAGE |
| MESH_EDITION | MAILLAGE_EDITION | MESH_EDITION |
| MESH_INFO | MAILLAGE_INFO | MAILLAGE_INFO |
| MESH_VARIABLE | MAILLAGE_VARIABLE | MAILLAGE_VARIABLE |
| MEV | MEV | MEV |
| MODE | MODE | MODE |
| MONITORING | MONITORING | MONITORING |
| MONOKINETIC | MONOCINETIQUE | MONOCINETIQUE |
| MONO_DIR | MONO_DIRECTIONNEL | MONO_DIR |
| MULTI | MULTI | MULTI |
| MULTIGROUP | MULTIGROUPE | MULTIGROUP |
| MULTIGROUP_HOMOGENEOUS | MULTIGROUPE_HOMOGENE | MULTIGROUP_HOMOGENE |
| MULTIPLE | MULTIPLE | MULTIPLE |
| MU_TABULATED | TABULE_MU | MU_TABULATED |

| English (english) | French (french) | Default (old) |
|---|---|---|
| NAME | NOM | NAME |
| NATURAL | NATUREL | NATURAL |
| NAUCHI_KINETIC_PARAMETERS | PARAMETRES_CINETIQUES_NAUCHI | NAUCHI_KINETIC_PARAMETERS |
| NB_ATOM | NB_ATOME | NB_ATOM |
| NB_BATCH | NB_BATCH | NB_BATCH |
| NB_COLLISION | NB_COLLISION | NB_COLLISION |
| NB_POINTS | NB_POINTS | NB_POINTS |
| NEUTRON | NEUTRON | NEUTRON |
| NEW_SCORES | NEW_SCORES | NEW_SCORES |
| NJOY | NJOY | NJOY |
| NO_COMPOSITION | PAS_DE_COMPOSITION | NO_COMPOSITION |
| NO_ELECTRON_STRAGGLING | PAS_DE_STRAGGLING_ELECTRON | NO_ELECTRON_STRAGGLING |
| NONE | AUCUN | NONE |
| NONPOLARIZABLE | NONPOLARISABLE | NONPOLARIZABLE |
| NORM | NORME | NORME |
| NORMALIZE | NORMALISER | NORMALIZE |
| NOT_FOR_ZERO_VALUES | PAS_POUR_LES_ZEROS | NOT_FOR_ZERO_VALUES |
| NOT_INDUCED_BY_INTERACTION | NON_INDUIT_PAR_INTERACTION | NOT_INDUCED_BY_INTERACTION |
| NO_ANOMALOUS_SCATTERING_FACTOR | PAS_DE_FACTEUR_DE_DIFFUSION_ANORMALE | NO_ANOMALOUS_SF |
| NO_ELECTRON_MULTIPLE_SCATTERING | PAS_DE_COLLISION_MULTIPLE_ELECTRON | NO_ELECTRON_MULTIPLE_SCATTERING |
| NO_GEOMSOURCE_WARNING | PAS_D_AVERTISSEMENT_GEOMSOURCE | NO_GEOMSOURCE_WARNING |
| NU_FISSION | NU_FISSION | NU_FISSION |
| NUCLEI | NOYAUX | NUCLEI |
| NUCLEI_CONTRIBUTIONS | CONTRIBUTIONS_NUCLEI | NUCLEI_CONTRIBUTIONS |
| NUCLEUS | NOYAU | NUCLEUS |
| OMEGA | OMEGA | OMEGA |
| ONLY_PROMPT_GAMMA_ENERGY | ENERGIE_PROMPTE_GAMMA_SEULE | ONLY_PROMPT_GAMMA_ENERGY |
| ONLY_SVT | ONLY_SVT | ONLY_SVT |
| OPTICAL_VARIANCE_REDUCTION | PONDERATION_OPTIQUE | OPTICAL_PONDERATION |
| OPTIMIZE | OPTIMISATION | OPTIMIZE |
| ORDER | ORDRE | ORDER |
| ORDERED | ORDONNE | ORDERED |
| ORIENTATION | ORIENTATION | ORIENTATION |
| ORIENTED_FRONTIER | FRONTIERE_ORIENTEE | ORIENTED_FRONTIER |
| OUTPUT | OUTPUT | OUTPUT |
| PACKET_LENGTH | LONGUEUR_PAQUET | PACKET_LENGTH |
| PARALLEL | PARALLELE | PARALLEL |
| PARAM | PARAM | PARAM |
| PARTICLE | PARTICULE | PARTICLE |
| PATH | PARCOURS | PATH |
| PERTURBATION | PERTURBATION | PERTURBATION |
| PERTURBATION_LIST | LISTE_PERTURBATION | LIST_PERTURBATION |
| PERTURBED_DENOMINATOR | DENOMINATEUR_PERTURBE | PERTURBED_DENOMINATOR |
| PHI_TABULATED | TABULE_PHI | PHI_TABULATED |
| PHOTON | PHOTON | PHOTON |
| PHOTON_FROM_NEUTRON_SOURCE | PHOTON_DE_SOURCE_NEUTRON | PHOTON_FROM_NEUTRON_SOURCE |
| PHOTON_NEUTRON_RATIO | PHOTON_NEUTRON_RATIO | PHOTON_NEUTRON_RATIO |
| PHOTON_REPARTITION | PHOTON_REPARTITION | PHOTON_REPARTITION |
| PHOTO_NUCLEAR | PHOTO_NUCLEAIRE | PHOTO_NUCLEAR |
| PHOTONUCLEAR_ROULETTE | ROULETTE_PHOTONUCLEAIRE | PHOTONUCLEAR_ROULETTE |
| PHYSICS | PHYSIQUE | PHYSICS |
| PHYSICS_LIST | PHYSICS_LIST | PHYSICS_LIST |
| PIJ | PIJ | PIJ |
| PLANE | PLAN | PLANE |
| POINT | POINT | POINT |
| POINT_WISE | PONCTUEL | PUNCTUAL |
| POSITRON | POSITRON | POSITRON |
| PRECISION | PRECISION | PRECISION |
| PREINIT | PREINIT | PREINIT |
| PRODUCTION | PRODUCTION | PRODUCTION |

| English (english) | French (french) | Default (old) |
|---|---|---|
| PROMPT | PROMPTE | PROMPT |
| PROMPT_FISSION_ONLY | FISSION_PROMPTE_SEULE | PROMPT_FISSION_ONLY |
| PROMPT_NUSIGF | NUSIGF_PROMPTE | PROMPT_NUSIGF |
| PROTON | PROTON | PROTON |
| QFISSION | QFISSION | QFISSION |
| QUOTA_SAMPLING | QUOTA_SAMPLING | QUOTA_SAMPLING |
| RADIUS | RAYON | RADIUS |
| RANDOM | RANDOM | RANDOM |
| RAY_SPECTRUM | SPECTRE_RAIE | RAY_SPECTRE |
| REACTION | REACTION | REACTION |
| REACTION_RATES_SPHERICAL_HARMONICS | REACTION_RATES_SPHERICAL_HARMONICS | REACTION_RATES_SPHERICAL_HARMONICS |
| REACTIVITY_CONTRIBUTIONS | CONTRIBUTIONS_REACTIVITE | REACTIVITY_CONTRIBUTIONS |
| RECOIL_ENERGY | RECUL_ENERGIE | RECOIL_ENERGY |
| REDUCED_TRACKS | TRACKS_REDUITES | REDUCED_TRACKS |
| REFLECTION | REFLEXION | REFLECTION |
| REPLICATE | REPLICAT | REPLICATE |
| RESAMPLE_COLLISIONS | REFAIRE_COLLISIONS | RESAMPLE_COLLISIONS |
| RESCALE_DELAYED_FISSION_PHOTONS | NORMALISE_PHOTONS_RETARDES_FISSION | RESCALE_DELAYED_FISSION_PHOTONS |
| RESPONSES | REPONSES | REPONSES |
| RING | ANNEAU | RING |
| ROOT_GEOMETRY | ROOT_GEOMETRIE | ROOT_GEOMETRY |
| ROOT_LIMIT | ROOT_LIMITE | ROOT_LIMIT |
| MORET_GEOMETRY | MORET_GEOMETRIE | MORET_GEOMETRY |
| RUSSIAN_ROULETTE | ROULETTE_RUSSE | RUSSIAN_ROULETTE |
| SAME_HISTORY | MEME_HISTOIRE | SAME_HISTORY |
| SCAN_GEOMCOMP | BALAYAGE_GEOMCOMP | SCAN_GEOMCOMP |
| SCAN_GEOM_1D | BALAYAGE_GEOM_1D | SCAN_GEOM_1D |
| SCATTERING_TRANSFER_FUNCTION | FONCTION_TRANSFERT_DE_DIFFUSION | SCATTERING_TRANSFER_FUNCTION |
| SCORE | SCORE | SCORE |
| SCORE_COLL | SCORE_COLL | SCORE_COLL |
| SCORE_FXPT | SCORE_FXPT | SCORE_FXPT |
| SCORE_SEGMENTS | SCORE_SEGMENTS | SCORE_SEGMENTS |
| SCORE_SURF | SCORE_SURF | SCORE_SURF |
| SCORE_TRACK | SCORE_CORDES | SCORE_TRACK |
| SECONDARY_SPECTRUM | SPECTRE_SECONDAIRE | SECONDARY_SPECTRUM |
| SECTION | SECTION | SECTION |
| SET | SET | SET |
| SET_OMEGA | SET_OMEGA | SET_OMEGA |
| SHANNON | SHANNON | SHANNON |
| SHIELDING | PROTECTION | PROTECTION |
| SIMULATION | SIMULATION | SIMULATION |
| SIX_FACTOR_FORMULA | FORMULE_DES_SIX_FACTEURS | SIX_FACTOR_FORMULA |
| SIZE | TAILLE | SIZE |
| SLOWING_DOWN_CURRENT | COURANT_DE_RALENTISSEMENT | SLOWING_DOWN_CURRENT |
| SMOOTHING | LISSAGE | SMOOTHING |
| SOBOL_POINTS | POINTS_SOBOL | SOBOL_POINTS |
| SOURCE | SOURCE | SOURCE |
| SOURCES_ALL_TABULATIONS | SOURCES_ALL_TABULATIONS | SOURCES_ALL_TABULATIONS |
| SOURCES_CONTRIBUTIONS | SOURCES_CONTRIBUTIONS | SOURCES_CONTRIBUTIONS |
| SOURCES_INTER_VOLU | SOURCES_INTER_VOLU | SOURCES_INTER_VOLU |
| SOURCES_LIST | LISTE_SOURCES | LIST_SOURCES |
| SOURCES_NUMBER | SOURCES_NUMBER | SOURCES_NUMBER |
| SOURCE_STORAGE | SOURCE_STOCKAGE | SOURCE_STOCKAGE |
| SOURCE_WEIGHTING | SOURCE_PONDEREE | SOURCE_WEIGHTING |
| SPATIAL | SPATIAL | SPATIAL |
| SPECTRUM | SPECTRE | SPECTRE |
| SPEED_UP | VITESSE | SPEED_UP |
| SPHERE | SPHERE | SPHERE |
| SPLIT | DUPLIQUE | SPLIT |
| SPONTANEOUS_FISSION | FISSION_SPONTANEE | SPONTANEOUS_FISSION |

| English (english) | French (french) | Default (old) |
|---|---|---|
| SQRT | RACINE_CARREE | SQRT |
| STORAGE | STOCKAGE | STOCKAGE |
| STORE_IN_FILE | SORTIE_FICHIER | STORE_IN_FILE |
| STORE_SOURCES_IN_FILE | SORTIE_SOURCES | STORE_SOURCES_IN_FILE |
| STORE_COLLISIONS_IN_FILE | SORTIE_COLLISIONS | STORE_COLLISIONS_IN_FILE |
| STORE_LEAKAGE_IN_FILE | SORTIE_FUITES | STORE_LEAKAGE_IN_FILE |
| STRENGTH | FORCE | STRENGTH |
| SUM | SOMME | SOMME |
| SUPER_HISTORY | SUPER_HISTOIRE | SUPER_HISTORY |
| SUPER_HISTORY_CYCLE_LENGTH | SUPER_HISTORY_CYCLE_LENGTH | SUPER_HISTORY_CYCLE_LENGTH |
| SUPER_HISTORY_KINETIC_PARAMETERS | PARAMETRES_CINETIQUES_SUPER_HISTORY | SUPER_HISTORY_KINETIC_PARAMETERS |
| SUPER_HISTORY_PERTURBATION_LIST | LISTE_PERTURBATIONS_SUPER_HISTORY | SUPER_HISTORY_PERTURBATION_LIST |
| SUPER_HISTORY_SENSITIVITY_LIST | LISTE_SENSIBILITES_SUPER_HISTORY | SUPER_HISTORY_SENSITIVITY_LIST |
| SUPPRESS_MESH_EDITION_TEXT | SUPPRESSION_EDITION_TEXTE_MAILLAGE | SUPPRESS_MESH_EDITION_TEXT |
| SURF | SURF | SURF |
| SURFACE | SURFACE | SURFACE |
| SVT | SVT | SVT |
| SWITCH_ZERO_TO_MAX_VALUES | CHANGER_LES_ZEROS_EN_MAX | SWITCH_ZERO_TO_MAX_VALUES |
| T4HOMOT4 | T4HOMOT4 | T4HOMOT4 |
| TABULATED | TABULE | TABULATED |
| TARGET | CIBLE | TARGET |
| TARGET_SURF | SURFACE_CIBLE | TARGET_SURF |
| TAYLOR_PERTURBATION | TAYLOR_PERTURBATION | TAYLOR_PERTURBATION |
| TEMPERATURE_INTERPOL | INTERPOL_TEMPERATURE | TEMPERATURE_INTERPOL |
| THERMAL | THERMIQUE | THERMAL |
| THICK_TARGET_BREMSSTRAHLUNG | BREMSSTRAHLUNG_CIBLE_EPAISSE | THICK_TARGET_BREMSSTRAHLUNG |
| THRESHOLD_RATIO | RATIO_SEUIL | THRESHOLD_RATIO |
| TIME | CINETIQUE | CINETIC |
| TIME_DISTRIBUTION | DISTRIBUTION_TEMPS | TIME_DISTRIBUTION |
| TIME_INF | TEMPS_INF | TIME_INF |
| TIME_SUP | TEMPS_SUP | TIME_SUP |
| TOTAL | TOTAL | TOTAL |
| TOTAL_FISSION | FISSION_TOTALE | TOTAL_FISSION |
| TOWARDS | VERS | TOWARDS |
| TRACK | CORDES | TRACK |
| TRACKS | TRACKS | TRACKS |
| TRACK_INTEGRATION | INTEGRATION_CORDES | TRACK_INTEGRATION |
| TRANSLATION | TRANSLATION | TRANSLATION |
| TRANSPORT_BIASING | TRANSPORT_BIAISE | TRANSPORT_BIASING |
| TRIANGLE | TRIANGLE | TRIANGLE |
| TRITON | TRITON | TRITON |
| TYPE | TYPE | TYPE |
| UNCONSTRAINED_COEFFICIENTS | COEFFICIENTS_NON_CONTRAINTS | UNCONSTRAINED_COEFFICIENTS |
| UNCOLLIDED | DIRECT | UNCOLLIDED |
| UNFACTORIZED | NON_FACTORISE | UNFACTORIZED |
| UNIFORM | UNIFORME | UNIFORM |
| UNITS | UNITES | UNITS |
| UPPER_LIMIT | LIMITE_SUPERIEURE | UPPER_LIMIT |
| USE_FILE | DEPUIS_FICHIER | USE_FILE |
| VARIANCE_REDUCTION | PONDERATION | PONDERATION |
| VECTOR | VECTEUR | VECTOR |
| VOID | VIDE | VOID |
| VOLENERCUT | COUPURE_ENER_PAR_VOL | VOLENERCUT |
| VOLSURF | VOLSURF | VOLSURF |
| VOLUME | VOLUME | VOLU |
| VOLUME_COLLISION | VOLUME_COLLISION | VOLUME_COLLISION |
| VOLUME_SPLITTING | VOLUME_SPLITTING | VOLUME_SPLITTING |
| VOLUME_TARGET | VOLUME_CIBLE | VOLUME_TARGET |
| VOLUMIC_SCORES | SCORES_VOLUMIQUES | VOLUMIC_SCORES |
| VOV | VOV | VOV |

| English (english) | French (french) | Default (old) |
|---|---|---|
| WATT_SPECTRUM | SPECTRE_WATT | WATT_SPECTRE |
| WCM | WCM | WCM |
| WEIGHTED_COURANT | WEIGHTED_COURANT | WEIGHTED_COURANT |
| WEIGHT_OF_BATCH | WEIGHT_OF_BATCH | WEIGHT_OF_BATCH |
| WEIGHT_WINDOWS | WEIGHT_WINDOWS | WEIGHT_WINDOWS |
| WHOLEGRID | DECOUPAGE_COMPLET | WHOLE_DECOUPAGE |
| WIELANDT_CYCLE_LENGTH | WIELANDT_CYCLE_LENGTH | WIELANDT_CYCLE_LENGTH |
| WIELANDT_KINETIC_PARAMETERS | WIELANDT_KINETIC_PARAMETERS | WIELANDT_KINETIC_PARAMETERS |
| WIELANDT_METHOD | WIELANDT_METHOD | WIELANDT_METHOD |
| WIELANDT_PERTURBATION_LIST | LISTE_PERTURBATIONS_WIELANDT | WIELANDT_PERTURBATION_LIST |
| WIELANDT_SENSITIVITY_LIST | LISTE_SENSIBILITES_WIELANDT | WIELANDT_SENSITIVITY_LIST |
| WINDOW | FENETRE | WINDOW |
| WINDOW_VARIABLE | FENETRE_VARIABLE | WINDOW_VARIABLE |
| XDR | XDR | XDR |
| XDR_BUFFER | XDR_BUFFER | XDR_BUFFER |
| XML_EXPORT | SORTIE_XML | XML_EXPORT |
| XML_NOCOMPLETE | XML_PARTIEL | XML_NOCOMPLETE |
| XVGR | XVGR | XVGR |
| YIELD_SIGMA_N2N | YIELD_SIGMA_N2N | YIELD_SIGMA_N2N |
| YIELD_SIGMA_N3N | YIELD_SIGMA_N3N | YIELD_SIGMA_N3N |
| YIELD_SIGMA_N4N | YIELD_SIGMA_N4N | YIELD_SIGMA_N4N |
| PEPIN: | | |
| DAY | JOUR | JOUR |
| END_PEPIN | FIN_PEPIN | FIN_PEPIN |
| HISTORY | HISTORIQUE | HISTORIQUE |
| HOUR | HEUR | HEUR |
| ISOTOPES | ISOTOPES | ISOTOPES |
| MINU | MINU | MINU |
| PEPIN | PEPIN | PEPIN |
| PEPIN_GRID | PEPIN_DECOUPAGE | PEPIN_DECOUPAGE |
| PFNL | PFNL | PFNL |
| SCORE_MODE | SCORE_MODE | SCORE_MODE |
| SECO | SECO | SECO |
| YEAR | ANNE | ANNE |

# Appendix D

# BIBLIOGRAPHY

This appendix lists the documents quoted herein, as well as additional references.

# Bibliography

[1] TRIPOLI-3, Monte Carlo transport code for neutral particles. version 3.5. User's Manual. Technical report, CEA-R-5950, 2001.

[2] Gandini A. Nucl. Sci. Eng., 79:426–430, 1981.

[3] Working Group ANS-6.1.1 American Nuclear Society Standards Comittee. American National Standard Neutron and Gamma-ray Flux-to-Dose-Rate Factors. Technical Report ANSI/ANS-6.1.1-1977 (N666), American National Standards Institute, 1977.

[4] Becker B. PhD thesis, Institut fur Kernenergetik und Energiesysteme, University of Stuttgart, Germany, 2010.

[5] Becker B., Dagan R., and Lohnert G. Ann. Nucl. Eng., 36:470–474, 2009.

[6] M. J. Berger and S. M. Seltzer. Bremsstrahlung and photoneutrons from thick tungsten and tantalum targets. Phys. Rev. C, 2:621–631, August 1970.

[7] T. Booth. Genesis of the Weight Window and Weight Window Generator in MCNP - A Personal History. Technical Report LA-UR-06-5807, Los Alamos National Laboratory, 2006.

[8] J.-P. Both. A Communication Library for TRIPOLI-4. In Nuclear Energy Agency, editor, Advanced Monte Carlo Computer Programs for Radiation Transport (OECD Documents), Saclay (France), April 1993.

[9] J.-P. Both, G. Dejonghe, J.-C. Nimal, P. Dos Santos, and T. Vergnaud. Automated importance generation and biaising techniques for Monte-Carlo shielding techniques by the TRIPOLI-3 code. In PHYSOR 90 : Conférence Internationale sur la Physique des Réacteurs pour la Conception, la Modélisation et l'Exploitation, pages (XII) 59–68, Marseille (France), Avril 1990.

[10] J.-P. Both, H. Derriennic, B. Morillon, and J.-C. Nimal. A survey of TRIPOLI-4. In Inc. American Nuclear Society, editor, 8th International Conference on Radiation Shielding, Arlington (USA), April 1994.

[11] J.-P. Both and Y.K. Lee. Computations of homogenized multigroup cross sections with the Monte-Carlo code TRIPOLI-4. In Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications, pages 439–445, Saratoga (USA), October 1997.

[12] J.-P. Both, G. Néron de Surgy, and Y. Péneliau. Notice d'utilisation de TRIPOLI-4: code de transport de particules par la méthode de Monte Carlo. Technical report, CEA-R-6044, 2003.

[13] J.-P. Both, J.-C. Nimal, and T. Vergnaud. Automated importance generation and biaising techniques for Monte-Carlo shielding techniques by the TRIPOLI-3 code. Progress in Nuclear Energy, 24(1-3):273–281, 1990.

[14] J.-P. Both and Y. Péneliau. The Monte Carlo code TRIPOLI-4 and its First Benchmark Interpretations. In PHYSOR 96 : International Conference on the Physics of Reactors, page C 175, Mito, Ibaraki (Japan), September 1996.

[15] J.P. Both, Y. K. Lee, Y. Péneliau, O. Petit, and B. Roesslinger. TRIPOLI-4 - A Three Dimensional Polykinetic Particle Transport Monte Carlo Code. In SNA'2003, Paris (France), September, 22–24 2003.

[16] J.P. Both, Y. K. Lee, Y. Péneliau, O. Petit, and B. Roesslinger. TRIPOLI-4 : Code de transport Monte Carlo - Fonctionnalités et Applications. In Société Française de Radioprotection Actes Journées Scientifiques Francophones, Sochaux (France), October, 2–3 2003.

[17] S. Bourganel, O.Petit, M. Chiron, and C.M. Diop. Pseudo-Adjoint Method for 3D Monte Carlo Particle Transport. Application to Parametric Studies. In The American Nuclear Society's 14th Biennial Topical Meeting of the Radiation Protection and Shielding Division, Carlsbad (New Mexico, USA), April, 3–6 2006.

[18] E. Brun, F. Damian, C.M. Diop, E. Dumonteil, F.X. Hugot, C. Jouanne, Y.K. Lee, F. Malvagi, A. Mazzolo, O. Petit, J.C. Trama, T. Visonneau, and A. Zoia. TRIPOLI-4, CEA, EDF and AREVA Reference Monte Carlo Code. Annals of Nuclear Energy, 82:151–160, 2015.

[19] G. Campioni. Etude et modélisation des sources froides de neutrons. PhD thesis, CEA, 2004.

[20] H. Carcreff, A. Alberman, L. Barbot, F. Rozenblum, D. Beretz, and Y.K. Lee. Dosimetry Requirements for Pressure Vessel Steels Toughness Curve in the Ductile to Brittle Range. In J. of ASTM International (JAI), Volume 3, Issue 3, March 2006.

[21] H. Chang, X. Jing, Y. Yang, Y. K. Lee, and X. Raepsaet. Monte Carlo Method Treatment of Stochastic Geometry in a Pebble Bed Reactor. In The Monte Carlo Method: Versatility Unbounded In A Dynamic Computing World, Chattanooga (Tennessee, USA), April, 17–21 2005.

[22] H. Chang, X. Raepsaet, F. Damian, Y. K. Lee, O. Koberl, X. Jing, and Y. Yang. Analysis of HTR-10 First Criticality with Monte Carlo Code TRIPOLI-4.3. In Int. Topical Meeting on HTR Technology, Bejing (China), September 2004.

[23] IBM Corporation. QuickStart Guide Version 3 Release 1 Modification 4. In IBM Visualization Data Explorer, pages 85–96, Yorktown Heights (USA), May 1997.

[24] D. Cullen, R. Blomquist, C. Dean, D. Heinrichs, M. Kalugin, M. Lee, Y. K. Lee, R. MacFarlane, Y. Nagaya, and A. Trkov. How Accurately can we Calculate Thermal Systems ? Technical Report UCRL-TR-203892, April 2004.

[25] Lee D., Smith K., and Rhodes J. Ann. Nucl. Eng., 36:274–280, 2009.

[26] Mancusi D. and Zoia A. Chaos in eigenvalue search methods. Ann. Nucl. Eng., 112:354–363, 2018.

[27] Pierre J. Dossantos-Uzarralde. Automatisation de la recherche des fonctions d'importance dans un code Monte-Carlo. PhD thesis, Université Paris VI, 1990.

[28] E. Dumonteil, A. Le Peillet, Y. K. Lee, O. Petit, C. Jouanne, and A. Mazzolo. Source convergence diagnostics using Boltzmann entropy criterion - Application to different OECD/NEA criticality benchmarks with the 3-D Monte Carlo code TRIPOLI-4. In PHYSOR'2006, Vancouver (BC, Canada), September 2006.

[29] C.M. Diop et al. TRIPOLI-4 workshop. In M&C 2005 Conference Proceedings, Avignon (France), September, 12–15 2005.

[30] C.M. Diop et al. TRIPOLI-4 : a 3D Continuous-energy Monte Carlo transport code. In PHYTRA-1, Marrakech (Maroc), March, 14–16 2007.

[31] E. Dumonteil et al. TRIPOLI-4 workshop. In M&C 2007 Conference Proceedings, Monterrey, (CA, USA), April, 15–19 2007.

[32] F. Laye et al. Comparison of activation calculations with measurements for the biological shield of the Bugey 1 Graphite-Gas Reactor. In 5th International Decommissioning and Radioactive Waste Workshop, Kendal (UK), 30 October - 2 November 2006.

[33] J.C. Trama et al. Review of the TRIPOLI-4 Monte Carlo transport code. In ICAPP 2007, Nice (France), May, 13–18 2007.

[34] O. Petit et al. TRIPOLI-4 : a 3D Continuous-Energy Monte Carlo Transport Code. In ANS Winter Meeting, Albuquerque (NM, USA), November, 12–16 2006.

[35] P. Mac Farlane and al. ENDF102 : Data formats and procedures for the evaluated nuclear data file ENDF/B-VI and ENDF/B-VII. Technical Report BNL-90365-2009, National Nuclear Data Center, Brookhaven National Laboratory, 2009.

[36] Robert E. Mac Farlane and al. HEATR user guide, version 91.91. Technical report, Nuclear Theory and Applications Group, Los Alamos National Laboratory, 2000.

[37] Margaux Faucher, Davide Mancusi, and Andrea Zoia. New kinetic simulation capabilities for tripoli-4Â®: Methods and applications. Annals of Nuclear Energy, 120:74–88, 2018.

[38] Margaux Faucher, Davide Mancusi, and Andrea Zoia. Variance-reduction methods for Monte Carlo kinetic simulations. In International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, M&C 2019, pages 130–139, 2019.

[39] H. Flyvbjerg and H.G. Petersen. Error estimates on averages of correlated data. J. Chem. Physics, 91:461, 1989.

[40] E. Gagnier, Y.K. Lee, L. Aguiar, and N. Vedrenne. Validation of the 3D TRIPOLI-4.3 Monte Carlo code for metallic fissile media configurations with JEF2.2 and ENDF/B6 evaluations. In 7th Int. Conf. on Nuclear Criticality Safety,ICNC'2003, Tokai-mura (Japan), October, 20–24 2003.

[41] Geant4 collaboration. Geant4 Application Developer Guide, December 2017. version 10.4.

[42] F. X. Hugot and Y. K. Lee. Verification Tools for TRIPOLI-4 Criticality Code Validation. In ICNC'2007, St. Petersburg (Russia), 28 May - 1st June 2007.

[43] F.-X. Hugot, Y.-K. Lee, and F. Malvagi. Recent R&D around the Monte-Carlo code Tripoli-4 for criticality calculation. In Physor'08, Interlaken, 2008.

[44] F.X. Hugot, A. Jinaphanh, C. Jouanne, C. Larmier, Y.K. Lee, D. Mancusi, O. Petit, T. Visonneau, and A. Zoia. Overview of the TRIPOLI-4 Monte Carlo code, version 12. EPJN Nuclear Sci. Technol., 10:17, 2024.

[45] Working Group International Commission on Radiological Protection. International Commission on Radiological Protection. Technical report, 1997.

[46] Donald E. Knuth. The Art of Computer Programming, volume 2 (Seminumerical Algorithms). Addison-Wesley, Boston, 1998.

[47] A. L'Abbate, T. Courau, and E. Dumonteil. Monte Carlo calculations : source convergence and dominant ratio in an infinite lattice using MCNP and TRIPOLI-4. In PHYTRA-1, Marrakech (Maroc), March, 14–16 2006.

[48] S. Lahaye, A. Anne, R. Baron, T.-D Huynh, and A. Tsilanizara. New Bateman equation solvers in MENDEL version 3.1. In ICNC'2023, Sendai (Japan), October, 1–6 2023.

[49] C. Larmier. Stochastic particle transport in disordered media: beyond the Boltzmann equation. PhD thesis, Université Paris-Saclay, 2018.

[50] Y. K. Lee. TRIPOLI-4 solution for SIGMA graphite moderated Am-Be neutron field (CONRAD-P3). In EURADOS Working Group 6. Workshop (CONRAD WP4) on Uncertainty assessment in Computational Dosimetry, Bologna (Italy), October 8–10 2007.

[51] Y. K. Lee. Validation of TRIPOLI-4 Perturbation in Criticality Safety Analysis. In ICNC'2007, St. Petersburg (Russia), 28 May - 1st June 2007.

[52] Y. K. Lee and J.C. Trama. Analysis of Manganese Bath Method for Am-Be Neutron Source Calibration using TRIPOLI-4 Monte Carlo code (CONRAD-P5). In EURADOS Working Group 6. Workshop (CONRAD WP4) on Uncertainty Assessment in Computational Dosimetry, Bologna (Italy), October 8–10 2007.

[53] Y.K. Lee, , S.H. Zheng, G. Néron end J.P. Both, Y. Pénéliau, and M.C. Diop. ICNC'9, Cristal, Criticality Safety Package Validation: TRIPOLI-4 Monte Carlo Code, JEF2.2 library and ICSBEP Experiments. In Sixth International Conference on Nuclear Criticality Safety, Versailles, France, 20-24 September 1999.

[54] Y.K. Lee. A burnup credit calculation methodology for PWR spent fuel transportation. In PATRAM'98 : the 12th Int. Conf. on the packaging and transportation of radioactive materials. Volume 2, page 825, Albuquerque, (USA), May 1998.

[55] Y.K. Lee. Evaluation of CRISTO II Storage arrays Benchmark with TRIPOLI-4.2 Criticality Calculations. In Monte Carlo 2000, page 809, Lisbon (Portugal), October, 23–26 2000.

[56] Y.K. Lee. Analysis of the LEU-COMP-THERM-049 MARACAS Critical Configurations using TRIPOLI-4.3 3D Lattices Geometry and JEFF3.0 Library. In 7th Int. Conf. on Nuclear Criticality Safety,ICNC'2003, Tokai-mura (Japan), October, 20–24 2003.

[57] Y.K. Lee. Analysis of the NRC PCA Pressure Vessel Dosimetry Benchmark using TRIPOLI-4.3 Monte Carlo Code and ENDF/B-VI.4, JEF2.2 and IRDF-90 Libraries. In ANS Topical Meeting in Nuclear Mathematics & Computations M&C'2003, Gatlinburg (TN, USA), April, 6–11 2003.

[58] Y.K. Lee. QUADOS Benchmark - TRIPOLI-4.3 solution for Problem P8-C. In Proc. International Workshop on the Usage of Computational Codes in Radiation Dosimetry, page 326, Bologna (Italy), July, 14–16 2003.

[59] Y.K. Lee. Analysis of Neutron and Photon Response of a TLD-Albedo Personal Dosemeter on an ISO Slab Phantom Using TRIPOLI-4.3 Monte Carlo Code. Radiation Protection Dosimetry, pages 329–333, 2005.

[60] Y.K. Lee. Analysis of the VENUS-3 PWR Pressure Vessel Surveillance Benchmark Using TRIPOLI-4.3 Monte Carlo Code. In 12th Int. Symposium on Reactor Dosimetry, Gatlinburg (TN., USA), May, 8–13 2005.

[61] Y.K. Lee. Evaluation of N50 Neutron Slab Monitor Detection Efficiency with TRIPOLI-4.3 Monte Carlo Code. Radiation Protection Dosimetry, pages 369–374, 2005.

[62] Y.K. Lee. Analysis of the VENUS-3 PWR Pressure Vessel Surveillance Benchmark Using TRIPOLI-4.3 Monte Carlo Code. In J. of ASTM International (JAI), Volume 3, Issue 10, November/December 2006.

[63] Y.K. Lee. Comparative Analysis of Isotopic Composition of Spent Fuel from Takahama-3 PWR PIE database using TRIPOLI-PEPIN Code (Abstract). In PHYSOR'2006, Vancouver (BC, Canada), September 2006.

[64] Y.K. Lee. Investigation of Cf-252 Fission Spectra and SINBAD Water Shielding Benchmarks Using TRIPOLI-4.3 Monte Carlo Code. In The American Nuclear Society's 14th Biennial Topical Meeting of the Radiation Protection and Shielding Division, Carlsbad (New Mexico, USA), April, 3–6 2006.

[65] Y.K. Lee. Use of TRIPOLI-4.3 Tally functions to Investigate Pin Power and Assembly Power Maps of PWR Critical Lattices Experiments. In PHYSOR'2006, Vancouver (BC, Canada), September 2006.

[66] Y.K. Lee, JC. David, and H. Carcreff. A Gamma Heating Calculation Methodology for Research Reactor Application. In 5th Int. Topical Meeting on Research Reactor Fuel Management RRFM 2001, page 147, Aachen, April 2001.

[67] Y.K. Lee, G. Néron end J.P. Both, Y. Pénéliau, and M.C. Diop. Validation of Monte Carlo Code TRIPOLI-4 with PWR Critical Lattices by using JEF2.2 and ENDF/B-VI Evaluations. In JIC MM && SNA, page 253, Saratoga Springs, New-York, USA, October 1997.

[68] Y.K. Lee, A. Monnier, J.-P. Both, and J.-C. Nimal. Benchmark validation of the criticality code TRIMARAN-2. In The 5th Int. Conf. on Nucl. Criticality Safety, Volume 1, pages 3–12, Albuquerque, (USA), September 1995.

[69] Y.K. Lee, G. Néron, J.-P. Both, Y. Pénéliau, and C.M. Diop. Validation of Monte Carlo code TRIPOLI-4 with PWR critical lattices by using JEF2.2 and ENDF/B-VI evaluations. In Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications, pages 253–262, Saratoga (USA), October 1997.

[70] Y.K. Lee, G. Néron, M. Nobile, J.-P. Both, Y. Pénéliau, and C.M. Diop. Monte Carlo transport code for research reactor in core calculations. In The 10th Int. Symposium on Reactor Dosimetry (é paraétre), pages 3–12, Osaka, (Japan), September 1999.

[71] Y.K. Lee, B. Roesslinger, and A. Tsilanizara. TRIPOLI-PEPIN depletion code and its first numerical benchmarks for PWR high-burnup UO2 and MOX spent fuel. In M&C 2005 Conference Proceedings, Avignon (France), September, 12–15 2005.

[72] O. Litaize and O. Serot. Investigation of phenomenological models for the Monte Carlo simulation of the prompt fission neutron and $\gamma$ emission. Phys. Rev. C, Nucl. Phys., 82(5), 2010.

[73] S. Loubière, R. Sanchez, M. Coste, A. Hébert, and I. Smijarevic Z. Stankovski, C. Van Der Gucht. APOLLO2, twelve years later. In M & C 99, Madrid, Spain, September, 27–30 1999.

[74] H. Louvin. Development of an adaptive variance reduction technique for Monte Carlo particle transport. PhD thesis, Université Paris-Saclay, 2017.

[75] T. Nishimura M. Matsumoto. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. In ACM: transaction on Modeling and Computer Simulations: Special Issue on Uniform Random Number Generation, 1998.

[76] Wieczorek M.A. and Meschede M. Shtools: Tools for working with spherical harmonics. Geochemistry, Geophysics, Geosystems, 19:2574–2592, 2018.

[77] Li Mao and J.C. Nimal. Semianalytical procedures for obtaining low-order non-equally-probable step-function representations in multigroup Monte Carlo. Nucl. Sci. Eng. 136, pages 409–414, 2000.

[78] B. Morillon. Une technique de biaisage automatique. In Nuclear Energy Agency, editor, Advanced Monte Carlo Computer Programs for Radiation Transport (OECD Documents), pages 71–77, Saclay (France), April 1993.

[79] B. Morillon. Méthode de Monte Carlo non analogue, application à la simulation des neutrons. PhD thesis, Université Paris XI, 1996.

[80] B. Morillon, J.-P. Both, H. Derriennic, C.M. Diop, and J.-C. Nimal. How many Collisions for a Neutron in a TRIPOLI Non Analogue Simulation? In 8th International Conference on Radiation Shielding, pages 424–430, Arlington (USA), April 1994.

[81] B. Morillon, J.-P. Both, and J.-C. Nimal. Importance function by collision probability for Monte Carlo code TRIPOLI. Karlsruhe (Germany), April 1993.

[82] P. Mosca and E. Masiello. Specification of the APOLLO3(R) external library file for neutron microscopic cross sections. Technical Report DEN/DANS/DM2S/SERMA/LLPR/NT/12-5241/B, Note CEA/SERMA, 2012.

[83] P. Mosca and E. Masiello. Specification of the APOLLO3(R) external library file for gamma calculations. Technical Report DEN/DANS/DM2S/SERMA/LLPR/NT/17-6241/A, Note CEA/SERMA, 2017.

[84] Terranova N., Mancusi D., and Zoia A. New perturbation and sensitivity capabilities in TRIPOLI-4. Ann. Nucl. Eng., 121:335–349, 2018.

[85] Terranova N., Truchet G., Zmijarevic I., and Zoia A. Adjoint neutron flux calculations with Tripoli-4: Verification and comparison to deterministic codes. Ann. Nucl. Eng., 114:136–148, 2018.

[86] Yasunobu Nagaya and Takamasa Mori. Impact of perturbed fission source on the effective multiplication factor in Monte Carlo perturbation calculations. Journal of Nuclear Science and Technology, 42(5):428–441, 2005.

[87] Yasushi Nauchi and Takanori Kameyama. Proposal of direct calculation of kinetic parameters $\beta_{eff}$ and $\lambda$ based on continuous energy Monte-Carlo method. Journal of Nuclear Science and Technology., 42:503–514, 2005.

[88] J.-C. Nimal and T. Vergnaud. TRIPOLI : a general Monte Carlo code, present state and future prospects. Progress in Nuclear Energy, 24(1-3):195–200, 1990.

[89] J.-C. Nimal and T. Vergnaud. TRIPOLI capabilities proved by a set of solved problems. Progress in Nuclear Energ, 24(1-3):201–210, 1990.

[90] ORNL. Advantg - An Automated Variance Reduction Parameter Generator. Rev.1, August 2015. Report number: ORNL/TM-2013/416.

[91] Y. Pénéliau. Electron Photon Shower Simulation. Advanced Monte Carlo for Radiation Physics, October 2000.

[92] Y. Pénéliau and J.P. Both. Parallelization of the Monte Carlo Code TRIPOLI-4. Mathematical and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications, page 412, September 1999.

[93] N. Petrov, NP. Kolev, G. Todorova, FX Hugot, and T. Visonneau. Tripoli-4 solutions to VVER-1000 assembly and core benchmarks. In M & C, Saratoga Springs, 2009.

[94] G.M. Prabhu S. Aluru and J. Gustafson. A random number generator for parallel computers. In Parallel Computing, 1992.

[95] R. Sanchez, J. Mondot, Z. Stankovski, A. Cossic, and I. Zmijarevic. APOLLO2 : A user-oriented, portable, modular code for multigroup transport assembly calculations. Nuclear Science and Engineering, 100:352–362, 1988.

[96] S. M. Seltzer and M. J. Berger. Bremsstrahlung energy spectra from electrons with kinetic energy 1 keV–10 GeV incident on screened nuclei and orbital electrons of neutral atoms with $Z = 1$–100. Atom. Data Nucl. Data, 35(3):345–418, 1986.

[97] Bart Laurens Sjenitzer. The Dynamic Monte Carlo Method for Transient Analysis of Nuclear Reactors. PhD thesis, Delft University of Technology, Netherlands, 2013.

[98] Guillaume Truchet, Pierre Leconte, Alain Santamarina, Emeric Brun, Frédéric Damian, and Andrea Zoia. Computing adjoint-weighted kinetics parameters in Tripoli-4® by the Iterated Fission Probability method. Ann. Nucl. Eng., 85:17–26, 2015.

[99] A. Tsilanizara and C.M. Diop et al. DARWIN: An evolution Code System for a Large Range of Applications. Technical report, Journal of Nuclear Science and Technology, Supplement 1, 2000.

[100] T. Vergnaud. Overview of Monte Carlo Methodologies in TRIPOLI-3. In Nuclear Energy Agency, editor, Advanced Monte Carlo Computer Programs for Radiation Transport (OECD Documents), pages 21–30, Saclay (France), Avril 1993.

[101] B. P. Welford. Note on a method for calculating corrected sums of squares and products. Technometrics, 4 (3):419–420, 1962.

[102] ESARDA NDA Working Group. Results of Monte Carlo Simple Case Benchmark Exercise. In ESARDA Bulletin 33. http://esarda2.jrc.it/bulletin/bulletin_33/index.html, February 2006.

[103] S.H. Zheng, B. Morillon, C.M. Diop, P. Ribon, C. Raepsaet, and J.-C. Nimal. Probability Tables, Pointwise and Multigroup Neutron Cross Sections in the Monte Carlo code TRIPOLI-3. In Inc. American Nuclear Society, editor, 8th International Conference on Radiation Shielding, pages 398–405, Arlington (USA), April 1994.

[104] I. Zmijarevic and R. Sanchez. Deterministic solutions for 3D Kobayashi benchmarks. Prog. Nucl. Energy, 39(2):207–221, 2001.

# Appendix E

# INDEX

# Index

activation, 468–476
adjoint flux, 300
albedo, 121
albedo search, 303
AMS, see biasing
analog transport, 289
atom relaxation, 26

batch, 286
beta effective, 296
biasing, 238–274
    Adaptive Multilevel Splitting (AMS), 261
    analytical, 241, 244, 251
    CADIS technique, 274
    energy, 252
    for coupled simulations, 255
    graphical display, 257
    IDT importance maps, 274
    mesh, 245
    source, 156, 157, 164
    spatial, 241
    time, 252
    weight windows, 270
Boron search, 303
boundary conditions, 120–125
bremsstrahlung, 319

CADIS technique, see biasing
calculation time, 107
cell, 50, 81, 84, 87–99, 220
charge deposition, see deposited charge, 218
checkpoint restart, see Keep Result
collision files, 399–401
collisions sites, 337–342
colors, 100
commands, 38
comments, 44–45
compositions, 126–137
conventions, 32, 34
coupling
    GEANT4, 187, 478–488
    MORET, 489–490
    ROOT, 409–417
critical parameter search, 303
criticality, 294
cross-sections, 24
    multigroup, 132
    multigroup homogenized, 131
    output, 385
    pointwise, 128
current, 189

DBRC, 315
decay constants, 296
density, 331–332
depletion, 389–390
deposited charge, 190
deposited charge, 208
detector, 241
detector deposited energy spectrum, 190
dictionary, 28–30
directions of interest, 250
directory, 26–27
distribution
    angular, 153
    energy, 161
    spatial, 140
    time, 166
Diven dispersion factor, 307
dose, 189
dpa, 190

elastic kernel, 315
electromagnetic shower, 129, 192, 319
electron, 287, 319
electron source, 190
electronic properties, 129
energy deposition, 189
entropy, 221, 225, 233
epithermal broadening, 315
estimator
    collision, 215
    point-based, 215
    surface-based, 215
    track length, 215
    volume-based, 214
execution, 20, 391
extended mesh, 224

FILE, 45, 127, 530
filtering, 216
fission model, 289
fission spectrum, 190
flux, 189
format, 21
frames, 35

g42so, 479, 480, 483–486, 488
gas production, 190
Gaussian Energy Brodening, 220
GEANT4, 187, 478–488
generalized perturbation theory, 378
geometry, 46–117

check, 103
combinatorial, 50, 65
surface-based, 50, 67
torus, 62
graphical display, 101
Green's functions, 402–408
grid, 211–213

help, 22
homogenization, 284–285

IDT importance maps, see biasing
IFP, 296, 307
importance, 240
inelastic threshold correction, 289
inserting files, 45, 127, 530
intersection, see operator

keep result, 295
Keep Result, 295
kinematic limits, 189
kinetic, 309
kinetics parameters, 296

lambda effective, 296
lattice, 50, 81–99, 142
lattice code
APOLLO, 126, 131
lattice of lattices, 86
launch, see execution
leakage, 120
library, 24

macroscopic reaction rate, 189
mass, 332
mesh tally, 224
migration area, 296
MORET, 489–490
multiple mode, 323
multiple scattering, 319

natural elements, 128
Nauchi, 296
neutron importance, 300
neutronics-shielding coupling, 323
TRIPOLI-NJOY coupling, 132
normal, 138, 179
nuclear transmutation, 195
number of fission neutrons per fission, 190

operator, 51, 70
intersection, 78
overriding, 71
truncation, 73
union, 78
output, 343–347, 383–386
overriding, see operator

parallelism, 391–398
particle, 286
path file, 21, 26
perturbation, 351–360
perturbations
concentration, 352
cross-section, 354
photon source, 190
photonuclear reactions, 197, 288
positron, 286, 287, 319
precursors, 309
production of particles, 190, 192

R2S, 468–476
random number, 296
reaction rate, 189–200
reflection, 121
cosine, 122
response, 189–210, 217
ROOT, 409–417
Rossi alpha, 296
rotation, 50, 69, 121
Russian roulette, 288

Scheduler, 491
score, 214–233
shape, 52–58
simulation, 286–328
six factor formula, 306
slowing down current, 306
Sobol, 104, 144
source, 138–188
distributed source, 140
external source, 179
multi-particle, 182
point source, 140
source function, 181
standard perturbation theory, 361
statistical weight, see importance
stopping power, 319
storage, 240, 348–350
sub-critical, 347
super-history simulation, 291
surface, 58–64, 118–119
surface restart, 170–172
SVT, 315

t4depletion, 443–467
t4path, 21, 26
T4toMED, 440
tally, see score
temperature interpolation, 290
thick-target bremsstrahlung, 319
time dependent, 309
transient, 309
translation, 121
truncation, see operator
type, 52

union, see operator
unit-base interpolation, 289
units, 22, 31

variance reduction, see biasing
verification and validation, 23
virtual volume, 51, 77, 142

visualizer, 420
volume, 51–99, 118–119, 329–331, 418–419
volume restart, 169

WCM, 315
weight windows, see biasing
Wielandt simulation, 291

XML, 296
XVGR, 383

ZA distribution of fission products, 190