

Link of the Geometry and Material routines between the EA-MC code and FLUKA-99

Sylvain David

ISN, IN2P3/CNRS/Université J. Fourier, Grenoble, France

Alfredo Ferrari, Yacine Kadi and Carlo Rubbia

CERN, Geneva, Switzerland

ABSTRACT

The FLUKACMCS program allows to run FLUKA-99 directly from the same Geometry.dat and Materials.Dat input files which describe the geometry and material composition in the EA-MC code package [1,2]. The source consists of 3 files: CMCS_GEOLINK.f, CMCS_MATLINK.f and CMCS_GEO.f which have been linked with FLUKA-99 [3a,3b] and can be run directly on the CVEET188 V-Class HP computer. Contrary to the existing HEXTOFL program [4,5], FLUKACMCS does not simply convert the EA-MC geometry and material input files into a FLUKA.INP file.

The FLUKACMCS program can run directly as a stand alone program for neutron-photon coupled transport for instance, or it can be called by EA-MC to carry out the high-energy particle transport calculation above 20 MeV. It creates a HISTORY.DAT file which can be re-used for subsequent calculations and other files of interest such as the energy deposition in the system.

Table of Content

<i>Table of Content</i>	2
<i>List of Figures</i>	3
<i>1 — Introduction</i>	4
<i>2 — Rounding zone definition</i>	4
<i>3 — Lattice treatment</i>	7
<i>4 — Material Link</i>	9
<i>5 — Tests Performed</i>	10
<i>6 — Conclusions</i>	13
<i>7 — References</i>	14
<i>APPENDIX A: Input description of the EA-MC geometry package (extracted from Ref. [2])</i>	15
<i>Input description of the EA-MC material composition (extracted from Ref. [1])</i>	20

List of Figures

- Figure 1:** *This is an example of boundary detection by the previous geometrical routine in the case of fuel pins arranged in a square lattice. 8*
- Figure 2:** *This is an example of boundary detection by the new geometrical routine in the case of fuel pins arranged in a multi-level square lattice. 9*
- Figure 3:** *Schematic view of the reactor system assembly (Source: Ansaldo Nucleare ref. [6]) 11*
- Figure 4:** *Windowless target in-vessel mechanical drawing (Source: Ansaldo Nucleare ref. [6]) 12*

1 — Introduction

The link between the EA-MC geometry [1,2] and FLUKA's [3a,3b] is carried out through the G1WR routine of FLUKA, which calls three routines of the EA-MC code: HOWFAR, MAKE_PATH and GET_NORM. HOWFAR and MAKE_PATH have been modified in order to have a consistent treatment of the boundary between EA-MC and FLUKA.

When transporting neutrons in the EA-MC code, a particle crossing a boundary is pushed on the other side in order to avoid the rounding problems at the surface. This treatment does not allow the transport of charged particles, mainly due to the scattering effects on the boundary. Thus, the surface treatment of EA-MC has been modified, as it is explain in this report.

2 — Rounding zone definition

For each surface, we define a rounding zone. The width of this region is the maximum of the position of the particle and the dimension of the shape, multiplied by a factor 10^{-12} . In order to avoid computer rounding problems when we are in the rounding zone, the region of the particle is defined by its direction. The particle is in the cell into which it is entering. This treatment is done by the CHCK_RND and CHCK_INSIDE routines.

The CHCK_RND routine corresponds to the old CHCK_INSIDE routine. Three parameters were added : ROUND_TEST, IPARRND and ROUNDING.

Subroutine CHCK_RND (p, igeoin, yesno, round_test, iparrnd, rounding)

input :

p : position of the particle
igeoin : element we are testing

output :

yesno : logical = true if we are in, false if you are out
round_test : logical = true if we are in the rounding zone false if we are not
iparrnd : index of the surface on which we are (only if round_test=true)
rounding : width of the rounding zone

If we are in the rounding zone, the routine CHCK_INSIDE defines the region of the particle.

Subroutine CHCK_INSIDE (p, igeoin, yesno, round_test, iparrnd, rounding)

input :

p : position of the particle

igeoin : element we are testing

round_test : logical = true if we are in the rounding zone false if we are not

iparrnd : index of the surface on which we are (only if round_test=true)

rounding : width of the rounding zone

output :

yesno : logical = true if we are in, false if we are out (definitive)

In order to find the region the particle is entering, we have to propagate the particle in its direction to go out of the rounding zone. In fact, we propagate the particle in the direction of the normal, to avoid the problems that may appear when the direction of the particle is parallel to the surface.

A new routine GET_NORM has been written which is described further. The particle is shifted in the direction of the normal by ten times the width of the surface, then, CHCK_RND is called assuming that TEST_ROUND will be false.

Subroutine GET_NORM (pinter, igeoin, ipar, dunorm)

input :

pinter : position on the surface

igeoin : element hit

ipar index of the surface hit

output :

dunorm : normal coordinates

The IPAR parameter identifies the surface; when we are on the surface of a tank, we have to know if we are on the half-sphere, on the cylinder or on the top plane. The IPAR parameter (output from CHCK_INSIDE) is the code for the surface and allows the normal calculation. This routine will be called by G1WR to allow the treatment of scattering for charged particles.

An important modification has been made in the GET_DIST routine of the EA-MC code, which computes the distance to the surface hit by the particle. The calculation of the distance is completely different if we are in the rounding

zone or not. If we are not, nothing has changed, but if we are in, we do not have to take into account the surface towards which we compute the distance.

Let us consider the example of a box and suppose we are on the surface $x=-Hx/2$ in the direction of X-axis. The distances to the two X planes can be positive. In the "classical" treatment, we take the minimum, but now we have to take the maximum value. The calculation of the distances to the Y and Z planes remains the same. Therefore, GET_DIST must take into account the rounding zone and the type of the surface that is crossed.

Subroutine GET_DIST (p, udir, igeoin, distw, ihit, ipar)

input :

p : position

udir : direction

igeoin : element to which we calculate the distance

output :

distw : distance to wall

ihit : logical true if igeoin is reached

ipar : type of the surface reached

These modifications allow the treatment of charged particles transport for simple shapes, further developments have been made for the lattice treatment.

3 — Lattice treatment

In the original version of the EA-MC code, the particle did not encounter the envelope of a lattice when it was going out. The MAKE_PATH routine overwrote the real position by the translated position, and the distance from the real position to the envelope of the lattice was never calculated. This approximation cannot be used for charged particles treatment. FLUKA has very strict tests at each boundary crossing, which forbids reaching a new region without crossing a surface before. This was the case when a particle was going out the envelope of a lattice, as shown in Figure 1.

This problem has been solved by saving each translation in a table, exactly like in the case of IPATH and it is clearly illustrated in Figure 2. The number of the translations NOTRN and the table ITRNHIST are output parameters of the MAKE_PATH routine.

Subroutine MAKE_PATH (p,udir,ipath,nopath,itrnhist,notrn)

input :

p : position

udir : direction

output

ipath : path

nopath : length of path

itrnhist : translation history

itrnhist(1,itrn) = ix

itrnhist(2,itrn) = iy

notrn = number of translations (itrn = 1,...,notrn)

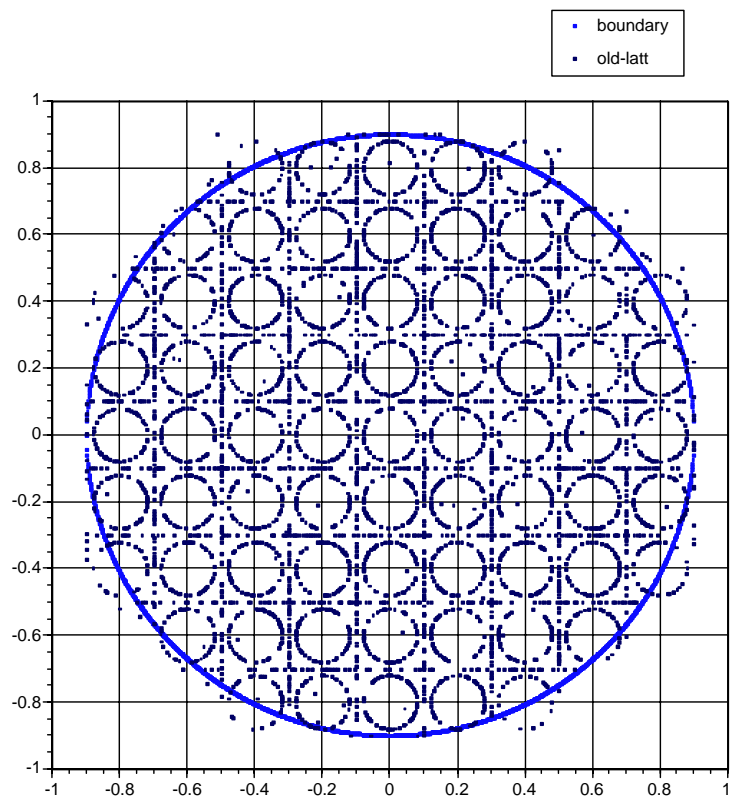


Figure 1: This is an example of boundary detection by the previous geometrical routine in the case of fuel pins arranged in a square lattice.

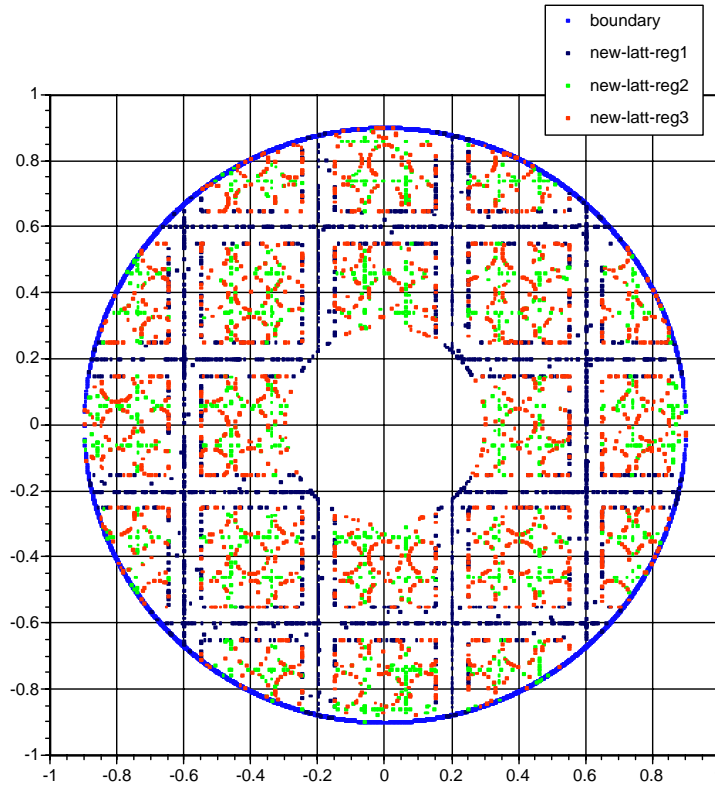


Figure 2: This is an example of boundary detection by the new geometrical routine in the case of fuel pins arranged in a multi-level square lattice.

In HOWFAR, we do the inverse translations to compute the distance from the particle to the envelope of the lattice. If this distance is smaller than the precedent, this distance is kept, and the intermediate position (after x translations) is saved as PGEOHIT, in order to compute the normal in G1WR.

4 — Material Link

The routine GET_MATERIALS has been modified in order to have a material description compatible with FLUKA requirements. Four FLUKA routines allow giving FLUKA the instruction without re-writing a FLUKA input file. These routines are MATCRD for the MATERIAL card, CMPCRD for the COMPOUND card, ASMCRD for the ASSIGNMAT card and LWMCRD for the LOW-MAT card. These routines fill the FLUKA material COMMON and are called from the GET_MATERIALS routine, which reads and decode the EA-MC material input file.

The program expects to read a NOTINDATABASE.NAME file, which is a list of nuclides that are not in the FLUKA database (for example ^{240}Pu). These nuclides are automatically replaced by lead (with a LOW-MAT call) for the neutron transport below 20 MeV (most of the time, we will not compute the transport of neutrons below 20 MeV since it will be carried out by EA-MC).

In order to treat electron transport, the program generates an AUTO.PEG file for each material defined by a MATCRD call. These files are written in a PEGDIR directory expected by the program. If this directory does not exist, a

warning message is written in the FORT.60 file. These automatic *.PEG files have to be modified manually when they refer to complex isotopic distribution (as in the case of the fuel mixture). They can be directly used when they refer to mixtures of natural element (fuel clad, lead-bismuth coolant, etc.).

5 — Tests Performed

The program has been extensively tested on DEC/Alpha and Convex HP V-Class operating system in stand alone mode for both neutron (EMF off) and coupled neutron-photon (EMF on) transport. A most detailed geometrical and material description of the Energy Amplifier Prototype as proposed by Ansaldo was used in the simulation (see Figures 3 and 4) and reported in Appendices A and B respectively.

The geometry considered consisted of some 750 material regions each containing about 20 nuclides on average. A complete coupled neutron-photon calculation starting with 600 MeV protons lasted on average 6.3 seconds per incident particle on the DEC/Alpha and 14.6 s/protons on the Convex HP V-Class. This corresponded to some 7×10^7 low-energy neutron interactions, with a production of about 10.6 photons and 16.5 neutrons per primary protons. More details are given in Appendix C which summarizes the output of the FLUKA-99 run of the test case in question.

The FLUKACMCS code was also run on the Convex HP V-Class in conjunction with an EA-MC low-energy neutron transport calculation for the test case mentioned above. A complete calculation starting with 600 MeV protons down to the 19.6 MeV cut-off energy lasted on average 0.04 seconds per incident particle, with a production of about 14.55 low-energy neutrons per primary protons. More details are given in Appendix D which summarizes the output of the FLUKA-99 run of the test case in question.

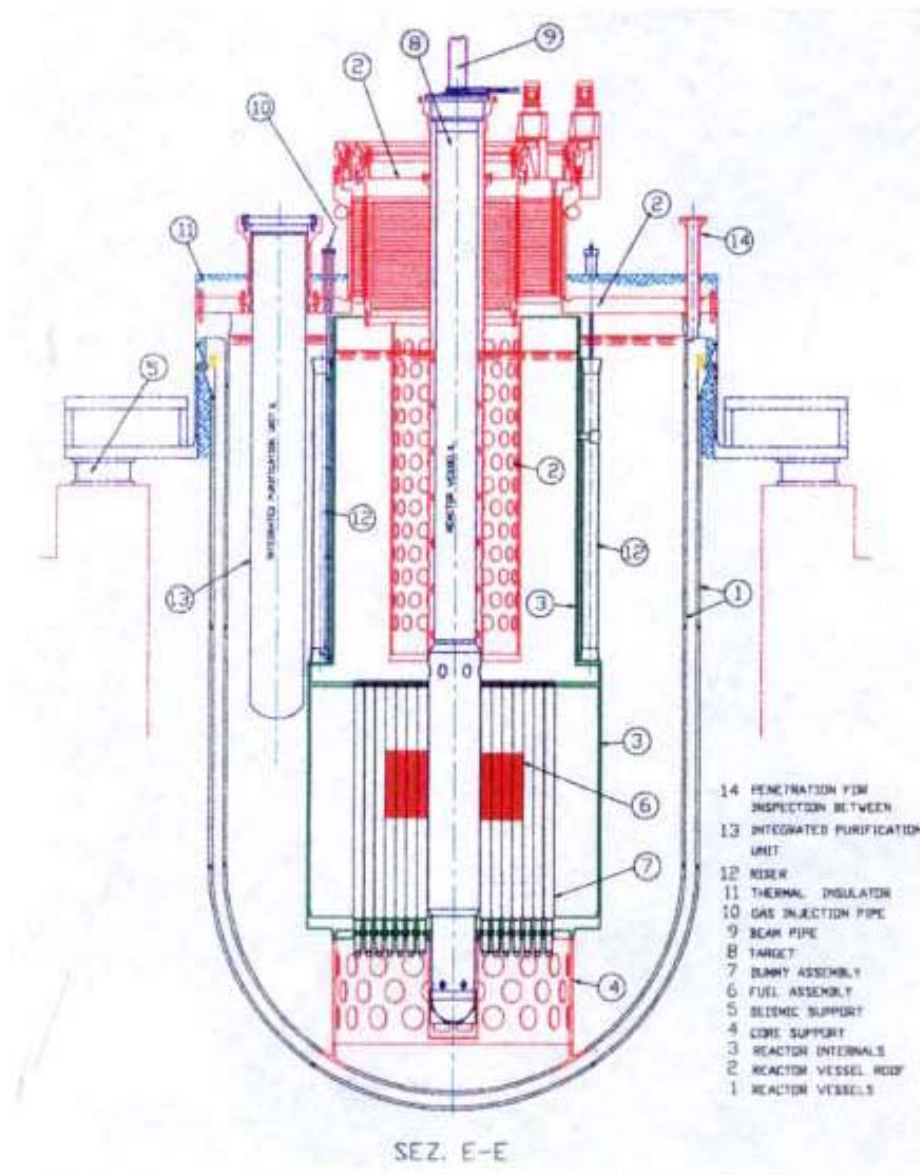


Figure 3: Schematic view of the reactor system assembly (Source: Ansaldo Nucleare ref. [6])

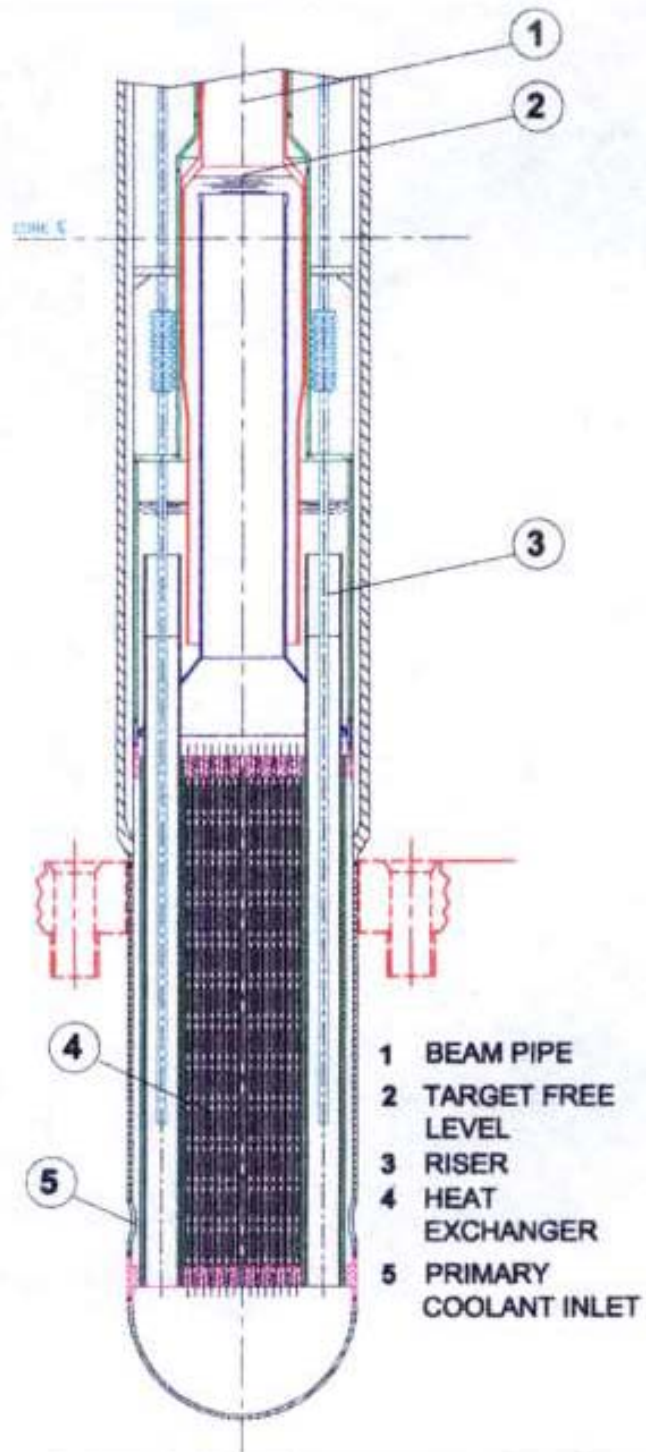


Figure 4: Windowless target in-vessel mechanical drawing (Source: Ansaldo Nucleare ref. [6])

6 — Conclusions

The EA-MC geometry package has been successfully implemented in FLUKA-99 and thoroughly tested both in stand alone mode and in conjunction with an EA-MC low-energy neutron transport calculation. It was also thoroughly tested for both neutron and charged particle transport problems including electromagnetic transport, for energies ranging from primary beam particles down to thermal neutron energies.

This implementation of the EA-MC geometry package will enable interested users to carry out detailed sensitivity studies with respect to the design of the spallation neutron target in the context of the European ADS Demonstration Facility, but also in the design of any other particular system involving spallation targets for any other application of particle accelerators.

7 — References

- [1] C. Rubbia et al., “Conceptual Design of a Fast Neutron Operated High Power Energy Amplifier”, CERN report CERN/AT/95-44 (ET), Geneva, 29th September 1995.
See also C. Rubbia, “A High Gain Energy Amplifier Operated with Fast Neutrons”, AIP Conference Proceedings 346, International Conference on Accelerator-Driven Transmutation Technologies and Applications, Las Vegas, July 1994.
For more details on the TRU’s incineration see also C. Rubbia et al., “A realistic Plutonium elimination scheme with Energy Amplifiers and Thorium-Plutonium Fuel” CERN report CERN/AT/95-53 (ET), Geneva, 12th December, 1995.
- [2] C. Rubbia, “A New Geometry Package for the EA-MC Code”, CERN internal report, CERN/LHC/EET 97-xxx (not published), Geneva, 27th October, 1997.
- [3a] A. Fassó et al., in “Intermediate Energy Nuclear Data: Models and Codes”, Proceedings of a Specialists' Meeting, Issy les Moulineaux (France) 30 May-1-June 1994, p. 271, published by OECD, 1994, and references therein.
A. Fassó, A. Ferrari, J. Ranft, P.R. Sala, G.R. Stevenson and J.M. Zazula, Nuclear Instruments and Methods A, 332, 459 (1993), also, CERN report CERN/TIS-RP/93-2/PP (1993).
see also A. Ferrari et al., “FLUKA-96 Manuel”, INFN, Milan, 1996.
- [3b] A. Ferrari and P.R. Sala, “The Physics of High Energy Reactions”, INFN, Milan, lecture given at the Workshop on Nuclear Reaction Data and Nuclear Reactors Physics, Design and Safety, International Centre for Theoretical Physics, Trieste, Italy, 15th April - 17th May, 1996.
- [4] F. Carminati, M. Embid and I. Goulas, “Description of the Program which Converts the Input Geometry of the EA Monte Carlo to a FLUKA-96 Geometry Input and a Graphic Visualisation Input”, CERN internal report, CERN/LHC/EET 97-009, Geneva, 4th July, 1997.
- [5] F. Carminati and C. Rubbia, “Conversion between HexGeo and FLUKA”, CERN internal report, CERN/LHC/EET 97-xxx (not published), Geneva, 14th July, 1997.
- [6] “Energy Amplifier Demonstration Facility Reference Configuration: Summary Report”, ANSALDO Nucleare, EA-B0.00-1-200 - Rev. 0, January 1999.

APPENDIX A:
Input description of the EA-MC geometry package
(extracted from Ref. [2])

1.- General rules. The geometry is defined by a concatenation of volumes, similar to the folder structure in the computers. The material element is located following the hierarchical chain defined by the chainig of such folders, called Holders. Holders can contain:

- (1) other Holders;
- (2) geometrical shapes, called Simple Elements;
- (3) geometrical Lattices, i.e. repetitive chains of elements or of Holders.

The entry point of the chain is defined by a Master Holder, usually the large volume containing all the elements.

Each simple element is characterized by a material code, which will be used by the programme to define the material composition of the components. The chain of cards must end with and END card. Comment cards have a * in the first column. A TITLE card is at the beginning.

2.-Simple elements. Each element is characterized by an element name and a shape (i.e. cylinder, rectangle etc.) and by as number of parameters, defining its centre and dimension and by the associated material code. If undefined, it is set to -1, normally handled as error. Numbers (Parameters) are free format and the denomination, to be used later on to identify the element, is within " ". Material index is defined anywhere in the field by -mat= value instruction, with format free integer value=val. Comment statement is allowed, following the comment mark !. The following elements have been implemented sofar:

2.1-CYLZ : a vertical cylinder, defined by its centre (X_o, Y_o, Z_o) its radius (R) and its height (H).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
CYLZ	"name"	Xo	Yo	Zo	R	1/2 H	

2.2- CYLY : a along the y-axis cylinder, defined by its centre (X_o, Y_o, Z_o) its radius (R) and its height (H).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
CYLY	"name"	Xo	Yo	Zo	R	1/2 H	

2.3- CYLX: a along the x-axis cylinder, defined by its centre (X_o, Y_o, Z_o) its radius (R) and its height (H).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
CYLX	“name”	Xo	Yo	Zo	R	1/2 H	

2.4- RECT: a rectangular parallelepiped, defined by its centre (X_o, Y_o, Z_o) its radius (R) and its 3 half sizes, (DX, DY, DZ).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
RECT	“name”	Xo	Yo	Zo	DX	DY	DZ

2.5- SPHE: a sphere, defined by its centre (X_o, Y_o, Z_o) its radius (R) .

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
SPHE	“name”	Xo	Yo	Zo	R		

2.6- TANK: A z-oriented vertical cylinder, closed below by a spherical shape (window), defined by its centre (X_o, Y_o, Z_o) which is the centre of the sphere its radius (R) and the full height of the cylindric part (H).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
TANK	“name”	Xo	Yo	Zo	R	H	

2.7- RNGZ: A z-oriented coaxial ring structure limited by two cylinders, defined by its centre (X_o, Y_o, Z_o) its inner and outer radii (RIN,ROUT) and the half height of the cylindric part (H).

LABEL	Denom.	Par 1	Par 2	Par 3	Par 4	Par 5	Par 6
RNGZ	“name”	Xo	Yo	Zo	RIN	ROUT	H

3.- Holders. The Holders are used to contain either Simple Elements, other Holders or Lattices. Each Holder is characterized by a volumic shape, which is the volume in which all elements are to be contained. It represents also the default containment. Objects which exceed the size of the Holder are truncated to the volume defined by the Holder. There are no parameters in the Holder but only denominations, which have to be bracketed, namely between “”. The name of the volumic shape, which must be defined beforehand, cannot be the same as the name of the Holder. A practical way is to use small letters for all simple volumes and a Capitalized name for the corresponding Holder. For the moment we have implemented the following options:

3.1- HOLD: a Holder in which physically separate elements, (Holders Lattices and simple shapes) can be introduced. The program will first check containment in the volumic shape (VOL0) and then in all the subsequent, already defined, elements (VOL1, VOL2..... VOLn). The last acceptable volume will be retained when constructing the path. The default volume id VOL0. If the last accepted element is a simple volume with a -imat denomination, the path making process will stop (end of path) and the material code will be the

one associated with this last element. This is an effective procedure to generate out of geometrical shaper bodies with appropriate material assignments.

LABEL	ret. vol.	Ele 1	Ele 2	Ele 3	Ele k	Ele n
HOLD	"VOL0"	"VOL1"	"VOL2"	"VOL3"	"VOLk"	"....."	"VOLn"

3.2- CONT: a Holder similar to HOLD but in which physically subsequently contained elements, (Holders Lattices and simpe shapes) can be introduced. The program will first check containement in the volumic shape (VOL0) and then in the subsequent, already defined, elements (VOL1, VOL2..... VOLn). The first non accepted volume will stop the scan and the previous, last accepted will be retained when constructing the path. The default volume id VOL0. If the last accepted element is a simple volume with a -imat denomination, the path making process will stop (end of path) and the material code will be the one associated with this last element. This is an effective procedure to generate out of geometrical shaper bodies with appropriate material assignments. The difference between HOLD and CONT is to be found in computing speed since the procedure is stopped as soon as the volume is not containing the point.

LABEL	ret. vol.	Ele 1	Ele 2	Ele 3	Ele k	Ele n
CONT	"VOL0"	"VOL1"	"VOL2"	"VOL3"	"VOLk"	"....."	"VOLn"

3.3- MAIN: It is used to define the beginning of the path. It has only one volume denomination, the preceedingly defined, presumably Holder, from which to start the path.

LABEL	start v.						
MAIN	"VOL0"						

Some examples are here mandatory. Suppose we want to construct a spallation target, made of a large tank made of -imat=1, filled with -imat=2 and in which a target tube of -imat=3 is introduced, empty inside, -imat=4. The procedure will be the following:

TITLE " An example of a spallation target"

First we define a larger outer box 10 m tall, 2 m on the sides , -imat=0 which contains the whole device:

RECT "all" 0. 0. 0. 1.0 1.0 5.0 -imat=0 ! big box z=from -5 to 5

Then we define the 4 geometrical shapes which correspond to the walls of the two tanks and the corresponding materials. The centre of the sphere is centered accordingly:

TANK "wall4" 0. 0. -3.0 1.0 8.0 -imat=1! mat is wall mat

TANK "wall3" 0. 0. -3.0 0.95 8.0 -imat=2 ! mat is tank fill

```
TANK "wall2" 0. 0. 0.0 0.1 5.0 -imat=3      ! mat is wall mat
TANK "wall1" 0. 0. 0.0 0.09 5.0 -imat=4     ! mat is void mat
```

Then we make holders which define the actual hardware elements:

```
CONT "Outertank" "wall4" "wall3"          ! outer container
CONT "Beampipe" "wall2" "wall1"          ! outer container
```

Finally we insert the elements in the device:

```
CONT "SpallTarget" "all" "Outertank" "Beampipe" ! done
```

In order to start the path we must define a MAIN element:

```
MAIN "SpallTarget"      ! start from here
END                      ! terminator
```

4.- Lattices: Inside the structure it may be necessary to introduce a periodic structure, like the fuel bars in the fuel volume etc. For this reason it is possible to generate a lattice of these structures, namely a repetitive chain of elements (Holders) in which in turn additional structure, including smaller lattices can be introduced. Suppose for instance that the fuel is compounded of fuel bundles, each made of fuel rods. We can generate this using twice the lattice approach, first for the bundles and next for the fuel rods. At each lattice command the co-ordinate system is translated to the centre of the simple volume associated with the lattice. Therefore all subsequent elements must be referred to this new origin co-ordinates. For the moment the following lattice command has been implemented:

4.1 - RLAT: a rectangular lattice. The command has both 2 element names and 3 parameters, in subsequent order. The lattice is characterized by a Holder VOL1 contained in a retaining volume VOL0. As already pointed out the reference axis of the VOL1 elements and subsequents are changed and are the centre of VOL0. Parameters PX and PY represent the periodicity of the lattice and must be not larger than the maximum sizes of the presumably RECT representing VOL0. Periodicity is attained shifting by $NINT(X/PX)*PX$ the point X and by $NINT(Y/PY)*PY$ the point Y. VOL0 is also the default volume if VOL1 is smaller than VOL0. Transformation of axis occurs after VOL0 and before VOL1. Hence, VOL1 has to be compatible with the transformed geometry.

LABEL	NAME	Ret. Vol.	holder	Par 1	Par 2	Par 3	
RLAT		"VOL0"	"VOL1 "	PX	PY	0.0	

Assume for instance that we want to insert in the previous example a heat exchanger made of a ring (RNGZ) filled with tubes of -imat=1 in which a cooling liquid with -imat=5 is circulating.

We define first the outer envelope of the heat exchanger:

```
RNGZ "heatExchw" 0. 0. 4. 0.2 0.7 0.5 -imat=1 ! imat=mat outer walls
```

We also define the material -imat=2 of the inside of the heat exchanger and the CONT for both, with the following total sequence:

```
RNGZ "heatExchw" 0. 0. 4. 0.2 0.7 0.5 -imat=1 ! imat=mat outer walls
```

```
RNGZ "heatExchin" 0. 0. 4. 0.25 0.65 0.45 -imat=2 ! imat=mat inside
```

```
CONT "HeatExch" "heatExchw" "heatExchin" ! Heat exchanger
```

The heat exchanger is made of tubes separated by about 1 cm, therefore, we need a holding volume for the tubes, with default -imat=2, which is the basis for the lattice matrix, and the tube and the coolant:

```
RECT "tubeholder" 0. 0. 0. 0.05 0.05 0.45 -imat=2 ! lattice matrix
```

```
CYLZ "tube" 0. 0. 0. 0.03 0.45 -imat=1 ! mat=walls
```

```
CYLZ "coolant" 0. 0. 0. 0.025 0.45 -imat=5 ! mat=coolant
```

With all that, we make the holder for the lattice structure:

```
CONT "ExchElem" "tubeholder" "tube" "coolant" ! cooling element
```

We are now ready to introduce the lattice definition:

```
RLAT "CoolStruct" "heatExchin" "ExchElem" 0.05 0.05 0.0
```

And the CONT card must be modified as follows:

```
CONT "HeatExch" "heatExchw" "CoolStruct" ! Heat exchanger
```

Input description of the EA-MC material composition (extracted from Ref. [1])

CARD FORMAT FILE "Materials.dat"

ELEMENTS CAN HAVE DIFFERENT FORMATS:

"PB" = natural element mixture ,single letter allowed for "U" etc
(needs no extra blank) : ALL CAPITALS

"PB-208" = isotope

"AM-242*" = isomer (only first)

GENERAL RULES:

space is separator

character strings between " marks i.e. "string"

numbers on free format

maximum length of all names (strings) = 15 characters

CHEM "comp name" "ELE(1)" chemical weight(1) "ELE(2)"
chemical weight(2) ..."ELE(n)" chemical weight(n) -dens=10.0 [-smea=0.95
-temp=700.0]

chemical weight= multiplicity in chemical formula

default smear=1 temp=300

MIXT "mix name" "ELE(1)" mass fraction(1) "ELE(2)" mass fraction(2)
..."ELE(n)" mass fraction(n) -dens=10.0 [-smea=0.95 -temp=700.0]

MATE "mat name" JMAT "comp/mix name(1)" volumic fraction(1)
"comp/mix name(2)" volumic fraction(2)"comp/mix name(n)"
volumic fraction(n)

COPY "mat name to be copied" "new mat name" JMAT(of new material)

END last card

Material composition input listing of the configuration considered in the test case:

```

CHEM "Void" "PB" 1. -dens=0.0001 -smea=1.0 -temp=900.0
CHEM "DU" "U-234" .000043 "U-235" .005238 "U-236" .000050 "U-238" .994669 "O" 2. -dens=10.407
-smea=1.0 -temp=900.0
CHEM "DNp" "NP-237" 1. "O" 2. -dens=10.407 -smea=1.0 -temp=900.0
CHEM "DPu" "PU-238" .002800 "PU-239" .708407 "PU-240" .249273 "PU-241" .026509 "PU-242"
.013011 "O" 2. -dens=10.407 -smea=1.0 -temp=900.0
CHEM "DAm" "AM-241" 1. "O" 2. -dens=10.407 -smea=1.0 -temp=900.0
MIXT "Pb-Bi" "PB" 0.45 "BI" 0.55 -dens=10.23 -smea=1. -temp=900.0
MIXT "AISI316" "FE" 0.6415 "CR" 0.17 "NI" 0.125 "MO" 0.0275 "MN" 0.02 "SI" 0.01 "TI" 0.006
-dens=7.87 -smea=1.0 -temp=900.0
MIXT "9Cr1Mo" "FE" 0.88285 "CR" 0.09 "MO" 0.01 "C" 0.0015 "MN" 0.00455 "P" 0.0003 "S"
0.0003 "SI" 0.0105 -dens=7.87 -smea=1.0 -temp=900.0
MIXT "Air" "N" 0.75558 "O" 0.23159 "AR" 0.01283 -dens=0.001225 -smea=1.0 -temp=900.0
MIXT "InertGas" "AR" 1.0 -dens=0.0017837 -smea=1.0 -temp=900.0
MIXT "OrgCool" "C" 0.88 "H" 0.12 -dens=0.8 -smea=1.0 -temp=900.0
MATE "VesRoof" 1 "Air" 1.0
MATE "CovPlate" 2 "9Cr1Mo" 1.0
COPY "CovPlate" "SafeVes" 3
MATE "VesGap" 4 "InertGas" 1.0
COPY "CovPlate" "ReacVes" 5
MATE "PrimCool" 6 "Pb-Bi" 1.0
COPY "CovPlate" "TargVes" 7
COPY "PrimCool" "TargCool" 8
MATE "TargHX" 9 "Pb-Bi" 0.5 "9Cr1Mo" 0.5
COPY "CovPlate" "TargInj" 10
COPY "VesGap" "TargGas" 11
COPY "CovPlate" "BeamTube1" 12
COPY "CovPlate" "BeamTube2" 13
MATE "SpalTarg1" 14 "Void" 1.0
COPY "CovPlate" "SpalTube" 15
COPY "PrimCool" "SpalTarg2" 16
MATE "UpFlowTube" 17 "Pb-Bi" 0.5 "9Cr1Mo" 0.5
MATE "UpFlowCov" 18 "Pb-Bi" 0.8 "9Cr1Mo" 0.2
MATE "LoFlowTube" 19 "Pb-Bi" 0.1 "9Cr1Mo" 0.9
MATE "LoFlowCov" 20 "Pb-Bi" 0.6 "9Cr1Mo" 0.4
COPY "CovPlate" "HXWall" 21
COPY "PrimCool" "HXCool1" 22
COPY "CovPlate" "HXTubes" 23
MATE "HXCool21" 24 "OrgCool" 1.0 ! secondary coolant axial zone 1
COPY "CovPlate" "PuriWall" 25
COPY "PrimCool" "PuriCool" 26 ! purification unit primary coolant (Pb-Bi)
COPY "CovPlate" "MHDWall" 27 ! MHD units support walls
COPY "CovPlate" "MHD Tubes" 28 ! MHD tubes
MATE "MHDCool" 29 "Pb-Bi" 0.93 "InertGas" 0.07 ! MHD unit coolant (93% Pb-Bi, 7% Argon)
COPY "CovPlate" "CoreProt" 30 ! core lateral protection
MATE "CoreTop" 31 "Pb-Bi" 0.6 "9Cr1Mo" 0.4 ! core top support plate
COPY "CoreTop" "CoreBot" 32 ! core bottom support plate
COPY "CovPlate" "TopRef1" 33 ! top reflector core assemblies
MATE "CoreCool" 34 "Pb-Bi" 0.9924 "9Cr1Mo" 0.0076 ! core primary coolant (central pin
diluted)
COPY "CovPlate" "TopPlug" 35 ! fuel pin top plug
MATE "InnerVoid" 36 "InertGas" 1.0 ! fuel pin inner void + gap
COPY "CovPlate" "IntPlug" 37 ! fuel pin intermediate plug
MATE "Grid1" 38 "Pb-Bi" 0.5 "9Cr1Mo" 0.5 ! fuel pin top support grid
COPY "CovPlate" "Grid1clad" 39 ! fuel pin top support grid walls

```

COPY "CovPlate" "TopPlenum" 40 ! fuel pin top plenum chamber
 COPY "CovPlate" "blkt1clad" 41 ! fuel pin top Uranium blanket walls
 MATE "TopBlankt" 42 "DU" 1.0 ! fuel pin top Uranium blanket pellet
 COPY "CovPlate" "fuel1clad" 43 ! fuel pin pellets 1-10 walls
 MATE "Fuel1" 44 "DU" 0.794858 "DNp" 0.000047 "DPu" 0.200346 "DAm" 0.004749 ! fuel pin pellets 1-10
 COPY "CovPlate" "fuel2clad" 45 ! fuel pin pellets 11-20 walls
 COPY "Fuel1" "Fuel2" 46 ! fuel pin pellets 11-20
 COPY "CovPlate" "fuel3clad" 47 ! fuel pin pellets 21-22 walls
 COPY "Fuel1" "Fuel3" 48 ! fuel pin pellets 21-22
 COPY "Grid1" "Grid2" 49 ! fuel pin top mixing grid
 COPY "CovPlate" "grid2clad" 50 ! fuel pin mixing grid pellets 23-25 walls
 COPY "Fuel1" "grid2mat" 51 ! fuel pin mixing grid pellets 23-25
 COPY "CovPlate" "fuel4clad" 52 ! fuel pin pellets 26-30 walls
 COPY "Fuel1" "Fuel4" 53 ! fuel pin pellets 26-30
 COPY "CovPlate" "fuel5clad" 54 ! fuel pin pellets 31-40 walls
 COPY "Fuel1" "Fuel5" 55 ! fuel pin pellets 31-40
 COPY "CovPlate" "fuel6clad" 56 ! fuel pin pellets 41-47 walls
 COPY "Fuel1" "Fuel6" 57 ! fuel pin pellets 41-47
 COPY "CovPlate" "fuel7clad" 58 ! fuel pin pellets 48-57 walls
 COPY "Fuel1" "Fuel7" 59 ! fuel pin pellets 48-57
 COPY "CovPlate" "fuel8clad" 60 ! fuel pin pellets 58-62 walls
 COPY "Fuel1" "Fuel8" 61 ! fuel pin pellets 58-62
 COPY "Grid1" "Grid3" 62 ! fuel pin bottom mixing grid
 COPY "CovPlate" "grid3clad" 63 ! fuel pin mixing grid pellets 63-65 walls
 COPY "Fuel1" "grid3mat" 64 ! fuel pin mixing grid pellets 63-65
 COPY "CovPlate" "fuel9clad" 65 ! fuel pin pellets 66-67 walls
 COPY "Fuel1" "Fuel9" 66 ! fuel pin pellets 66-67
 COPY "CovPlate" "fuel10clad" 67 ! fuel pin pellets 68-77 walls
 COPY "Fuel1" "Fuel10" 68 ! fuel pin pellets 68-77
 COPY "CovPlate" "fuel11clad" 69 ! fuel pin pellets 78-87 walls
 COPY "Fuel1" "Fuel11" 70 ! fuel pin pellets 78-87
 COPY "CovPlate" "blkt2clad" 71 ! fuel pin bottom Uranium blanket walls
 COPY "TopBlankt" "BotBlankt" 72 ! fuel pin bottom Uranium blanket pellet
 COPY "CovPlate" "BotPlenum" 73 ! fuel pin bottom plenum chamber
 COPY "CovPlate" "BotPlug" 74 ! fuel pin bottom plug
 COPY "CovPlate" "blanwall" 75 ! wall of the side reflector core assemblies
 COPY "CovPlate" "blantube" 76 ! central supporting metallic pin of the side reflector core assemblies
 COPY "CovPlate" "refwall2" 77 ! wall of the bottom reflector core assemblies
 COPY "CovPlate" "BotRef" 78 ! central supporting metallic pin of the bottom reflector
 COPY "HXCool21" "HXCool22" 79 ! secondary coolant axial zone 2
 COPY "HXCool21" "HXCool23" 80 ! secondary coolant axial zone 3
 COPY "HXCool21" "HXCool24" 81 ! secondary coolant axial zone 4
 COPY "HXCool21" "HXCool25" 82 ! secondary coolant axial zone 5
 COPY "HXCool21" "HXCool26" 83 ! secondary coolant axial zone 6
 COPY "HXCool21" "HXCool27" 84 ! secondary coolant axial zone 7
 COPY "HXCool21" "HXCool28" 85 ! secondary coolant axial zone 8
 COPY "CovPlate" "HexWall" 86
 END