**TECHNICAL REPORT**
**IGE–293**

# A USER GUIDE FOR TRIVAC VERSION4

A. Hébert

Institut de génie nucléaire
Département de génie mécanique
École Polytechnique de Montréal
December 2, 2019

## Copyright Notice for TRIVAC

The development of TRIVAC is financially supported, directly or indirectly, by various organizations including École Polytechnique de Montréal, Hydro–Québec and the Hydro–Québec chair in nuclear engineering, the Natural Science and Engineering Research Council of Canada (NSERC), Atomic Energy of Canada limited (AECL) and the CANDU Owners Group (COG). 

## Acknowledgments

The computer code TRIVAC results from a concerted effort made at École Polytechnique de Montréal. The main authors of this report would therefore like to express their thanks to École Polytechnique de Montréal for its support along the years as well as to the graduate students and research associates which have contributed to the development of TRIVAC along the years. Finally, the TRIVAC team would never have survived without the financial support of the Natural Science and Engineering Research Council of Canada (NSERC), Hydro–Québec and Atomic Energy of Canada Limited (AECL).

SUMMARY

TRIVAC is a computer code intended to compute the neutron flux in a fractional or in a full core representation of a nuclear reactor. Interested readers can obtain fundamental informations about full-core calculations in Chapter 5 of Ref. 1. The multigroup and multidimensional form of the diffusion equation or simplified $P_n$ equation is first discretized to produce a consistent matrix system. This matrix system is subsequently solved using iterative techniques (inverse or preconditioned power method with ADI preconditioning) and sparse matrix algebra techniques (triangular factorization). The actual implementation of TRIVAC allows the discretization of 1-D geometries (slab and cylindrical), 2-D geometries (Cartesian, cylindrical and hexagonal) and 3-D geometries (Cartesian and hexagonal). Many discretization techniques are available, including mesh corner or mesh centered finite difference methods, collocation techniques of various order and finite element methods based on a primal or dual functional formulation. TRIVAC also permits the equations of the generalized perturbation theory (GPT) to be solved as fixed source eigenvalue problems. Finally, several implicit numerical schemes are available for the solving of space-time neutron kinetics problems.

The execution of TRIVAC is controlled by the generalized GAN driver.[2] It is modular and can be interfaced easily with other production codes.

# Contents

# List of Figures

# List of Tables

# 1 INPUT DATA SPECIFICATIONS

## 1.1 Syntactic rules for input data specifications

The input data to any module is read in free format using the subroutine `REDGET`. The rules for specifying the input data are therefore given in this section. The users guide was written using the following conventions:

- the parameters surrounded by single square brackets '[ ]' denote an optional input;
- the parameters surrounded by double square brackets '[[ ]]' denote an optional input which may be repeated as many times as desired;
- the parameters in braces separated by vertical bars '{ | | }' denote a choice of input where (one and *only* one is mandatory);
- the parameters in **bold face** and in brackets '( )' denote an input structure;
- the parameters in italics and in brackets with an index '(*data*(i), i=1,n)' denote a set of n inputs;
- the words using the typewriter font are character constants `keywordS` used as keywords;
- the words in italics are user defined variables, they should be lower case and are of type integer (starting with *i* to *n*) and real (starting with *a* to *h* or *o* to *z*) or of type character in uppercase *CHARACTER*.

## 1.2 The global input structure

TRIVAC is built around the GAN generalized driver.[2] Input data must therefore follow the calling specifications given below:

Table 1: Structure **(TRIVAC)**

```
[ LINKED_LIST [[ NAME1 ]] ; ]
[ XSM_FILE [[ NAME2 ]] ; ]
[ SEQ_BINARY [[ NAME3 ]] ; ]
[ SEQ_ASCII [[ NAME4 ]] ; ]
[ MODULE [[ NAME5 ]] ; ]
[[ (specif) ]]
END: ;
```

where

| | |
|---|---|
| *NAME1* | `Character*12` name of a LCM object. |
| *NAME2* | `Character*12` name of an XSM file. |
| *NAME3* | `Character*12` name of a sequential binary file. |
| *NAME4* | `Character*12` name of a sequential ASCII file. |
| *NAME5* | `Character*12` name of a module. |
| **(specif)** | Input specifications for a single module. Specifications for TRIVAC modules will be given in the following sections. |

The input data always begin with the declaration of each LCM object, XSM file, sequential (binary or ASCII) file that will be required by the following modules. This is followed by the declaration of the modules actually used in the input data deck. The following data describe a sequence of module calls, in the format of the GAN generalized driver. As indicated in Fig. 1, the modules communicate with each other throught LCM objects or XSM files whose specifications are given in section 2. The TRIVAC user generally have the choice to declare its data structures as `LINKED_LIST` to reduce CPU time resources or as `XSM_FILE` to reduce CPU memory resources.

The input data always end with a call to the `END:` module.

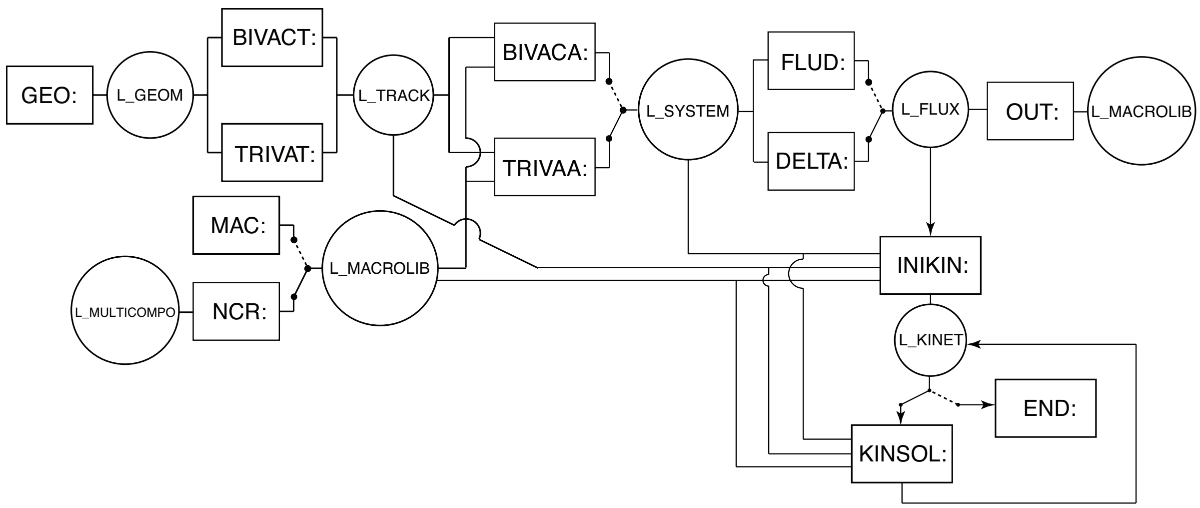


Figure 1: The TRIVAC modular approach.

### 1.3 The `GEO:` module

The `GEO:` module is used to create or modify a geometry. The geometry definition module in TRIVAC permits all the characteristics (coordinates, material mixture type indices and boundary conditions) of a simple or complex geometry to be specified. The method used to specify the geometry is independent of the discretization module to be used subsequently. Each geometry is represented by a name (`character*12`) and is saved in a LCM object or an XSM file under its given name. It is always possible to modify a given existing geometry or copy it into a neighbouring LCM object under a new name. The calling specifications are:

Table 2: Structure **(GEO:)**

| |
|---|
| { *GEOM1* := `GEO:` :: **(geo_data1)** \| <br> *GEOM1* := `GEO:` { *GEOM1* \| *GEOM2* } :: **(geo_data2)** } |

where

*GEOM1* `character*12` name of the LCM object (type `L_GEOM`) that will contain the geometry.

*GEOM2* `character*12` name of a LCM object (type `L_GEOM`) containing the existing geometry. The type and all the characteristics of *GEOM2* will be copied onto *GEOM1*.

**(geo_data1)** structure describing the characteristics of a new geometry (see Sect. 1.3.1).

**(geo_data2)** structure describing the change to the characteristics of an existing geometry (see Sect. 1.3.1).

### *1.3.1 Data input for module* `GEO:`

Structures **(geo_data1)** and **(geo_data2)** serve to define the principle components of a geometry (dimensions, materials, boundary conditions):

Table 3: Structure **(geo_data1)**

{ `HOMOGE` | `CAR1D` *lx* | `TUBE` *lr* | `SPHERE` *lr* | `CAR2D` *lx* *ly* | `TUBEZ` *lr* *lz* | `CAR3D` *lx* *ly* *lz* |
`HEX` *lh* | `HEXZ` *lh* *lz* }
[ `EDIT` *iprint* ]
**(descBC)**
**(descMC)**
**(descPOS)**
;

Table 4: Structure **(geo_data2)**

[ `EDIT` *iprint* ]
**(descBC)**
**(descMC)**
**(descPOS)**
;

where

`HOMOGE` infinite homogeneous geometry.

`CAR1D` one dimensional plane geometry (infinite slabs).

`TUBE` cylindrical geometry (infinite tubes or cylinders).

`SPHERE` spherical geometry (concentric spheres).

`CAR2D` two-dimensional cartesian geometry.

`TUBEZ` polar geometry $(R - Z)$.

`CAR3D` three-dimensional cartesian geometry.

`HEX` two-dimensional hexagonal geometry.

`HEXZ` three-dimensional hexagonal geometry.

*lx* number of subdivisions along the $X$ axis (before mesh-splitting).

*ly* number of subdivisions along the $Y$ axis (before mesh-splitting).

*lz* number of subdivisions along the $Z$ axis (before mesh-splitting).

*lr* number of cylinders or spherical shells (before mesh-splitting).

*lh* number of hexagons in an axial plane (including the virtual hexagons).

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `GEO:`. =0 for no print; =1 for minimum printing (default value); =2 for printing the geometry state vector.

**(descBC)** structure allowing the boundary conditions surrounding the geometry to be treated.

**(descMC)** structure allowing material mixtures to be associated with a geometry.

**(descPOS)** structure allowing the coordinates of a geometry to be described.

The inputs corresponding to the **(descBC)** structure are the following:

Table 5: Structure **(descBC)**

[ `X-` { `VOID` | `REFL` | `DIAG` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO`
     | `CYLI` | `ACYL` { *albedo* | *icode* } } ]
[ `X+` { `VOID` | `REFL` | `DIAG` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO`
     | `CYLI` | `ACYL` { *albedo* | *icode* } } ]
[ `Y-` { `VOID` | `REFL` | `DIAG` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO`
     | `CYLI` | `ACYL` { *albedo* | *icode* } } ]
[ `Y+` { `VOID` | `REFL` | `DIAG` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO`
     | `CYLI` | `ACYL` { *albedo* | *icode* } } ]
[ `Z-` { `VOID` | `REFL` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO` } ]
[ `Z+` { `VOID` | `REFL` | `TRAN` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO` } ]
[ `R+` { `VOID` | `REFL` | `ALBE` { *albedo* | *icode* } | `ZERO` | } ]
[ `HBC` { `S30` | `SA60` | `SB60` | `S90` | `R120` | `R180` | `SA180` | `SB180` | `COMPLETE` }
{ `VOID` | `REFL` | `SYME` | `ALBE` { *albedo* | *icode* } | `ZERO` } ]
[ `RADS` [ `ANG` ] *nrads* (*xrad*(ir), *rrad*(ir) [, *ang*(ir) ], ir=1,nrads ) ]

where

`X-` negative $X$ side.

`Y-` negative $Y$ side.

`Z-` negative $Z$ side.

`X+` positive $X$ side.

`Y+` positive $Y$ side.

`Z+` positive $Z$ side.

`R+` side surrounding cylinders or spheres.

`HBC` side surrounding a hexagonal geometry.

`VOID` the side under consideration has a zero incoming current boundary condition.

`REFL` the side under consideration has a reflective boundary condition.

`DIAG` the side under consideration is external to a diagonal axis of symmetry.

`TRAN` the side under consideration is connected to the opposite side of the domain. This option permits a translation condition to be treated.

`SYME` the side under consideration is next to an axial axis of symmetry. (symmetric with respect to the central axis of the last row of volumes). The `SYME` condition can also be used in hexagonal geometry, but only with `S30` and `SA60` symmetries.

`ALBE` the side under consideration has an arbitrary albedo to be specified.

*albedo* geometrical albedo corresponding to the boundary condition `ALBE` ($albedo \geq 0.0$).

*icode* index of a physical albedo corresponding to the boundary condition `ALBE`. The numerical values of the physical albedo are supplied by the module `MAC:`.

`ZERO` the side under consideration has a zero flux boundary condition.

`CYLI` the side under consideration has a zero incoming current boundary condition with a circular correction applied on the Cartesian boundary. This option is only available in the $X$–$Y$ plane for `CAR2D` and `CAR3D` geometries defined for TRIVAC full–core calculations.

`ACYL` the side under consideration has an arbitrary albedo with a circular correction applied on the Cartesian boundary. This option is only available in the $X$–$Y$ plane for `CAR2D` and `CAR3D` geometries defined for TRIVAC full–core calculations.

`S30` hexagonal symmetry of one twelfth of an assembly (see Fig. 2).

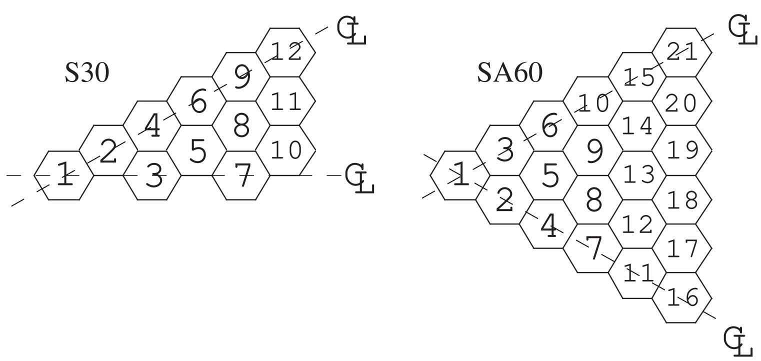


Figure 2: Hexagonal geometries of type S30 and SA60

`SA60` hexagonal symmetry of one sixth of an assembly of type A (see Fig. 2).

`SB60` hexagonal symmetry of one sixth of an assembly of type B (see Fig. 3).

`S90` hexagonal symmetry of one quarter of an assembly (see Fig. 3).

`R120` hexagonal symmetry of one third of an assembly (rotational symmetry) (see Fig. 4).

`R180` rotational symmetry of a half assembly (see Fig. 4).

`SA180` hexagonal symmetry of half a type A assembly (see Fig. 5).

`SB180` hexagonal symmetry of half a type B assembly (see Fig. 6).

`COMPLETE` complete hexagonal assembly (see Fig. 7).

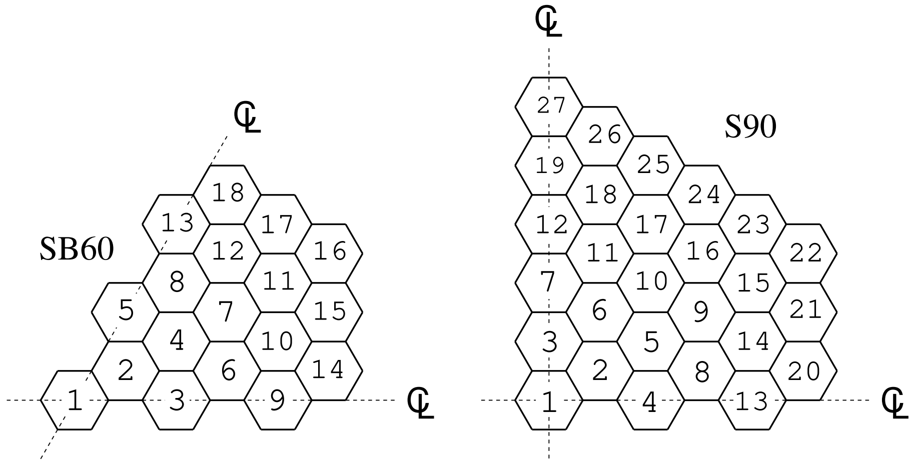


Figure 3: Hexagonal geometries of type SB60 and S90

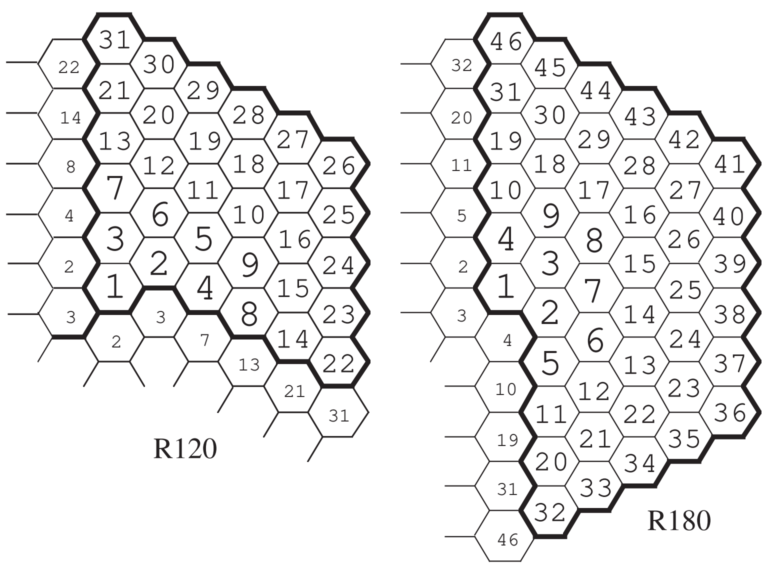


Figure 4: Hexagonal geometries of type R120 and R180

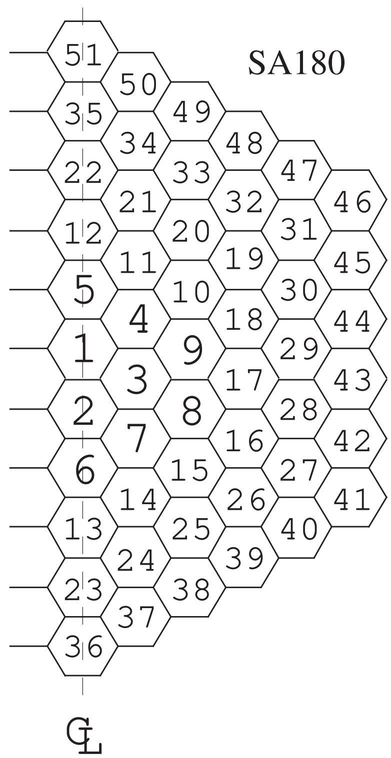


Figure 5: Hexagonal geometry of type SA180

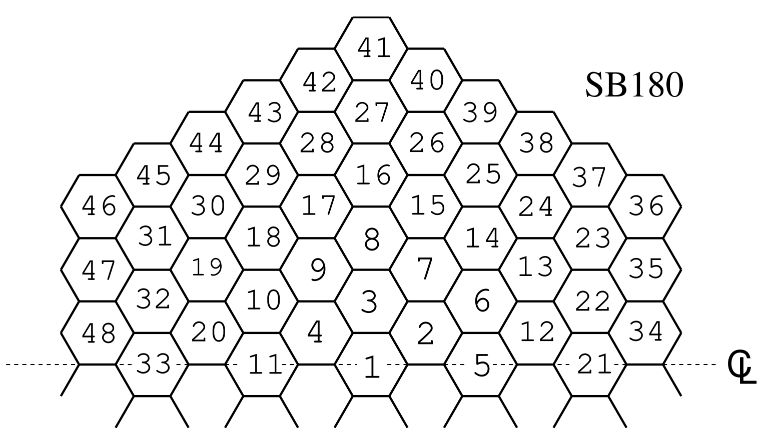


Figure 6: Hexagonal geometry of type SB180

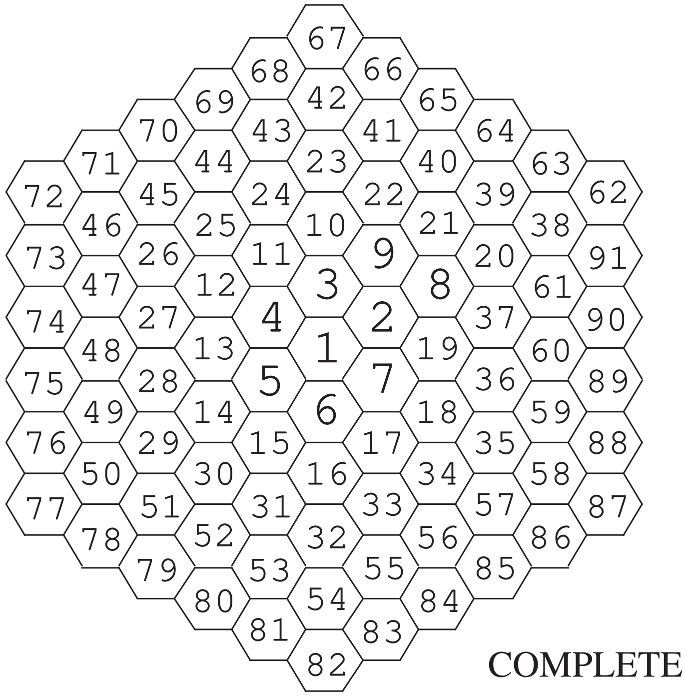


Figure 7: Hexagonal geometry of type COMPLETE

`RADS` This keyword is used to specify the cylindrical correction applied in the $X-Y$ plane for `CAR2D` and `CAR3D` geometries.[12]

`ANG` This keyword allows the angle (see Fig. 8) of the cylindrical notch to be set. By default, no notch is present.

*nrads* Number of different corrections along the cylinder main axis (i.e. the $Z$ axis).

*xrad*(ir) Coordinate of the $Z$ axis from which the correction is applied.

*rrad*(ir) Radius of the real cylindrical boundary.

*ang*(ir) Angle of the cylindrical notch. This data is given if and only if the keyword `ANG` is present. $ang(\mathrm{ir}) = \frac{\pi}{2}$ by default (i.e. the correction is applied at every angle).

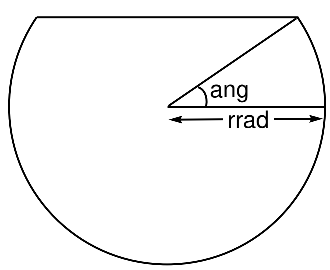


Figure 8: Cylindrical correction in Cartesian geometry

The only combinations of diagonal symmetry permitted are: `X+ DIAG Y- DIAG` and `X- DIAG Y+ DIAG`. In these cases the geometry must be a square. The only combinations of translational symmetry permitted are: `X- TRAN X+ TRAN`, `Y- TRAN Y+ TRAN` and `Z- TRAN Z+ TRAN`.

The input corresponding to the **(descMC)** structure are the following:

Table 6: Structure **(descMC)**

[ `MIX` { (*imix*(i),i=1,*lreg*) |
[[ `PLANE` *iplan* { (*imix*(i),i=1,*lp*) | `SAME` *iplan1*
| [[ `CROWN` { (*imix*(i),i=1,*lc*) | `ALL` *jmix* | `SAME` *iplan1* } ]]
| [[ `UPTO` *ic* `ALL` *jmix* | `SAME` *iplan1* } ]] ]] }
]

where

`MIX` keyword to attribute an material mixture number to each volume inside the axes of symmetry. When a volume is located inside the axes of symmetry but outside the calculation region it must be declared 'virtual' (for example, the corners of a nuclear reactor). The material mixture number should be specified for each volume before mesh-splitting.

*imix* type of material mixture associated with a region. It is important that *imix*≤*nmixt* where *nmixt* is defined in the  module. If *imix*=0, the corresponding volume is replaced by a `VOID` boundary condition. In this case the volume is considered to be virtual and the flux is not calculated. In the case of a diagonal symmetry, the type indicator must not be specified for the volumes outside the axis of symmetry. These values must be specified in the following order: from `X-` to `X+`, from `Y-` to `Y+`, from `Z-` to `Z+` and finally radially from the inside out.

`PLANE` keyword to attribute mixture numbers to each volume inside a single 2D plane. This option is valid only for 3D geometries, Cartesian or hexagonal.

*iplan* plane number for which material mixture are input.

`SAME` keyword to attribute the same material mixture numbers of the *iplan1* plane to the *iplan* plane. In hexagonal geometry, it can indicate that the mixture numbers of the current crown of the *iplan*th plane will be identical to those of the same crown of the *iplan1*th plane.

*iplan1* plane number used as reference to input the current plane or crown(s).

*lp* number of volumes in a plane. In Cartesian geometry, $lp = lx * ly$ and in hexagonal geometry, $lp = lh$.

`CROWN` keyword to attribute mixture numbers to each hexagon of a single crown. This option is only valid for `COMPLETE` hexagonal geometry definition. Each use of the keyword `CROWN` increases the crown number by 1. So it is not required to give its number, but crowns must be defined from the center to the peripherical regions of a plane.

*lc* number of hexagons in the current crown. For the *i*th crown of a compelete hexagonal plane, $lc = (i - 1) * 6$. The first crown is composed of only one hexagon.

`ALL` keyword to specify that the *lc* material mixture number of the current crown have the same value *jmix*.

`UPTO` keyword to attribute material mixture numbers of the current crown up to the *ic* one.

*ic* number of the last crown in `UPTO` option. Its value must be greater than equal to the current crown number.

Here we will assume that *lreg* is the exact number of cells or elementary cases to be considered. For example, if we had used the `DIAG` option with a geometry of type `CAR3D` ($lx$=$ly$), we would have: $lreg=(lx+1)*ly*lz/2$.

The following dimensional constraints must also be respected:

- *nmerge*=number of merged cells (with $nmerge \geq lreg$.),
- *ngen*=number of generation cells (with $ngen \geq nmerge$.),

The inputs corresponding to the **(descPOS)** structure are the following:

Table 7: Structure **(descPOS)**

[ `MESHX` (*xxx*(i),i=1,*lx*+1) ]
[ `MESHY` (*yyy*(i),i=1,*ly*+1) ]
[ `MESHZ` (*zzz*(i),i=1,*lz*+1) ]
[ `RADIUS` (*rrr*(i),i=1,*lr*+1) ]
[ `SIDE` *sidhex* ]
[ `SPLITX` (*ispltx*(i),i=1,*lx*) ]
[ `SPLITY` (*isplty*(i),i=1,*ly*) ]
[ `SPLITZ` (*ispltz*(i),i=1,*lz*) ]
[ `SPLITR` (*ispltr*(i),i=1,*lr*) ]

where

| | |
|---|---|
| `MESHX` | keyword for the mesh of the geometry along the $X$ axis. |
| `MESHY` | keyword for the mesh of the geometry along the $Y$ axis. |
| `MESHZ` | keyword for the mesh of the geometry along the $Z$ axis. |
| `RADIUS` | keyword for the mesh of the geometry in the radial direction. |
| `SIDE` | keyword for the length of a side of a hexagon. |
| *xxx* | abscissa, corresponding to the limits of the regions making up the geometry. These values must be given in order, from `X-` to `X+`. If the geometry presents a diagonal symmetry this data will also be used for the ordinate. |
| *yyy* | ordinate, corresponding to the limits of the regions making up the geometry. These values must be given in order, from `Y-` to `Y+`. |
| *zzz* | height, corresponding to the limits of the regions making up the geometry. These values must be given in order, from `Z-` to `Z+`. |
| *rrr* | Radii in the cases of cylindrical (`TUBE` or `TUBEZ`), spherical (`SPHERE`). It is important to note that we must have *rrr*(1)=0.0. |
| *sidhex* | length of a side of a hexagon. |
| `SPLITX` | keyword for mesh splitting of the geometry along the $X$ axis. |
| `SPLITY` | keyword for mesh splitting of the geometry along the $Y$ axis. |
| `SPLITZ` | keyword for mesh splitting of the geometry along the $Z$ axis. |
| `SPLITR` | keyword for mesh splitting of the geometry in the radial direction. |

*ispltx* number of sub-volumes that will be defined for each row of the volume along the $X$-axis. If the geometry presents a diagonal symmetry this input will also be used for the splitting along the $Y$-axis. By default, *ispltx*=1.

*isplty* number of sub-volumes that will be defined for each row of the volume along the $Y$-axis. If the geometry presents a diagonal symmetry this input will also be used for the splitting along the $X$-axis. By default, *isplty*=1.

*ispltz* number of sub-volumes that will be defined for each row of the volume along the $Z$-axis. By default, *ispltz*(i)=1.

*ispltr* the value of *ispltr* gives the number of sub-volumes that will be defined for each tube or each spherical shell. A negative value permits a splitting into equal sub-volumes; a positive value permits a splitting into equal sub-radius spacings. By default, *ispltr*=1.

The user of the options described above should take care not to exceed the limits imposed by the amount of dynamically allocated memory available. For a pure geometry, let us define the variables $lxp$, $lyp$, $lzp$ and $lrp$ as:

$$\begin{aligned} lxp &= \sum_{i=1}^{lx} ispltx(i) \\ lyp &= \sum_{i=1}^{ly} isplty(i) \\ lzp &= \sum_{i=1}^{lz} ispltz(i) \\ lrp &= \sum_{i=1}^{lr} ispltr(i) \end{aligned}$$

thus, the limits that must be respected are the following:

- $lxp \geq maxpts$ for a `CAR1D` geometry.
- $lh \geq maxpts$ for a `HEX` geometry.
- $lrp \geq maxpts$ for the `TUBE` and `SPHERE` geometries.
- $lxp * lyp \geq maxpts$ for the `CAR2D` geometry without diagonal symmetry.
- $lxp * (lyp + 1)/2 \geq maxpts$ for the `CAR2D` geometry with diagonal symmetry.
- $lrp * lzp \geq maxpts$ for the `TUBEZ` geometry.
- $lxp * lyp * lzp \geq maxpts$ for the `CAR3D` geometry without diagonal symmetry.
- $lxp * (lyp + 1) * lzp/2 \geq maxpts$ for the `CAR3D` geometry with diagonal symmetry.
- $lh * lzp \geq maxpts$ for the `HEXZ` geometry.

### *1.3.2 Examples of geometries*

We will now give a few examples which will permit users to better understand the procedure used to define the geometries in TRIVAC.

1. Slab geometry (see Fig. 9):

```
GEOMETRY1 := GEO: :: CAR1D 6
 X- VOID X+ ALBE 1.2
 MESHX 0.0 0.1 0.3 0.5 0.6 0.8 1.0
 SPLITX 2 2 2 1 2 1
 MIX 1 2 3 4 5 6
 ;
```


Figure 9: Slab geometry with mesh-splitting

2. Two-dimensional hexagonal geometry (see Fig. 10):

```
GEOMETRY4 := GEO: :: HEX 12
 HBC S30 ALBE 1.6
 SIDE 1.3
 MIX 1 1 1 2 2 2 3 3 3 4 5 6
 ;
```

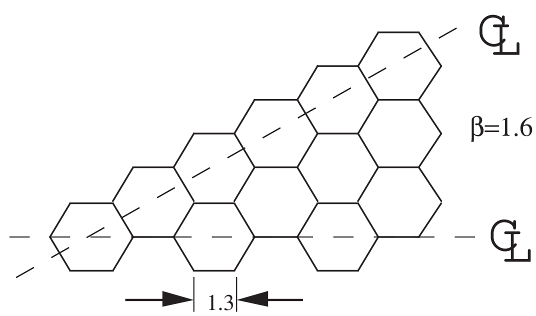


Figure 10: Two-dimensional hexagonal geometry

### 1.4 The `MAC:` module

In TRIVAC the macroscopic cross sections and diffusion coefficients are read from the input data file using REDLEC. The general format of the data for the `MAC:` module in TRIVAC is the following:

Table 8: Structure **(MAC:)**

| |
|---|
| *MACR1* `:=` `MAC:` [ { *MACR1* \| *MACR2* } ] `::` **(mac_data)** |

where

*MACR1* `character*12` name of the LCM object (type `L_MACROLIB`) containing the new Macrolib produced by the module. A Macrolib contains macroscopic cross sections and diffusion coefficients. If *MACR1* appears on both LHS and RHS, it is updated; otherwise, it is created. If *MACR1* is created, all macroscopic cross sections and diffusion coefficients are first initialized to zero.

*MACR2* `character*12` name of the LCM object (type `L_MACROLIB`) containing a read-only Macrolib. The information existing in *MACR2* is copied into *MACR1*, but *MACR2* is not modified.

**(mac_data)** structure containing the data to module `MAC:` (see Sect. 1.4.1).

#### *1.4.1 Data input for module* `MAC:`

Table 9: Structure **(mac_data)**

```
[ EDIT iprint ]
[ NGRO ngroup ]
[ NIFI nifiss ]
[ DELP ndel ]
[ ANIS naniso ]
[ NMIX nmixt ]
[ DELP ndg ]
[ ANIS naniso ]
[ ALBP nalbp (albedp(ia),ia=1,nalbp) ]
[ READ INPUT { [[ (macxs) ]] | OLD (triv2) | DOLD (trip2) } ]
[[ STEP istep READ INPUT [[ (macxs) ]] ]]
;
```

where

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `MAC:`. =0 for no print. The macroscopic cross sections will be printed if the parameter *iprint* is greater than or equal to 2. The transfer cross sections will be printed if this parameter is greater than or equal to 3.

`NGRO` keyword used to define the number of energy groups. This data is given if and only if *MACR1* is created.

*ngroup* the number of energy groups used for the calculations in TRIVAC.

`NIFI` keyword used to specify the maximum number of fissile spectrum associated with each mixture. Each fission spectrum generally represents a fissile isotope. This information is required only if *MACLIB* is created and the cross sections are taken directly from the input data stream.

*nifiss* the maximum number of fissile isotopes per mixture. The default value is *nifiss*=1.

`DELP` keyword used to specify the number of delayed neutron groups.

*ndel* the number of delayed neutron groups. The default value is *ndel*=0.

`ANIS` keyword used to specify the maximum level of anisotropy permitted in the scattering cross sections. This information is required only if *MACLIB* is created and the cross sections are taken directly from the input data stream.

*naniso* number of Legendre orders for the representation of the scattering cross sections. The default value is *naniso*=1 corresponding to the use of isotropic scattering cross sections.

`NMIX` keyword used to define the number of material mixtures. This data is given if and only if *MACR1* is created.

*nmixt* the maximum number of material mixtures (a material mixture is characterized by a distinct set of macroscopic cross sections).

`DELP` keyword used to set *ndg*. This data is used only if the fission spectrum $\vec{\chi}_p$ is different from the delayed neutron spectrum $\vec{\chi}_i$ for each precursor group $i$.

*ndg* number of delayed neutron groups.

`ANIS` keyword used to specify the maximum level of anisotropy permitted in the diffusion cross sections. This data is given only if *MACR1* is created.

*naniso* the maximum level of anisotropy. The default value is *naniso*=1.

`ALBP` keyword used for the input of the physical albedos.

*nalbp* the maximum number of physical albedos.

*albedp* physical albedos (real numbers).

`STEP` keyword used to create a perturbation directory.

*istep* the index of the perturbation directory.

`READ` keyword used to specify input of the cross section information from default input by REDLEC.

**(macxs)** structure describing the format used for reading the mixture cross sections and diffusion coefficients (or perturbation values of the cross sections and diffusion coefficients) from the input data file.

`OLD` keyword used to specify input of the cross section information from default input by REDLEC in the TRIVAC-2 format. The nuclear data will be translated into TRIVAC format and printed on the listing.

**(triv2)** structure describing the format used for reading the mixture cross sections and diffusion coefficients from the input data file in TRIVAC-2 format.

`DOLD` keyword used to specify perturbed input of the cross section information from default input by REDLEC in the TRIVAC-2 format. The perturbed nuclear data will be translated into TRIVAC format and printed on the listing.

**(trip2)** structure describing the format used for reading the mixture values of the perturbed cross sections and diffusion coefficients from the input data file in TRIVAC-2 format.

### *1.4.2 Description of the nuclear data*

Table 10: Structure **(macxs)**

```
MIX matnum
  [ { NTOT0 | TOTAL } (xssigt(jg), jg=1,ngroup) ]
  [ NTOT1 (xssig1(jg), jg=1,ngroup) ]
  [ TRANC (xsstra(jg), jg=1,ngroup) ]
  [ NUSIGF ((xssigf(jf,jg), jg=1,ngroup), jf=1,nifiss) ]
  [ CHI ((xschi(jf,jg), jg=1,ngroup), jf=1,nifiss)]
  [ FIXE (xsfixe(jg), jg=1,ngroup) ]
  [ DIFF (diff(jg), jg=1,ngroup) ]
  [ DIFFX (xdiffx(jg), jg=1,ngroup) ]
  [ DIFFY (xdiffy(jg), jg=1,ngroup) ]
  [ DIFFZ (xdiffz(jg), jg=1,ngroup) ]
  [ NUSIGD (((xssigd(jf,idel,jg), jg=1,ngroup), idel=1,ndel), jf=1,nifiss) ]
  [ CHDL (((xschid(jf,idel,jg), jg=1,ngroup), idel=1,ndel), jf=1,nifiss)]
  [ OVERV (overv(jg), jg=1,ngroup) ]
  [ H-FACTOR (xhfact(jg), jg=1,ngroup) ]
  [ SCAT ((nbscat(jl,jg), ilastg(jl,jg), (scat(jl,jg,ig), ig=1,nbscat(jl,jg) ), jg=1,ngroup), jl=1,naniso) ]
```

where

`MIX` keyword to specify that the macroscopic cross sections associated with a new mixture are to be read.

*matnum* identifier for the next mixture to be read. The maximum value permitted for this identifier is *nmixt*. When *matnum* is absent, the mixtures are numbered consecutively starting with 1 or with the last mixture number read either on the GOXS or the input stream.

`NTOT0` keyword to specify that the total macroscopic cross sections for this mixture follows.

`TOTAL` alias keyword for `NTOT0`.

*xssigt* array representing the multigroup total macroscopic cross section ($\Sigma^g$ in $cm^{-1}$) associated with this mixture.

`NTOT1` keyword to specify that the $P_1$–weighted total macroscopic cross sections for this mixture follows.

*xssig1* array representing the multigroup $P_1$–weighted total macroscopic cross section ($\Sigma_1^g$ in $cm^{-1}$) associated with this mixture.

`TRANC` keyword to specify that the transport correction macroscopic cross sections for this mixture follows.

*xsstra* array representing the multigroup transport correction macroscopic cross section ($\Sigma^g_{\rm tc}$ in $\rm cm^{-1}$) associated with this mixture.

`NUSIGF` keyword to specify that the macroscopic fission cross section multiplied by the average number of neutrons per fission for this mixture follows.

*xssigf* array representing the multigroup macroscopic fission cross section multiplied by the average number of neutrons per fission ($\nu\Sigma^g_f$ in $\rm cm^{-1}$) for all the fissile isotopes associated with this mixture.

`CHI` keyword to specify that the fission spectrum for this mixture follows.

*xschi* array representing the multigroup fission spectrum ($\chi^g$) for all the fissile isotopes associated with this mixture.

`FIXE` keyword to specify that the fixed neutron source density for this mixture follows.

*xsfixe* array representing the multigroup fixed neutron source density for this mixture ($S^g$ in $s^{-1}cm^{-3}$).

`DIFF` keyword to specify that the isotropic diffusion coefficient for this mixture follows.

*diff* array representing the multigroup isotropic diffusion coefficient for this mixture ($D^g$ in $cm$).

`DIFFX` keyword for input of the $X$–directed diffusion coefficient.

*xdiffx* array representing the multigroup $X$–directed diffusion coefficient ($D^g_x$ in cm) for the mixture *matnum*.

`DIFFY` keyword for input of the $Y$–directed diffusion coefficient.

*xdiffy* array representing the multigroup $Y$–directed diffusion coefficient ($D^g_y$ in cm) for the mixture *matnum*.

`DIFFZ` keyword for input of the $Z$–directed diffusion coefficient.

*xdiffz* array representing the multigroup $Z$–directed diffusion coefficient ($D^g_z$ in cm) for the mixture *matnum*.

`NUSIGD` keyword to specify that the delayed macroscopic fission cross section multiplied by the average number of neutrons per fission for this mixture follows.

*xssigd* array representing the delayed multigroup macroscopic fission cross section multiplied by the average number of neutrons per fission ($\nu\Sigma^{g,idel}_f$ in $\rm cm^{-1}$) for all the fissile isotopes associated with this mixture.

`CHDL` keyword to specify that the delayed fission spectrum for this mixture follows.

*xschid* array representing the delayed multigroup fission spectrum ($\chi^{g,idel}$) for all the fissile isotopes associated with this mixture.

`OVERV` keyword for input of the multigroup average of the inverse neutron velocity.

*overv* array representing the multigroup average of the inverse neutron velocity ($< 1/v >^g_m$) for the mixture *matnum*.

`H-FACTOR` keyword to specify that the power factor for this mixture follows.

*hfact* array representing the multigroup power factor for this mixture ($H^g$ in $MeV\ cm^{-1}$).

`SCAT` keyword to specify that the macroscopic scattering cross section matrix for this mixture follows.

*nbscat* array representing the number of secondary groups ig with non vanishing macroscopic scattering cross section towards the primary group jg considered for each anisotropy level associated with this mixture.

*ilastg* array representing the group index of the most thermal group with non-vanishing macroscopic scattering cross section towards the primary group jg considered for each anisotropy level associated with this mixture.

*xsscat* array representing the multigroup macroscopic scattering cross section ($\Sigma_{sl}^{ig\rightarrow jg}$ in $\mathrm{cm}^{-1}$) from the secondary group ig towards the primary group jg considered for each anisotropy level associated with this mixture. The elements are ordered using decreasing secondary group number ig, from *ilastg* to (*ilastg*−*nbscat*+1), and an increasing primary group number jg.

For example, the two group isotropic and linearly anisotropic scattering cross sections (*ngroup*=2, *naniso*=2) given by:

| $L$ | $\Sigma_{s,l}^{1\rightarrow 1}$ | $\Sigma_{s,l}^{1\rightarrow 2}$ | $\Sigma_{s,l}^{2\rightarrow 1}$ | $\Sigma_{s,l}^{2\rightarrow 2}$ |
|---|---|---|---|---|
| 0 | 0.50 $\mathrm{cm}^{-1}$ | 0.20 $\mathrm{cm}^{-1}$ | 0.03 $\mathrm{cm}^{-1}$ | 0.40 $\mathrm{cm}^{-1}$ |
| 1 | 0.05 $\mathrm{cm}^{-1}$ | 0.00 $\mathrm{cm}^{-1}$ | 0.00 $\mathrm{cm}^{-1}$ | 0.04 $\mathrm{cm}^{-1}$ |

must be entered as:

```
SCAT  (*L=0*) 2 2 (*2->1*) 0.03 (*1->1*) 0.50
              2 2 (*2->2*) 0.40 (*1->2*) 0.20
      (*L=1*) 1 1               (*1->1*) 0.05
              1 2 (*2->2*) 0.04
```

### 1.5 The `BIVACT:` module

The `BIVACT:` module is used to perform a BIVAC-type TRACKING on a 1D/2D geometry.[3,4,14] The geometry is analyzed and a LCM object with signature `L_BIVAC` is created with the following information:

- Diagonal and hexagonal symmetries are unfolded and the mesh-splitting operations are performed. Volumes, material mixture and averaged flux recovery indices are computed on the resulting geometry.
- A finite element discretization is performed and the corresponding numbering is saved.
- The unit finite element matrices (mass, stiffness, etc.) are recovered.

The calling specifications are:

Table 11: Structure **(BIVACT:)**

*TRACK* `:=` `BIVACT:` [ *TRACK* ] *GEOM* `::` **(bivact_data)**

where

*TRACK* — `character*12` name of the LCM object (type `L_BIVAC`) containing the TRACKING information. If *TRACK* appears on the RHS, the previous settings will be applied by default.

*GEOM* — `character*12` name of the LCM object (type `L_GEOM`) containing the geometry.

**(bivact_data)** — structure containing the data to module `BIVACT:` (see Sect. 1.5.1).

#### *1.5.1 Data input for module* `BIVACT:`

Table 12: Structure **(bivact_data)**

[ `EDIT` *iprint* ]
[ `TITL` *TITLE* ]
[ `MAXR` *maxpts* ]
[ { `PRIM` [ *ielem icol* [ *isplh* ] ]
  | `DUAL` [ *ielem icol* [ *isplh* ] ]
  | `MCFD` *isplh* } ]
[ { `PN` | `SPN` } *n* [ `SCAT` [ `DIFF` ] *iscat* ] [ `VOID` *nvd* ] ]
;

where

`EDIT` — keyword used to set *iprint*.

*iprint* — index used to control the printing in module `BIVACT:`. =0 for no print; =1 for minimum printing (default value); Larger values produce increasing amounts of output.

`TITL` keyword which allows the run title to be set.

*TITLE* the title associated with a TRIVAC run. This title may contain up to 72 characters. The default when `TITL` is not specified is no title.

`MAXR` keyword which permits the maximum number of regions to be considered during a TRIVAC run to be specified.

*maxpts* maximum dimensions of the problem to be considered. The default value is set to the number of regions previously computed by the `GEO:` module but this value is insufficient if symmetries or mesh-splitting are specified.

`PRIM` keyword to set a primal finite element (classical) discretization.

`DUAL` keyword to set a mixed-dual finite element discretization. If the geometry is hexagonal, a Thomas-Raviart-Schneider method is used.[15]

`MCFD` keyword to set a mesh-centered finite difference discretization in hexagonal geometry.

*ielem* order of the finite element representation. The values permitted are 1 (linear polynomials), 2 (parabolic polynomials), 3 (cubic polynomials) or 4 (quartic polynomials). By default *ielem*=1.

*icol* type of quadrature used to integrate the mass matrices. The values permitted are 1 (analytical integration), 2 (Gauss-Lobatto quadrature) or 3 (Gauss-Legendre quadrature). By default *icol*=2. The analytical integration corresponds to classical finite elements; the Gauss-Lobatto quadrature corresponds to a variational or nodal type collocation and the Gauss-Legendre quadrature corresponds to superconvergent finite elements.

*isplh* type of hexagonal mesh-splitting. This data is given only if the geometry is 2D hexagonal. The values permitted are 1 (full hexagons), 2 for splitting each hexagon into 6 triangles, 3 for splitting each hexagon into 24 triangles, 5 for splitting each hexagon into 96 triangles, 9 for splitting each hexagon into 384 triangles and 17 for splitting each hexagon into 1536 triangles. The values permitted with the Thomas-Raviart-Schneider method are: 1 (3 lozanges per hexagon), $>1$ for performing a mesh-splitting in $3\times isplh^2$ losanges per hexagon.

`PN` keyword to set a spherical harmonics ($P_n$) expansion of the flux.[9]

`SPN` keyword to set a simplified spherical harmonics ($SP_n$) expansion of the flux.[9, 10] This option is currently available with 1D and 2D Cartesian geometries and with 2D hexagonal geometries.

*n* order of the $P_n$ or $SP_n$ expansion (odd number). Set to zero for diffusion theory (default value).

`SCAT` keyword to limit the anisotropy of scattering sources.

`DIFF` keyword to force using $1/3D^g$ as $\Sigma_1^g$ cross sections. A $P_1$ or $SP_1$ method will therefore behave as diffusion theory.

*iscat* number of terms in the scattering sources. $iscat = 1$ is used for isotropic scattering in the laboratory system. $iscat = 2$ is used for linearly anisotropic scattering in the laboratory system. The default value is set to $n+1$ in $P_n$ or $SP_n$ case.

`VOID` keyword to set the number of base points in the Gauss-Legendre quadrature used to integrate void boundary conditions if $icol = 3$ and $n \neq 0$.

*nvd* type of quadrature. The values permitted are: 0 (use a ($n$+2)–point quadrature consistent with $P_{\rm n}$ theory), 1 (use a ($n$+1)–point quadrature consistent with $S_{\rm n+1}$ theory), 2 (use an analytical integration of the void boundary conditions). By default *nvd*=0.

Various finite element approximations can be obtained by combining different values of *ielem* and *icol*:

- `PRIM 1 1` : Linear finite elements;
- `PRIM 1 2` : Mesh corner finite differences;
- `PRIM 1 3` : Linear superconvergent finite elements;
- `PRIM 2 1` : Quadratic finite elements;
- `PRIM 2 2` : Quadratic variational collocation method;
- `PRIM 2 3` : Quadratic superconvergent finite elements;
- `PRIM 3 1` : Cubic finite elements;
- `PRIM 3 2` : Cubic variational collocation method;
- `PRIM 3 3` : Cubic superconvergent finite elements;
- `PRIM 4 2` : Quartic variational collocation method;
- `DUAL 1 1` : Mixed-dual linear finite elements;
- `DUAL 1 2` : Mesh centered finite differences;
- `DUAL 1 3` : Mixed-dual linear superconvergent finite elements (numerically equivalent to `PRIM 1 3`);
- `DUAL 2 1` : Mixed-dual quadratic finite elements;
- `DUAL 2 2` : Quadratic nodal collocation method;
- `DUAL 2 3` : Mixed-dual quadratic superconvergent finite elements (numerically equivalent to `PRIM 2 3`);
- `DUAL 3 1` : Mixed-dual cubic finite elements;
- `DUAL 3 2` : Cubic nodal collocation method;
- `DUAL 3 3` : Mixed-dual cubic superconvergent finite elements (numerically equivalent to `PRIM 3 3`);
- `DUAL 4 2` : Quartic nodal collocation method;

## 1.6 The `TRIVAT:` module

The `TRIVAT:` module is used to perform a TRIVAC-type TRACKING on a 1D/2D/3D geometry.[4–8, 14] The geometry is analyzed and a LCM object with signature `L_TRIVAC` is created with the following information:

- Diagonal and hexagonal symmetries are unfolded and the mesh-splitting operations are performed. Volumes, material mixture and averaged flux recovery indices are computed on the resulting geometry.
- A finite element discretization is performed and the corresponding numbering is saved.
- The unit finite element matrices (mass, stiffness, etc.) are recovered.
- Indices related to an ADI preconditioning with or without supervectorization are saved.

The calling specifications are:

Table 13: Structure **(TRIVAT:)**

| |
|---|
| *TRACK* `:=` `TRIVAT:` [ *TRACK* ] *GEOM* `::` **(trivat_data)** |

where

*TRACK* `character*12` of the LCM object (type `L_TRIVAC`) containing the TRACKING information. If *TRACK* appears on the RHS, the previous settings will be applied by default.

*GEOM* `character*12` of the LCM object (type `L_GEOM`) containing the geometry.

**(trivat_data)** structure containing the data to module `TRIVAT:` (see Sect. 1.6.1).

### *1.6.1 Data input for module* `TRIVAT:`

Table 14: Structure **(trivat_data)**

| |
|---|
| [ `EDIT` *iprint* ]<br>[ `TITL` *TITLE* ]<br>[ `MAXR` *maxpts* ]<br>[ { `PRIM` [ *ielem* [ *isplh* ] ] \| `DUAL` [ *ielem icol* [ *isplh* ] ] \| `MCFD` [ *ielem* [ *isplh* ] ] \| `LUMP` [ *ielem* ] } ]<br>[ `SPN` *n* [ `SCAT` [ `DIFF` ] *iscat* ] [ `VOID` *nvd* ] ]<br>[ `ADI` *nadi* ]<br>[ `VECT` [ *iseg* ] [ `PRTV` *impv* ] ]<br>; |

where

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `TRIVAT:`. =0 for no print; =1 for minimum printing (default value); Larger values produce increasing amounts of output.

`TITL` keyword which allows the run title to be set.

*TITLE* the title associated with a TRIVAC run. This title may contain up to 72 characters. The default when `TITL` is not specified is no title.

`MAXR` keyword which permits the maximum number of regions to be considered during a TRIVAC run to be specified.

*maxpts* maximum dimensions of the problem to be considered. The default value is set to the number of regions previously computed by the `GEO:` module but this value is insufficient if symmetries or mesh-splitting are specified.

`PRIM` keyword to set a discretization based on the variational collocation method.

`DUAL` keyword to set a mixed-dual finite element discretization. If the geometry is hexagonal, a Thomas-Raviart-Schneider method is used.[15]

`MCFD` keyword to set a discretization based on the nodal collocation method. The mesh centered finite difference approximation is the default option and is generally set using `MCFD 1`. The `MCFD` approximations are numerically equivalent to the `DUAL` approximations with *icol*=2; however, the `MCFD` approximations are less expensive.

`LUMP` keyword to set a discretization based on the nodal collocation method with serendipity approximation. The serendipity approximation is different from the `MCFD` option in cases with *ielem*≥2. This option is not available for hexagonal geometries.

*ielem* order of the finite element representation. The values permitted are: 1 (linear polynomials), 2 (parabolic polynomials), 3 (cubic polynomials) or 4 (quartic polynomials). By default *ielem*=1.

*icol* type of quadrature used to integrate the mass matrices. The values permitted are: 1 (analytical integration), 2 (Gauss-Lobatto quadrature) or 3 (Gauss-Legendre quadrature). By default *icol*=2. The analytical integration corresponds to classical finite elements; the Gauss-Lobatto quadrature corresponds to a variational or nodal type collocation and the Gauss-Legendre quadrature corresponds to superconvergent finite elements.

*isplh* type of hexagonal mesh-splitting. This data is given only if the geometry is 2D or 3D hexagonal. The values permitted with the `MCFD` option are: 1 (full hexagons), 2 for splitting each hexagon into 6 triangles, 3 for splitting each hexagon into 24 triangles, etc. The values permitted with the `PRIM` option are: 1 (full hexagons) and 2 for splitting each hexagon into 6 triangles. The values permitted with the Thomas-Raviart-Schneider method are: 1 (3 lozenges per hexagon), $> 1$ for performing a mesh-splitting in $3\times isplh^2$ losanges per hexagon.

`SPN` keyword to set a simplified spherical harmonics ($SP_n$) expansion of the flux.[9,10] This option is available with 1D, 2D and 3D Cartesian geometries and with 2D and 3D hexagonal geometries.

*n* order of the $P_n$ or $SP_n$ expansion (odd number). Set to zero for diffusion theory (default value).

`SCAT` keyword to limit the anisotropy of scattering sources.

`DIFF` keyword to force using $1/3D^g$ as $\Sigma_1^g$ cross sections. A $P_1$ or $SP_1$ method will therefore behave as diffusion theory.

*iscat* number of terms in the scattering sources. $iscat = 1$ is used for isotropic scattering in the laboratory system. $iscat = 2$ is used for linearly anisotropic scattering in the laboratory system. The default value is set to $n + 1$ in $P_n$ or $SP_n$ case.

`VOID` keyword to set the number of base points in the Gauss-Legendre quadrature used to integrate void boundary conditions if *icol* = 3 and $n \neq 0$.

*nvd* type of quadrature. The values permitted are: 0 (use a ($n$+2)–point quadrature consistent with $P_{\rm n}$ theory), 1 (use a ($n$+1)–point quadrature consistent with $S_{\rm n+1}$ theory), 2 (use an analytical integration of the void boundary conditions). By default *nvd*=0.

`ADI` keyword to set the number of ADI iterations at the inner iterative level.

*nadi* number of ADI iterations (default: *nadi* = 2).

`VECT` keyword to set an ADI preconditionning with supervectorization. By default, TRIVAC uses an ADI preconditionning without supervectorization.

*iseg* width of a vectorial register. *iseg* is generally a multiple of 64. By default, *iseg*=64.

`PRTV` keyword used to set *impv*.

*impv* index used to control the printing in supervectorization subroutines. =0 for no print; =1 for minimum printing (default value); Larger values produce increasing amounts of output.

Various finite element approximations can be obtained by combining different values of *ielem* and *isplh* (see Sect. 1.5).

### 1.7 The BIVACA: module

The BIVACA: module is used to compute the finite element system matrices (type L_SYSTEM) corresponding to a BIVAC TRACKING (type L_BIVAC) and to a set of nuclear properties (type L_MACROLIB). The calling specifications are:

Table 15: Structure **(BIVACA:)**

| |
|---|
| *SYST* := BIVACA: [ *SYST* ] *MACRO* *TRACK* :: **(bivaca_data)** |

where

*SYST* — character*12 name of the LCM object (type L_SYSTEM) containing the system matrices. If *SYST* appears on the RHS, the system matrices previously stored in *SYST* are kept.

*MACRO* — character*12 name of the LCM object (type L_MACROLIB) containing the macroscopic cross sections and diffusion coefficients.

*TRACK* — character*12 name of the LCM object (type L_BIVAC) containing the BIVAC TRACKING.

**(bivaca_data)** — structure containing the data to module BIVACA: (see Sect. 1.7.1).

#### *1.7.1 Data input for module* BIVACA:

Table 16: Structure **(bivaca_data)**

| |
|---|
| [ EDIT *iprint* ] [ UNIT ]<br>; |

where

EDIT — keyword used to set *iprint*.

*iprint* — index used to control the printing in module BIVACA:. =0 for no print; =1 for minimum printing (default value); Larger values produce increasing amounts of output.

UNIT — A system matrix corresponding to cross sections all set to 1.0 is computed. This keyword is mandatory if the system matrices in *SYST* are going to be used by INIKIN: or KINSOL: modules (see Sects. 1.14 and 1.15).

### 1.8 The TRIVAA: module

The TRIVAA: module is used to compute the finite element system matrices (type `L_SYSTEM`) corresponding to a TRIVAC TRACKING (type `L_TRIVAC`) and to a set of nuclear properties (type `L_MACROLIB`). The calling specifications are:

Table 17: Structure **(TRIVAA:)**

| |
|---|
| *SYST* `:=` `TRIVAA:` [ *SYST* ] *MACRO* *TRACK* [ *DMACRO* ] `::` **(trivaa_data)** |

where

*SYST* — `character*12` name of the LCM object (type `L_SYSTEM`) containing the system matrices. If *SYST* appears on the RHS, the system matrices previously stored in *SYST* are kept.

*MACRO* — `character*12` name of the LCM object (type `L_MACROLIB`) containing the macroscopic cross sections and diffusion coefficients.

*TRACK* — `character*12` name of the LCM object (type `L_TRIVAC`) containing the TRIVAC TRACKING.

*DMACRO* — `character*12` name of the LCM object (type `L_MACROLIB`) containing derivatives or perturbations of the macroscopic cross sections and diffusion coefficients. If *DMACRO* is given, only the derivatives or perturbations of the system matrices are computed.

**(trivaa_data)** — structure containing the data to module `TRIVAA:` (see Sect. 1.8.1).

#### *1.8.1 Data input for module* `TRIVAA:`

Table 18: Structure **(trivaa_data)**

| |
|---|
| [ `EDIT` *iprint* ]<br>[ `SKIP` ] [{ `DERI` \| `PERT` }] [ `UNIT` ] [ `OVEL` ]<br>; |

where

`EDIT` — keyword used to set *iprint*.

*iprint* — index used to control the printing in module `TRIVAA:`. =0 for no print; =1 for minimum printing (default value); Larger values produce increasing amounts of output.

`SKIP` — keyword used to skip the system matrix assembly but to perform the $L-D-L^T$ factorization. Use the system matrices already present in *SYST*.

`DERI` — The information recovered from *DMACRO* is used as derivatives of nuclear properties with respect to a state variable. Derivatives of system matrices with respect to the same state variable are computed.

`PERT` The information recovered from *DMACRO* is used as the perturbation of the nuclear properties. Perturbations of the system matrices are computed.

`UNIT` A system matrix corresponding to cross sections all set to 1.0 is computed. This keyword is mandatory if the system matrices in *SYST* are going to be used by `INIKIN:` or `KINSOL:` modules (see Sects. 1.14 and 1.15).

`OVEL` The reciprocal neutron velocities for each material mixture are recovered from the input MACROLIB *MACRO* and used to compute the corresponding system matrices. This capability is deprecated.

### 1.9 The `FLUD:` module

The `FLUD:` module is used to compute the solution to an eigenvalue problem corresponding to a set of system matrices (type `L_SYSTEM`). The calling specifications are:

Table 19: Structure **(FLUD:)**

| |
|---|
| *FLUX* `:=` `FLUD:` [ *FLUX* ] *SYST* *TRACK* [ *MACRO* ] `::` **(flud_data)** |

where

| | |
|---|---|
| *FLUX* | `character*12` name of the LCM object (type `L_FLUX`) containing the solution. If *FLUX* appears on the RHS, the solution previously stored in *FLUX* is used to initialize the new iterative process; otherwise, a uniform unknown vector is used. |
| *SYST* | `character*12` name of the LCM object (type `L_SYSTEM`) containing the system matrices. |
| *TRACK* | `character*12` name of the LCM object (type `L_TRACK`) containing the TRACKING. |
| *MACRO* | `character*12` name of the optional LCM object (type `L_MACROLIB`) containing the cross sections. This object is only used to set a link to the MACROLIB name inside the FLUX object. By default, the name of the MACROLIB is recovered from the link in the SYSTEM object. |
| **(flud_data)** | structure containing the data to module `FLUD:` (see Sect. 1.9.1). |

#### *1.9.1 Data input for module* `FLUD:`

Table 20: Structure **(flud_data)**

| |
|---|
| [ `EDIT` *iprint* ]<br>[ { `VAR1` \| `ACCE` } *icl1* *icl2* ]<br>[ `EXTE` [ *maxout* ] [ *epsout* ] ]<br>[ `THER` [ *maxthr* ] [ *epsthr* ] ]<br>[ `ADI` *nadi* ]<br>[ `ADJ` ]<br>[ `MONI` *lmod* [ `RAND` ] ]<br>[ `RELAX` *relax* ]<br>; |

where

| | |
|---|---|
| `EDIT` | keyword used to set *iprint*. |
| *iprint* | index used to control the printing in module `FLUD:`. =0 for no print; =1 for minimum printing (default value); =2 iteration history is printed; =3 the solution is printed; =4 at each iteration, the new solution is compared to a reference solution previously stored in *FLUX* under name `REF`; =5 the convergence histogram is stored in *FLUX*. |

| | |
|---|---|
| `VAR1` | keyword used to set the parameters (*icl1* and *icl2*) of the symmetrical variational acceleration technique (SVAT). |
| `ACCE` | alias keyword for `VAR1`. |
| *icl1* | number of free outer iterations in a cycle of the SVAT. The default value is $icl1 = 3$. |
| *icl2* | number of accelerated outer iterations in a cycle of the SVAT. The default value is $icl2 = 3$. A convergence in free iterations is obtained by setting $icl1 = 200$ (or $icl1 = maxx0$) and $icl2 = 0$. |
| `EXTE` | keyword to specify that the control parameters for the external iteration are to be modified. |
| *maxout* | maximum number of external iterations. The fixed default value is $maxout = 200$. |
| *epsout* | convergence criterion for the external iterations. The fixed default value is $epsout = 1.0 \times 10^{-4}$. The outer iterations are stopped when the following criteria is reached: $$\max_i \lvert\Phi_i^{(k-1)} - \Phi_i^{(k)}\rvert \ \leq\ epsout \times \max_i \lvert\Phi_i^{(k)}\rvert$$ where $\vec{\Phi}^{(k)} = \mathrm{col}\{\Phi_i^{(k)} \ ; \ i = 1, I\}$ is the product of the $B$ matrix times the unknown vector at the $k$-th outer iteration. |
| `THER` | keyword to specify that the control parameters for the thermal iterations are to be modified. |
| *maxthr* | maximum number of thermal iterations. The fixed default value is $maxthr = 0$ corresponding to no thermal iterations. |
| *epsthr* | convergence criterion for the thermal iterations. The fixed default value is $epsthr = 1.0 \times 10^{-2}$. |
| `ADI` | keyword used to set *nadi* in cases where Trivac is used. |
| *nadi* | number of alternating direction implicit (ADI) inner iterations per outer iteration. The default value is $nadi = 1$. If this value causes a failure of the acceleration process, it is recommended that a larger value be tried. The optimal choice is generally the minimum value of *nadi* which allows a convergence in less than 75 outer iterations. $nadi = 1$ or $nadi = 2$ is generally the best choice for production-type calculations. The greater *nadi* is, the smaller the asymptotic convergence constant (ACC) becomes. Taking an arbitrary large value (e.g., $nadi = 20$) leads to numerical results identical to those of the inverse power method where the system matrices are accurately inverted at each outer iteration (at a prohibitive CPU cost). In this case, the ACC is almost equal to the dominance ratio of the iterative matrix. The default value is recovered in the state vector of the TRACKING object *TRACK*. |
| `ADJ` | keyword used to obtain the solution to both the direct and adjoint eigenvalue problems. The adjoint solution is required if we subsequently want to perform a perturbation calculation. |
| `MONI` | keyword used to obtain the first harmonics of the solution and to set *lmod*. *A full core representation of the reactor should be used to compute its harmonics. If symmetries are set in the geometry, some harmonics may be skipped. If the reactor is symmetric, a uniform initial estimate of the harmonics may cause some harmonics to be skipped; the keyword* `RAND` *should therefore be used.* |
| *lmod* | the *lmod* first bi-orthonormalized harmonics of the solution are computed using the SVAT-accelerated preconditioned power method with a Hotelling deflation procedure.[11] |
| `RAND` | keyword used to initialize the harmonics calculations (option `MONI`) with a random estimate rather than a uniform estimate. This option has no effect if *FLUX* appears on the RHS. |

`RELAX` keyword used to set the relaxation parameter. This keyword must be specified each time a relaxation is required.

*relax* relaxation parameter selected in the interval $0 < relax \leq 1.0$ and used to update the flux information in the *FLUX* object. The updated value is taken equal to $(1.0 - relax)$ times the previous value (given in the RHS *FLUX* object) plus *relax* times the value computed within current `FLUD:` call. The default value is $relax = 1.0$.

## 1.10 The DELTA: module

The DELTA: module is used to compute the source components of a fixed source eigenvalue problem corresponding to a set of unperturbed and perturbation system matrices (type L_SYSTEM).

In the direct case, the fixed source is computed as:

$$\vec{S} = (\delta\,\mathbb{A} - \lambda_o\,\delta\mathbb{B})\,\vec{\Phi} - \delta\lambda\;\mathbb{B}_o\,\vec{\Phi} \tag{1.1}$$

where the direct source vector $\vec{S}$ is orthogonal to the unperturbed adjoint flux $\Phi^*$.

In the adjoint case, the fixed source is computed as:

$$\vec{S}^* = \left(\delta\,\mathbb{A}^\top - \lambda_o\,\delta\mathbb{B}^\top\right)\vec{\Phi}^* - \delta\lambda\;\mathbb{B}_o^\top\,\vec{\Phi}^* \tag{1.2}$$

where the adjoint source vector $\vec{S}^*$ is orthogonal to the unperturbed direct flux $\Phi$ and where $\delta\lambda$ is the perturbation of the eigenvalue, as computed from the Rayleigh ratio.

The calling specifications are:

Table 21: Structure **(DELTA:)**

| |
|---|
| *GPT* := DELTA: [ *GPT* ] *FLUX0 SYST0 DSYST TRACK* :: **(delta_data)** |

where

*GPT* character*12 name of the LCM object (type L_GPT) containing the fixed source. If *GPT* appears on the RHS, this information is used to initialize the state vector.

*FLUX0* character*12 name of the LCM object (type L_FLUX) containing the unperturbed flux.

*SYST0* character*12 name of the LCM object (type L_SYSTEM) containing the unperturbed system matrices.

*DSYST* character*12 name of the LCM object (type L_SYSTEM) containing a perturbation to the system matrices.

*TRACK* character*12 name of the LCM object (type L_TRACK) containing the TRACKING.

**(delta_data)** structure containing the data to module DELTA: (see Sect. 1.10.1).

### *1.10.1 Data input for module* DELTA:

Table 22: Structure **(delta_data)**

| |
|---|
| [ EDIT *iprint* ]<br>[ ADJ ]<br>; |

where

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `DELTA:`.

`ADJ` keyword used to set the source on an adjoint fixed source eigenvalue problem.

## 1.11 The GPTFLU: module

The GPTFLU: module is used to compute the solution to a fixed source eigenvalue problem corresponding to a set of unperturbed system matrices and sources vectors.

If $\vec{S}$ is the source term of the explicit generalized adjoint equation, this module will solve:

$$(\mathbb{A}_o - \lambda_o \, \mathbb{B}_o) \, \vec{\Gamma}_i = \vec{S}_i \tag{1.3}$$

where the direct source vector $\vec{S}_i$ is orthogonal to the adjoint flux.

If $\vec{S}$ is the source term of the implicit generalized adjoint equation, this module will solve:

$$\left(\mathbb{A}_o^\top - \lambda_o \, \mathbb{B}_o^\top\right) \vec{\Gamma}_j^* = \vec{S}_j^* \tag{1.4}$$

where the adjoint source vector $\vec{S}_j^*$ is orthogonal to the direct flux.

The calling specifications are:

Table 23: Structure **(GPTFLU:)**

| |
|---|
| *FLUX_GPT* := GPTFLU: [ *FLUX_GPT* ] *GPT FLUX0 SYST TRACK* :: **(gptflu_data)** |

where

*FLUX_GPT* character*12 name of the LCM object (type L_FLUX) containing the GPT solution. If *FLUX_GPT* appears on the RHS, the solution previously stored in *FLUX_GPT* is used to initialize the new iterative process; otherwise, a uniform unknown vector is used.

*GPT* character*12 name of the LCM object (type L_GPT) containing the fixed sources.

*FLUX0* character*12 name of the LCM object (type L_FLUX) containing the unperturbed flux used to decontaminate the GPT solution.

*SYST* character*12 name of the LCM object (type L_SYSTEM) containing the unperturbed system matrices.

*TRACK* character*12 name of the LCM object (type L_TRACK) containing the TRACKING.

**(gptflu_data)** structure containing the data to module GPTFLU:.

### *1.11.1 Data input for module* GPTFLU:

Table 24: Structure **(gptflu_data)**

| |
|---|
| [ EDIT *iprint* ] |
| [ { VAR1 \| ACCE } *icl1 icl2* ] |
| [ EXTE [ *maxout* ] [ *epsout* ] ] |
| [ THER [ *maxthr* ] [ *epsthr* ] ] |
| [ ADI *nadi* ] |
| [ { EXPLICIT \| IMPLICIT } ] |
| FROM-TO { ALL \| $i_{src1}$ $i_{src2}$ } |
| ; |

where

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `GPTFLU:`. =0 for no print; =1 for minimum printing (default value); =2 iteration history is printed; =3 the solution is printed; =4 at each iteration, the new solution is compared to a reference solution previously stored in *FLUX_GPT* under the name `REF`; =5 the convergence histogram is stored in *FLUX_GPT*.

`VAR1` keyword used to set the parameters (*icl1* and *icl2*) of the variational acceleration technique.

`ACCE` alias keyword for `VAR1`.

*icl1* number of free outer iterations in a cycle of the SVAT. The default value is $icl1 = 3$.

*icl2* number of accelerated outer iterations in a cycle of the SVAT. The default value is $icl2 = 3$. A convergence in free iterations is obtained by setting $icl1 = 200$ (or $icl1 = maxx0$) and $icl2 = 0$.

`EXTE` keyword to specify that the control parameters for the external iteration are to be modified.

*maxout* maximum number of external iterations. The fixed default value is $maxout = 200$.

*epsout* convergence criterion for the external iterations. The fixed default value is $epsout = 1.0 \times 10^{-4}$. The outer iterations are stopped when the following criteria is reached:

$$\max_i |\Gamma_i^{(k-1)} - \Gamma_i^{(k)}| \ \leq \ epsout \times \max_i |\Gamma_i^{(k)}|$$

where $\vec{\Gamma}^{(k)} = \mathrm{col}\{\Gamma_i^{(k)} \ ; \ i = 1, I\}$ is the product of the $\mathbb{B}$ matrix times the unknown vector at the $k$-th outer iteration.

`THER` keyword to specify that the control parameters for the thermal iterations are to be modified.

*maxthr* maximum number of thermal iterations. The fixed default value is $maxthr = 0$ corresponding to no thermal iterations.

*epsthr* convergence criterion for the thermal iterations. The fixed default value is $epsthr = 1.0 \times 10^{-2}$.

`ADI` keyword used to set *nadi* in cases where Trivac is used.

*nadi* number of alternating direction implicit (ADI) inner iterations per outer iteration. The default value is $nadi = 1$. If this value causes a failure of the acceleration process, it is recommended that a larger value be tried. The optimal choice is generally the minimum value of *nadi* which allows a convergence in less than 75 outer iterations. $nadi = 1$ or $nadi = 2$ is generally the best choice for production-type calculations. The greater *nadi* is, the smaller the asymptotic convergence constant (ACC) becomes. Taking an arbitrary large value (e.g., $nadi = 20$) leads to numerical results identical to those obtained by inverting the system matrices at each outer iteration (at a prohibitive CPU cost). In this case, the ACC is almost equal to the dominance ratio of the iterative matrix.

`EXPLICIT` keyword used to obtain the solution of an direct fixed source eigenvalue problem.

`IMPLICIT` keyword used to obtain the solution of an adjoint fixed source eigenvalue problem. If neither '`EXPLICIT`' nor '`IMPLICIT`' are provided the default value will be chosen as a function of $n_{var}$ and $n_{cst} + 1$.

`FROM-TO` keyword used to specify the numbers of the sources for which a generalized adjoint will be calculated.

`ALL` keyword used to recover all sources available in *GPT*.

$i_{src1}$ number of the first source.

$i_{src1}$ number of the last source.

### 1.12 The `OUT:` module

The `OUT:` module is used to compute the reaction rates and to store them in an extended MACROLIB (type `L_MACROLIB`) corresponding to a solution (type `L_FLUX`) of the matrix system. The calling specifications are:

Table 25: Structure **(OUT:)**

| |
|---|
| *MACRO2* `:=` `OUT:` *FLUX TRACK MACRO GEOM* `::` **(out_data)** |

where

*MACRO2* `character*12` name of the LCM object (type `L_MACROLIB`) containing the extended MACROLIB.

*FLUX* `character*12` name of the LCM object (type `L_FLUX`) containing a solution.

*TRACK* `character*12` name of the LCM object (type `L_TRACK`) containing a TRACKING.

*MACRO* `character*12` name of the LCM object (type `L_MACROLIB`) containing the reference MACROLIB.

*GEOM* `character*12` name of the LCM object (type `L_GEOM`) containing the reference GEOMETRY.

**(out_data)** structure containing the data to module `OUT:`.

#### *1.12.1 Data input for module* `OUT:`

Table 26: Structure **(out_data)**

| |
|---|
| [ `EDIT` *iprint* ]<br>[ `MODE` *imode* ]<br>[ { `DIRE` \| `ADJO` } ]<br>[ `POWR` *power* ]<br>[ `INTG` { `IN` \| `MIX` \| (*ihom*(i), i=1,nreg )} ]<br>; |

where

`EDIT` keyword used to set *iprint*.

*iprint* index used to control the printing in module `OUT:`. =0 for no print; =1 for minimum printing (default value).

`MODE` keyword to specify the flux harmonic index *imode*.

*imode* index of the flux harmonic recovered by the `OUT:` module if the `MONI` keyword was set in module `FLUD:` (see Sect. 1.9.1). By default, it is assumed that the `MONI` keyword was not used.

`DIRE` use the direct flux to perform homogenization (default value).

`ADJO` use the adjoint flux to perform homogenization.

`POWR` keyword used to set *power*.

*power* value of the power in MW used to normalize the flux. By default, the flux is not normalized.

`INTG` keyword used to compute the reaction rates.

`IN` keyword for computing the reaction rates on the geometry mesh (see Sect. 1.3.1) before mesh-splitting.

`MIX` keyword for computing the reaction rates on the mixture mesh previously used to define the geometry (see Sect. 1.3.1) before mesh-splitting.

*ihom* index of the homogenized region corresponding to the each region of the geometry (see Sect. 1.3.1) before mesh-splitting.

### 1.13 The `ERROR:` module

The `ERROR:` module is used to compare reaction rates contained into two extended MACROLIBS and to print statistics regarding the comparison.

The QUANDRY-type power densities are first compared. These power densities are defined by the following relation:

$$P_i^{\rm quandry} = \frac{\sum\limits_i V_i}{V_i} \frac{P_i}{\sum\limits_i P_i}$$

where $P_i$ is the total power and $V_i$ is the volume of the region $i$. The maximum and averaged errors are respectively defined by:

$$\epsilon_{\rm max} = \max_i \frac{|P_i^{\rm quandry} - P_i^{\rm quandry*}|}{P_i^{\rm quandry*}}$$

and

$$\bar{\epsilon} = \frac{1}{V_{\rm core}} \sum_i \left[ \frac{|P_i^{\rm quandry} - P_i^{\rm quandry*}|}{P_i^{\rm quandry*}} \right] V_i$$

where $P_i^{\rm quandry*}$ is computed using the reference powers (stored in *MACRO1*) and $V_{\rm core}$ is the total volume of the regions where the power density is not equal to zero.

The normalized removal rates $T_{i,g}^{\rm norm}$ in each region $i$ and energy group $g$ are next computed using the following formula:

$$T_{i,g} = \left(\Sigma_{i,g} - \Sigma_{{\rm w}i,g}\right) \phi_{i,g} V_i$$

$$T_{i,g}^{\rm norm} = \frac{1}{\sum\limits_i \sum\limits_g T_{i,g}} \; T_{i,g}$$

where $\Sigma_{i,g}$ is the total macroscopic cross section, $\Sigma_{{\rm w}i,g}$ is the within-group scattering cross section and $\phi_{i,g}$ is the neutron flux. The maximum and averaged errors are respectively defined by:

$$\epsilon_{{\rm max}\ g} = \max_i \frac{|T_{i,g}^{\rm norm} - T_{i,g}^{\rm norm*}|}{T_{i,g}^{\rm norm*}}$$

and

$$\bar{\epsilon}_g = \frac{1}{N} \sum_i \left[ \frac{|T_{i,g}^{\rm norm} - T_{i,g}^{\rm norm*}|}{T_{i,g}^{\rm norm*}} \right]$$

where $T_{i,g}^{\rm norm*}$ is computed using the reference values (stored in *MACRO1*) and $N$ is the total number of regions in the MACROLIB.

The calling specifications are:

Table 27: Structure **(ERROR:)**

| |
|---|
| `ERROR:` *MACRO1 MACRO2* `::` [ `HREA` *hname* ] [ `NREG` *nreg* ] ; |

where

*MACRO1* — `character*12` name of the LCM object (type `L_MACROLIB`) containing the extended MACROLIB used to compute the reference reaction rates.

*MACRO2* — `character*12` name of the LCM object (type `L_MACROLIB`) containing the extended MACROLIB used to compute the approximate reaction rates.

`HREA` — keyword used to set the character name *hname*.

*hname* — character*8 name of the nuclear reaction used to compute the power map. By default, reaction `H-FACTOR` is used.

`NREG` — keyword used to set the *nreg* number.

*nreg* — integer number set to the number of regions used in statistics. By default, all available regions are used.

### 1.14 The `INIKIN:` module

The `INIKIN:` module is used to recover the steady-state solution and to initialize the kinetics parameters. The delayed neutron information can be provided directly from the input file or recovered from the MACROLIB data structure.

The initial presursor concentrations are obtained as a function of the strady-state solution. If $\phi_g(\boldsymbol{r}, t_0)$ is the initial flux in energy group $g$ divided by $k_{\text{eff}}$, the corresponding initial conditions of the precursors are obtained as

$$c_\ell(\boldsymbol{r}, t_0) = \frac{1}{\lambda_\ell} \sum_{h=1}^{G} \nu\Sigma^{\text{del}}_{\text{f}\ell,h}(\boldsymbol{r})\, \phi_h(\boldsymbol{r}, t_0); \quad \ell = 1, N_d. \tag{1.5}$$

where $\nu\Sigma^{\text{del}}_{\text{f}\ell,h}(\boldsymbol{r})$ is $\nu$ times the delayed macroscopic fission cross section in energy group $h$ for precursor group $\ell$.

The calling specifications are:

Table 28: Structure **(INIKIN:)**

*KINET* `:=` `INIKIN:` *MACRO TRACK SYST FLUX* `::` **(inikin_data)**

where

*KINET* `character*12` name of the LCM object (type `L_KINET`) to be created by the module.

*MACRO* `character*12` name of the LCM object (type `L_MACROLIB`) containing the MACROLIB information.

*TRACK* `character*12` name of the LCM object (type `L_TRACK`) containing the TRACKING information.

*SYST* `character*12` name of the LCM object (type `L_SYSTEM`) corresponding to MACROLIB *MACRO* and TRACKING *TRACK*.

*FLUX* `character*12` name of the LCM object (type `L_FLUX`) containing the initial steady-state solution.

**(inikin_data)** structure containing the data to module `INIKIN:` (see Sect. 1.14.1).

#### 1.14.1 Data input for module `INIKIN:`

Table 29: Structure **(inikin_data)**

[ `EDIT` *iprint* ]
[ `NGRP` *ngrp* ]
`NDEL` *ndg*
[ `BETA` (*beta*(i), i=1,*ndg*) ]
[ `LAMBDA` (*lambda*(i), i=1,*ndg*) ]
[ `CHID` ((*chid*(i), i=1,*ndg*), j=1,*ngrp*) ]

continued on next page

Structure **(inikin_data)** continued from last page

```
[ NORM { fnorm | MAX | POWER-INI power } ]
;
```

where

`EDIT` keyword used to set *iprint* index.

*iprint* integer index used to control the printing in module `INIKIN:`. =0 for no print; =1 for minimum printing (default value); larger values of *iprint* will produce increasing amounts of output.

`NGRP` keyword used to set the *ngrp* number. By default, this information is recovered from the solution object *FLUX*.

*ngrp* integer total number of energy groups.

`NDEL` keyword used to set the *ndg* number.

*ndg* integer total number of the delayed neutron groups.

`BETA` keyword used to indicate the reading of *beta* values from the input file. If these values are not provided, they should be recorded in the MACROLIB data structure.

*beta* real array containing the delayed neutron fractions for each delayed group.

`LAMBDA` keyword used to indicate the reading of *lambda* values from the input file. If these values are not provided, they should be recorded in the MACROLIB data structure.

*lambda* real array containing the precursors decay constants for each delayed group.

`CHID` keyword used to indicate the reading of *chid* values from the input file. If these values are not provided, they should be recorded in the MACROLIB data structure.

*chid* real array representing the delayed multigroup fission spectrum.

`NORM` keyword used to normalize the initial flux. By default, the flux is not normalized.

*fnorm* real normalization factor.

`MAX` keyword used to set the flux normalization factor to $1/f_{\max}$ where $f_{\max}$ is the maximum flux in the core.

`POWER-INI` keyword used to set the flux normalization factor to a given value of the initial power.

*power* real initial power in MW.

### 1.15 The `KINSOL:` module

The `KINSOL:` module is used to solve the space-time neutron kinetics equations at current time step of transient. Several implicit numerical schemes are available for this purpose. Consider first the differential equation for precursor concentrations:

$$\frac{\partial c_\ell(\boldsymbol{r},t)}{\partial t}+\lambda_\ell\, c_\ell(\boldsymbol{r},t)=\sum_{h=1}^{G}\nu\Sigma^{\rm del}_{{\rm f}\ell,h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t);\quad \ell=1,N_d. \tag{1.6}$$

Consider a solution between times $t_{n-1}$ and $t_n=t_{n-1}+\Delta t_n$. First, an analytic solution can be obtained by assuming a ramp variation of the fission reaction rates over time step $\Delta t_n$. This solution is written

$$\begin{aligned} c_\ell(\boldsymbol{r},t_n) &= c_\ell(\boldsymbol{r},t_{n-1})\,{\rm e}^{-\lambda_\ell\,\Delta t_n}+\frac{F_\ell(\boldsymbol{r},t_{n-1})}{\lambda_\ell}\left[\frac{1}{\lambda_\ell\,\Delta t_n}\left(1-{\rm e}^{-\lambda_\ell\,\Delta t_n}\right)-{\rm e}^{-\lambda_\ell\,\Delta t_n}\right]\\ &+ \frac{F_\ell(\boldsymbol{r},t_n)}{\lambda_\ell}\left[1-\frac{1}{\lambda_\ell\,\Delta t_n}\left(1-{\rm e}^{-\lambda_\ell\,\Delta t_n}\right)\right] \end{aligned}\tag{1.7}$$

where the delayed fission reaction rates are defined as

$$F_\ell(\boldsymbol{r},t_n)=\sum_{h=1}^{G}\nu\Sigma^{\rm del}_{{\rm f}\ell,h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_n)=\beta_\ell\sum_{h=1}^{G}\nu\Sigma_{{\rm f}h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_n). \tag{1.8}$$

An *implicit theta solution* is presented in Chapter 5 of Ref. 1. This solution is written

$$\begin{aligned} c_\ell(\boldsymbol{r},t_n) &= \left[\frac{1-(1-\Theta_{\rm p})\,\lambda_\ell\,\Delta t_n}{1+\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}\right]c_\ell(\boldsymbol{r},t_{n-1})+\frac{F_\ell(\boldsymbol{r},t_{n-1})}{\lambda_\ell}\left[\frac{(1-\Theta_{\rm p})\,\lambda_\ell\,\Delta t_n}{1+\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}\right]\\ &+ \frac{F_\ell(\boldsymbol{r},t_n)}{\lambda_\ell}\left[\frac{\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}{1+\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}\right] \end{aligned}\tag{1.9}$$

where $\Theta_{\rm p}$ is the theta-factor for precursors.

The fixed-source corresponding to the analytic solution for precursors is written

$$\begin{aligned} S^{\rm exact}_g(\boldsymbol{r},t_n) &= \frac{1}{V_{{\rm n},g}\,\Delta t_n}\,\phi_g(\boldsymbol{r},t_{n-1})+\sum_\ell\lambda_\ell\left[1-\Theta_{\rm f}+\Theta_{\rm f}\,{\rm e}^{-\lambda_\ell\,\Delta t_n}\right]\chi^{\rm del}_{\ell,g}(\boldsymbol{r})\,c_\ell(\boldsymbol{r},t_{n-1})\\ &+ (1-\Theta_{\rm f})\Big\{\boldsymbol{\nabla}\cdot\mathbb{D}_g(\boldsymbol{r})\boldsymbol{\nabla}\phi_g(\boldsymbol{r},t_{n-1})-\Sigma_{{\rm r}g}(\boldsymbol{r})\,\phi_g(\boldsymbol{r},t_{n-1})\\ &+ \sum_{\substack{h=1\\h\neq g}}^{G}\Sigma_{g\leftarrow h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_{n-1})+\chi^{\rm ss}_g(\boldsymbol{r})\,F(\boldsymbol{r},t_{n-1})\Big\}\\ &- \sum_\ell\left[1-\Theta_{\rm f}-\Theta_{\rm f}\left(\frac{1}{\lambda_\ell\,\Delta t_n}\left(1-{\rm e}^{-\lambda_\ell\,\Delta t_n}\right)-{\rm e}^{-\lambda_\ell\,\Delta t_n}\right)\right]\chi^{\rm del}_{\ell,g}(\boldsymbol{r})\,F_\ell(\boldsymbol{r},t_{n-1}) \end{aligned}\tag{1.10}$$

where the steady-state fission reaction rates are defined as

$$F(\boldsymbol{r},t_n)=\sum_{h=1}^{G}\nu\Sigma_{{\rm f}h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_n). \tag{1.11}$$

The fixed-source corresponding to the implicit theta solution is presented in Chapter 5 of Ref. 1 and is written

$$\begin{aligned} S^{\Theta}_g(\boldsymbol{r},t_n) &= \frac{1}{V_{{\rm n},g}\,\Delta t_n}\,\phi_g(\boldsymbol{r},t_{n-1})+\sum_\ell\lambda_\ell\left[1-\Theta_{\rm f}+\Theta_{\rm f}\,\frac{1-(1-\Theta_{\rm p})\,\lambda_\ell\,\Delta t_n}{1+\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}\right]\chi^{\rm del}_{\ell,g}(\boldsymbol{r})\,c_\ell(\boldsymbol{r},t_{n-1})\\ &+ (1-\Theta_{\rm f})\Big\{\boldsymbol{\nabla}\cdot\mathbb{D}_g(\boldsymbol{r})\boldsymbol{\nabla}\phi_g(\boldsymbol{r},t_{n-1})-\Sigma_{{\rm r}g}(\boldsymbol{r})\,\phi_g(\boldsymbol{r},t_{n-1})\\ &+ \sum_{\substack{h=1\\h\neq g}}^{G}\Sigma_{g\leftarrow h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_{n-1})+\chi^{\rm ss}_g(\boldsymbol{r})\,F(\boldsymbol{r},t_{n-1})\Big\}\\ &- \sum_\ell\left[1-\Theta_{\rm f}-\Theta_{\rm f}\,\frac{(1-\Theta_{\rm p})\,\lambda_\ell\,\Delta t_n}{1+\Theta_{\rm p}\,\lambda_\ell\,\Delta t_n}\right]\chi^{\rm del}_{\ell,g}(\boldsymbol{r})\,F_\ell(\boldsymbol{r},t_{n-1}). \end{aligned}\tag{1.12}$$

The flux equation at end-of-step is now presented. The equation corresponding to the analytic solution for precursors is written

$$\begin{aligned}
&\frac{1}{V_{\mathrm{n},g}\,\Delta t_n}\,\phi_g(\boldsymbol{r},t_n) - \Theta_\mathrm{f}\,\boldsymbol{\nabla}\cdot\mathbb{D}_g(\boldsymbol{r})\boldsymbol{\nabla}\phi_g(\boldsymbol{r},t_n) + \Theta_\mathrm{f}\,\Sigma_{\mathrm{r}g}(\boldsymbol{r})\,\phi_g(\boldsymbol{r},t_n)\\
&\quad = S_g^{\mathrm{exact}}(\boldsymbol{r},t_n) + \Theta_\mathrm{f}\sum_{\substack{h=1\\h\neq g}}^{G}\Sigma_{g\leftarrow h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_n)\\
&\quad + \Theta_\mathrm{f}\,\chi_g^{\mathrm{ss}}(\boldsymbol{r})\,F(\boldsymbol{r},t_n) - \Theta_\mathrm{f}\sum_\ell \chi_{\ell,g}^{\mathrm{del}}(\boldsymbol{r})\,\frac{1}{\lambda_\ell\,\Delta t_n}\left(1-\mathrm{e}^{-\lambda_\ell\,\Delta t_n}\right)F_\ell(\boldsymbol{r},t_n).
\end{aligned}\tag{1.13}$$

The equation corresponding to the implicit theta solution is presented in Chapter 5 of Ref. 1 and is written

$$\begin{aligned}
&\frac{1}{V_{\mathrm{n},g}\,\Delta t_n}\,\phi_g(\boldsymbol{r},t_n) - \Theta_\mathrm{f}\,\boldsymbol{\nabla}\cdot\mathbb{D}_g(\boldsymbol{r})\boldsymbol{\nabla}\phi_g(\boldsymbol{r},t_n) + \Theta_\mathrm{f}\,\Sigma_{\mathrm{r}g}(\boldsymbol{r})\,\phi_g(\boldsymbol{r},t_n)\\
&\quad = S_g^{\Theta}(\boldsymbol{r},t_n) + \Theta_\mathrm{f}\sum_{\substack{h=1\\h\neq g}}^{G}\Sigma_{g\leftarrow h}(\boldsymbol{r})\,\phi_h(\boldsymbol{r},t_n)\\
&\quad + \Theta_\mathrm{f}\,\chi_g^{\mathrm{ss}}(\boldsymbol{r})\,F(\boldsymbol{r},t_n) - \Theta_\mathrm{f}\sum_\ell \chi_{\ell,g}^{\mathrm{del}}(\boldsymbol{r})\,\frac{1}{1+\Theta_\mathrm{p}\,\lambda_\ell\,\Delta t_n}\,F_\ell(\boldsymbol{r},t_n).
\end{aligned}\tag{1.14}$$

The calling specifications are:

Table 30: Structure **(KINSOL:)**

| |
|---|
| *KINET* `:=` `KINSOL:` *KINET MACRO TRACK SYST* [ *MACRO_0 SYST_0* ] `::` **(kinsol_data)** |

where

*KINET* `character*12` name of the LCM object (type `L_KINET`) in modification mode.

*MACRO* `character*12` name of the LCM object (type `L_MACROLIB`) containing the MACROLIB information corresponding to the current time step of a transient.

*TRACK* `character*12` name of the LCM object (type `L_TRACK`) containing the TRACKING information.

*SYST* `character*12` name of the LCM object (type `L_SYSTEM`) corresponding to MACROLIB *MACRO* and TRACKING *TRACK*.

*MACRO_0* `character*12` name of the LCM object (type `L_MACROLIB`) containing the MACROLIB information corresponding to the beginning-of-step conditions in case a ramp variation of the cross sections in set. *Beginning-of-step conditions should not be confused with beginning-of-transient or initial conditions.* By default, a step variation is set where cross sections are assumed constant and given by *MACRO*.

*SYST_0* `character*12` name of the LCM object (type `L_SYSTEM`) corresponding to MACROLIB *MACRO_0* and TRACKING *TRACK*.

**(kinsol_data)** structure containing the data to module `KINSOL:` (see Sect. 1.15.1).

*1.15.1 Data input for module* `KINSOL:`

Table 31: Structure **(kinsol_data)**

```
[ EDIT iprint ]
 DELTA delta
 SCHEME FLUX { IMPLIC | CRANK | THETA ttflx }
 PREC { IMPLIC | CRANK | EXPON | THETA ttprc }
[ { VAR1 | ACCE } icl1 icl2 ]
[ EXTE [ maxout ] [ epsout ] ]
[ THER [ maxthr ] [ epsthr ] ]
[ ADI nadi ]
;
```

where

| | |
|---|---|
| `EDIT` | keyword used to set *iprint* index. |
| *iprint* | integer index used to control the printing in module `KINSOL:`. =0 for no print; =1 for minimum printing (default value); larger values of *iprint* will produce increasing amounts of output. |
| `DELTA` | keyword used to set the *delta* value. |
| *delta* | current time increment $\Delta t_n$ of transient. |
| `SCHEME` | keyword used to indicate the temporal numerical schemes. |
| `FLUX` | keyword used to select the temporal scheme for the fluxes equations. |
| `PREC` | keyword used to select the temporal scheme for the precursors equations. |
| `IMPLIC` | keyword used to indicate the full implicit temporal scheme. |
| `CRANK` | keyword used to indicate the Crank-Nicholson temporal scheme. |
| `EXPON` | keyword used to indicate the analytical integration scheme for precursors equations. |
| `THETA` | keyword used to indicate the general temporal scheme according to the *theta* method. |
| *ttflx* | value of *theta* parameter $\Theta_{\rm f}$ for the flux equations. This value should be greater than 0.5 and less than 1.0. |
| *ttprc* | value of *theta* parameter $\Theta_{\rm p}$ for the precursors equations. This value should be greater than 0.5 and less than 1.0. |
| `VAR1` | keyword used to set the parameters (*icl1* and *icl2*) of the symmetrical variational acceleration technique (SVAT). |
| `ACCE` | alias keyword for `VAR1`. |
| *icl1* | number of free outer iterations in a cycle of the SVAT. The default value is $icl1 = 3$. |
| *icl2* | number of accelerated outer iterations in a cycle of the SVAT. The default value is $icl2 = 3$. A convergence in free iterations is obtained by setting $icl1 = 200$ (or $icl1 = maxx0$) and $icl2 = 0$. |

`EXTE` keyword to specify that the control parameters for the external iteration are to be modified.

*maxout* maximum number of external iterations. The fixed default value is $maxout = 200$.

*epsout* convergence criterion for the external iterations. The fixed default value is $epsout = 1.0 \times 10^{-4}$. The outer iterations are stopped when the following criteria is reached:

$$\max_i |\Phi_i^{(k-1)} - \Phi_i^{(k)}| \ \leq \ epsout \times \max_i |\Phi_i^{(k)}|$$

where $\vec{\Phi}^{(k)} = \mathrm{col}\{\Phi_i^{(k)} \ ; \ i = 1, I\}$ is the product of the $B$ matrix times the unknown vector at the $k$-th outer iteration.

`THER` keyword to specify that the control parameters for the thermal iterations are to be modified.

*maxthr* maximum number of thermal iterations. The fixed default value is $maxthr = 0$ corresponding to no thermal iterations.

*epsthr* convergence criterion for the thermal iterations. The fixed default value is $epsthr = 1.0 \times 10^{-2}$.

`ADI` keyword used to set *nadi* in cases where Trivac is used.

*nadi* number of alternating direction implicit (ADI) inner iterations per outer iteration. The default value is $nadi = 1$. If this value causes a failure of the acceleration process, it is recommended that a larger value be tried. The optimal choice is generally the minimum value of *nadi* which allows a convergence in less than 75 outer iterations. $nadi = 1$ or $nadi = 2$ is generally the best choice for production-type calculations. The greater *nadi* is, the smaller the asymptotic convergence constant (ACC) becomes. Taking an arbitrary large value (e.g., $nadi = 20$) leads to numerical results identical to those obtained by inverting the system matrices at each outer iteration (at a prohibitive CPU cost). In this case, the ACC is almost equal to the dominance ratio of the iterative matrix. The default value is recovered in the state vector of the TRACKING object *TRACK*.

## 1.16 The `VAL:` module

The `VAL:` module supplies an interpolation of the flux in diffusion calculations for Cartesian geometries. The calling specifications are:

Table 32: Structure **(VAL:)**

| |
|---|
| *IFLU* `:=` `VAL:` *TRKNAM* *FLUNAM* `::` **(descval)** |

where

*IFLU* — `character*12` name of the INTERPFLUX data structure (`L_FVIEW` signature) where the interpolated flux distribution will be stored.

*TRKNAM* — `character*12` name of the read-only TRACKING data structure (`L_TRACK` signature) containing the tracking.

*FLUNAM* — `character*12` name of the read-only FLUXUNK data structure (`L_FLUX` signature) containing a transport solution.

**(descval)** — structure containing the input data to this module to compute interpolated flux (see Section 1.16.1).

### *1.16.1 Data input for module* `VAL:`

Table 33: Structure **(descval)**

| |
|---|
| [ `EDIT` *iprint* ]<br>[ `MODE` *imode* ]<br>`DIM` *dim* ($dxyz(i)$, $i = 1, dim$)<br>; |

where

`EDIT` — keyword used to modify the print level *iprint*.

*iprint* — integer index used to control the printing in module `VAL:`. =0 for no print; =1 for minimum printing (default value); larger values of *iprint* will produce increasing amounts of output.

`MODE` — keyword to specify the flux harmonic index *imode*.

*imode* — index of the flux harmonic recovered by the `VAL:` module if the `MONI` keyword was set in module `FLUD:` (see Sect. 1.9.1). By default, it is assumed that the `MONI` keyword was not used.

`DIM` — keyword to specify the number *dim*.

*dim* — number of dimension of the geometry.

*dxyz* mesh interval along each direction which is used to define the grid where the flux is interpolated.

# 2 EXAMPLES OF INPUT DATA FILES

## 2.1 IAEA-2D benchmark

The IAEA-2D benchmark is defined in Refs. 3,18 and its geometry is represented in Fig. 11. Here, it is solved using a parabolic variational collocation method without mesh splitting of the elements:

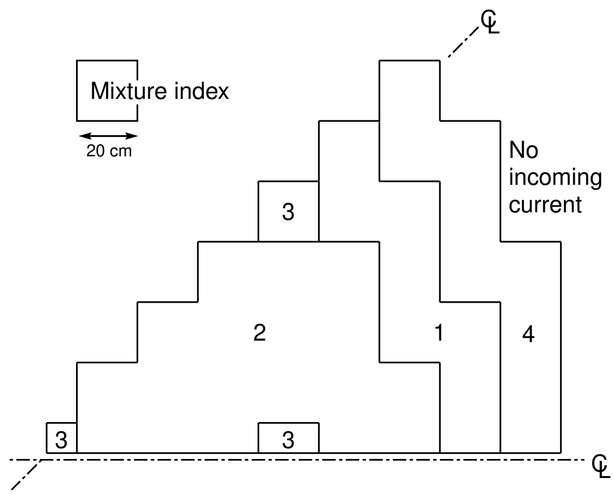


Figure 11: Description of the IAEA-2D benchmark.

```
LINKED_LIST IAEA MACRO TRACK SYSTEM FLUX EDIT ;
MODULE GEO: MAC: TRIVAT: TRIVAA: FLUD: OUT: END: ;
*
IAEA := GEO: :: CAR2D 9 9
           EDIT 2
           X- DIAG X+ VOID
           Y- SYME Y+ DIAG
           MIX  3 2 2 2 3 2 2 1 4
                  2 2 2 2 2 2 1 4
                    2 2 2 2 1 1 4
                      2 2 2 1 4 4
                        3 1 1 4 0
                          1 4 4 0
                            4 0 0
                              0 0
                                0
           MESHX 0.0 20.0 40.0 60.0 80.0 100.0 120.0 140.0 160.0 180.0
           ;
MACRO := MAC: ::
 EDIT 2 NGRO 2 NMIX 4
 READ INPUT
 MIX     1
      DIFF  1.500E+00  4.0000E-01
     TOTAL  3.012E-02  8.0032E-02
    NUSIGF  0.000E+00  1.3500E-01
  H-FACTOR  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     2
      DIFF  1.500E+00  4.0000E-01
```

```
     TOTAL  3.012E-02  8.5032E-02
    NUSIGF  0.000E+00  1.3500E-01
  H-FACTOR  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     3
      DIFF  1.500E+00  4.00000E-01
     TOTAL  3.012E-02  1.30032E-01
    NUSIGF  0.000E+00  1.35000E-01
  H-FACTOR  0.000E+00  1.35000E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     4
      DIFF  2.000E+00  3.0000E-01
     TOTAL  4.016E-02  1.0024E-02
      SCAT  1 1 0.0 2 2 0.0 0.4E-01
 ;
TRACK := TRIVAT: IAEA ::
      TITLE 'IAEA-2D BENCHMARK'
      MAXR 81 PRIM 2 ;
SYSTEM := TRIVAA: MACRO TRACK :: ;
FLUX := FLUD: SYSTEM ::
      EDIT 2 ;
EDIT := OUT: FLUX ::
       EDIT 2 INTG
       1  2  3  4  5  6  7  8  0
          9 10 11 12 13 14 15  0
            16 17 18 19 20 21  0
               22 23 24 25  0  0
                  26 27 28  0  0
                     29  0  0  0
                         0  0  0
                            0  0
                               0
       ;
END: ;
```

### 2.2 Biblis-2D benchmark

The rods-withdrawn configuration of the Biblis-2D benchmark is defined in Ref. 3 and its geometry is represented in Fig. 12. Here, it is solved using a parabolic variational collocation method without mesh splitting of the elements:

```
LINKED_LIST BIBLIS MACRO TRACK SYSTEM FLUX EDIT ;
MODULE GEO: MAC: TRIVAT: TRIVAA: FLUD: OUT: END: ;
*
BIBLIS := GEO: :: CAR2D 9 9
           EDIT 2
           X- DIAG X+ VOID
           Y- SYME Y+ DIAG
       MIX 1 8 2 6 1 7 1 4 3
             1 8 2 8 1 1 4 3
               1 8 2 7 1 4 3
                 2 8 1 8 4 3
                   2 5 4 3 3
                     4 4 3 0
                       3 3 0
                         0 0
```

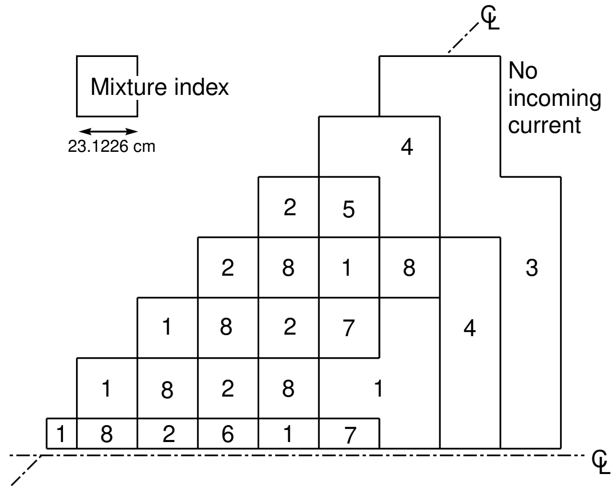


Figure 12: Description of the Biblis-2D benchmark, rods-withdrawn configuration.

```
                      0
     MESHX 0.0 23.1226 46.2452 69.3678 92.4904 115.613 138.7356
           161.8582 184.9808 208.1034
     ;
MACRO := MAC: ::
 EDIT 2 NGRO 2 NMIX 8
 READ INPUT
 MIX     1
      DIFF  1.436000E+00  3.635000E-01
     TOTAL  2.725820E-02  7.505800E-02
    NUSIGF  5.870800E-03  9.606700E-02
  H-FACTOR  2.376800E-03  3.889400E-02
      SCAT  1 1 0.0 2 2 0.0  1.775400E-02
 MIX     2
      DIFF  1.436600E+00  3.636000E-01
     TOTAL  2.729950E-02  7.843600E-02
    NUSIGF  6.190800E-03  1.035800E-01
  H-FACTOR  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.762100E-02
 MIX     3
      DIFF  1.320000E+00  2.772000E-01
     TOTAL  2.576220E-02  7.159600E-02
      SCAT  1 1 0.0 2 2 0.0  2.310600E-02
 MIX     4
      DIFF  1.438900E+00  3.638000E-01
     TOTAL  2.746400E-02  9.140800E-02
    NUSIGF  7.452700E-03  1.323600E-01
  H-FACTOR  3.017300E-03  5.358700E-02
      SCAT  1 1 0.0 2 2 0.0  1.710100E-02
 MIX     5
      DIFF  1.438100E+00  3.665000E-01
     TOTAL  2.729300E-02  8.482800E-02
    NUSIGF  6.190800E-03  1.035800E-01
  H-FACTOR  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.729000E-02
 MIX     6
```

```
      DIFF  1.438500E+00  3.665000E-01
     TOTAL  2.732400E-02  8.731400E-02
    NUSIGF  6.428500E-03  1.091100E-01
  H-FACTOR  2.602600E-03  4.417400E-02
      SCAT  1 1 0.0 2 2 0.0  1.719200E-02
 MIX     7
      DIFF  1.438900E+00  3.679000E-01
     TOTAL  2.729000E-02  8.802400E-02
    NUSIGF  6.190800E-03  1.035800E-01
  H-FACTOR  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.712500E-02
 MIX     8
      DIFF  1.439300E+00  3.680000E-01
     TOTAL  2.732100E-02  9.051000E-02
    NUSIGF  6.428500E-03  1.091100E-01
  H-FACTOR  2.602600E-03  4.417400E-02
      SCAT  1 1 0.0 2 2 0.0  1.702700E-02
       ;
TRACK := TRIVAT: BIBLIS ::
      TITLE 'BIBLIS BENCHMARK'
      EDIT 5 MAXR 81 PRIM 2 ;
SYSTEM := TRIVAA: MACRO TRACK ::
      EDIT 5 ;
FLUX := FLUD: SYSTEM ::
      EDIT 2 ;
EDIT := OUT: FLUX ::
       EDIT 2 INTG
       1  2  3  4  5  6  7  8  0
          9 10 11 12 13 14 15  0
            16 17 18 19 20 21  0
               22 23 24 25 26  0
                  27 28 29  0  0
                     30 31  0  0
                         0  0  0
                            0  0
                               0
       ;
END: ;
```

### 2.3 IAEA-3D benchmark

The IAEA-3D benchmark is defined in Ref. 18 and its geometry is represented in Fig. 13. Here, it is solved using a cubic mixed-dual method with mesh splitting of the second axial plane:

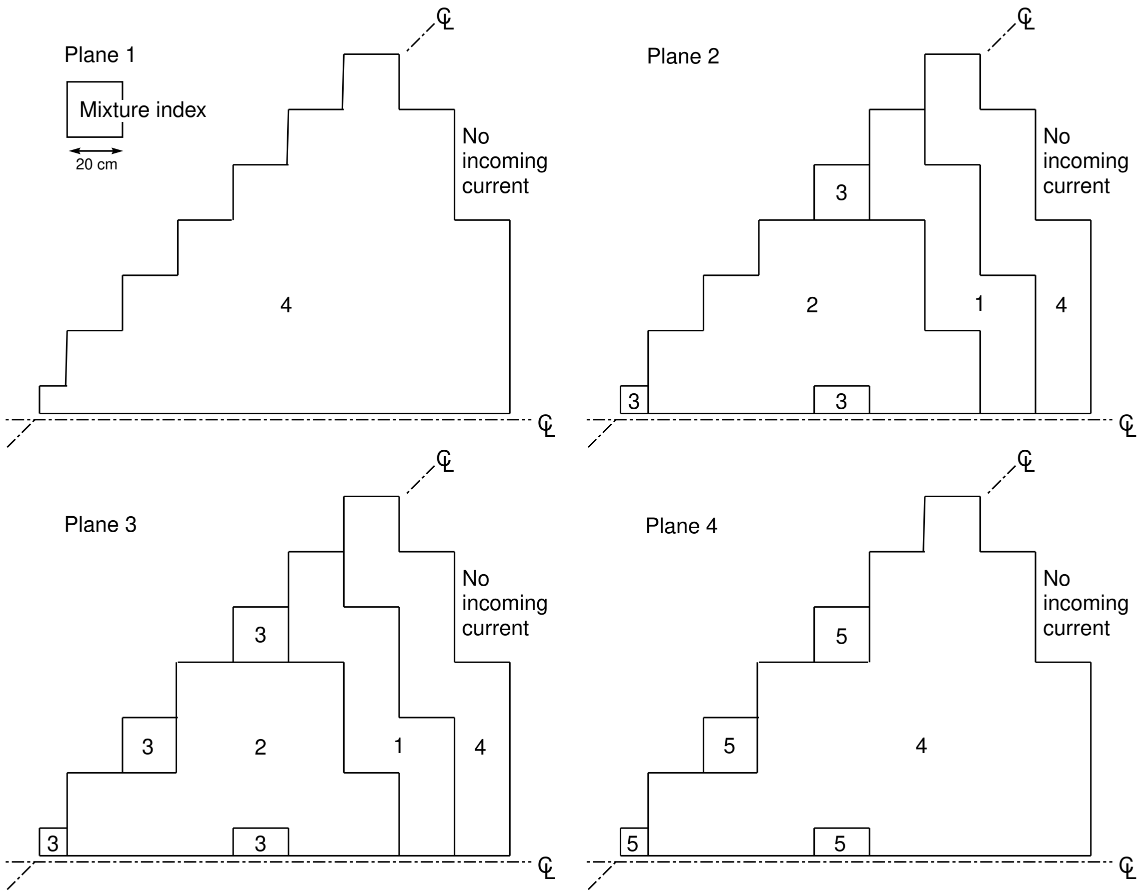


Figure 13: Description of the IAEA-3D benchmark.

```
LINKED_LIST IAEA3D MACRO TRACK SYSTEM FLUX EDIT ;
MODULE GEO: MAC: TRIVAT: TRIVAA: FLUD: OUT: END: ;
*
IAEA3D := GEO: :: CAR3D 9 9 4
          EDIT 2
          X- DIAG  X+ VOID
          Y- SYME  Y+ DIAG
          Z- VOID  Z+ VOID
          MESHX 0.0 20.0 40.0 60.0 80.0 100.0 120.0 140.0 160.0 180.0
          MESHZ 0.0 20.0 280.0 360.0 380.0
          SPLITZ 1 2 1 1
          (* PLANE NB 1 *)
          MIX 4 4 4 4 4 4 4 4 4
                4 4 4 4 4 4 4 4
                  4 4 4 4 4 4 4
                    4 4 4 4 4 4
                      4 4 4 4 0
                        4 4 4 0
                          4 0 0
```

```
                              0 0
                                0
              (* PLANE NB 2 *)
              3 2 2 2 3 2 2 1 4
                2 2 2 2 2 2 1 4
                  2 2 2 2 1 1 4
                    2 2 2 1 4 4
                      3 1 1 4 0
                        1 4 4 0
                          4 0 0
                            0 0
                              0
              (* PLANE NB 3 *)
              3 2 2 2 3 2 2 1 4
                2 2 2 2 2 2 1 4
                  3 2 2 2 1 1 4
                    2 2 2 1 4 4
                      3 1 1 4 0
                        1 4 4 0
                          4 0 0
                            0 0
                              0
              (* PLANE NB 4 *)
              5 4 4 4 5 4 4 4 4
                4 4 4 4 4 4 4 4
                  5 4 4 4 4 4 4
                    4 4 4 4 4 4
                      5 4 4 4 0
                        4 4 4 0
                          4 0 0
                            0 0
                              0
           ;
MACRO := MAC: ::
 EDIT 2 NGRO 2 NMIX 5
 READ INPUT
 MIX     1
      DIFF  1.500E+00  4.0000E-01
     TOTAL  3.000E-02  8.0000E-02
    NUSIGF  0.000E+00  1.3500E-01
  H-FACTOR  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     2
      DIFF  1.500E+00  4.0000E-01
     TOTAL  3.000E-02  8.5000E-02
    NUSIGF  0.000E+00  1.3500E-01
  H-FACTOR  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     3
      DIFF  1.500E+00  4.00000E-01
     TOTAL  3.000E-02  1.30000E-01
    NUSIGF  0.000E+00  1.35000E-01
  H-FACTOR  0.000E+00  1.35000E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     4
```

```
      DIFF  2.000E+00  3.0000E-01
     TOTAL  4.000E-02  1.0000E-02
      SCAT  1 1 0.0 2 2 0.0 0.4E-01
 MIX     5
      DIFF  2.000E+00  3.0000E-01
     TOTAL  4.000E-02  5.5000E-02
      SCAT  1 1 0.0 2 2 0.0 0.4E-01
 ;
TRACK := TRIVAT: IAEA3D ::
      TITLE 'TEST IAEA 3D'
      EDIT 5 MAXR 405 DUAL 3 1 ;
SYSTEM := TRIVAA: MACRO TRACK ::
      EDIT 5 ;
FLUX := FLUD: SYSTEM ::
      EDIT 2 ;
EDIT := OUT: FLUX ::
       EDIT 2 INTG
       (* PLANE NB 1 *)
       0  0  0  0  0  0  0  0  0
          0  0  0  0  0  0  0  0
             0  0  0  0  0  0  0
                0  0  0  0  0  0
                   0  0  0  0  0
                      0  0  0  0
                         0  0  0
                            0  0
                               0
       (* PLANE NB 2 *)
       1  2  3  4  5  6  7  8  0
          9 10 11 12 13 14 15  0
            16 17 18 19 20 21  0
               22 23 24 25  0  0
                  26 27 28  0  0
                     29  0  0  0
                         0  0  0
                            0  0
                               0
       (* PLANE NB 3 *)
       30 31 32 33 34 35 36 37  0
          38 39 40 41 42 43 44  0
             45 46 47 48 49 50  0
                51 52 53 54  0  0
                   55 56 57  0  0
                      58  0  0  0
                          0  0  0
                             0  0
                                0
       (* PLANE NB 4 *)
       0  0  0  0  0  0  0  0  0
          0  0  0  0  0  0  0  0
             0  0  0  0  0  0  0
                0  0  0  0  0  0
                   0  0  0  0  0
                      0  0  0  0
                         0  0  0
```

```
                              0   0
                                  0
       ;
END: ;
```

### 2.4 S30 hexagonal benchmark in 2-D

The S30 hexagonal benchmark in 2-D is defined in Ref. 14. Its geometry is represented in Fig. 14. Here, it is solved using a mesh centered finite difference method without mesh splitting of the hexagonal elements:

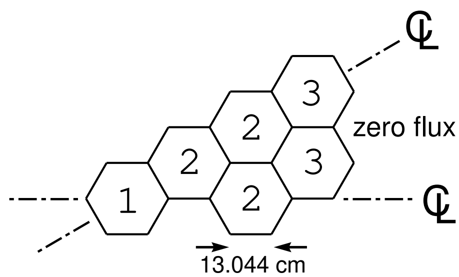


Figure 14: Description of the S30 hexagonal benchmark.

```
LINKED_LIST HEX MACRO TRACK SYSTEM FLUX EDIT ;
MODULE GEO: MAC: TRIVAT: TRIVAA: FLUD: OUT: END: ;
*
HEX := GEO: :: HEX   6
       EDIT 2
       HBC S30 ZERO
       SIDE 13.044
       MIX
       1
       2
       2  2
       3  3
       ;
MACRO := MAC: ::
 EDIT 2 NGRO 2 NMIX 3
 READ INPUT
 MIX     1
      DIFF  1.5E+00  4.00E-01
     TOTAL  3.0E-02  1.30E-01
    NUSIGF  0.0E+00  1.35E-01
  H-FACTOR  0.0E+00  1.35E-01
      SCAT  1 1 0.0 2 2 0.0  0.2E-01
 MIX     2
      DIFF  1.5E+00  4.00E-01
     TOTAL  3.0E-02  8.50E-02
    NUSIGF  0.0E+00  1.35E-01
  H-FACTOR  0.0E+00  1.35E-01
      SCAT  1 1 0.0 2 2 0.0  0.2E-01
 MIX     3
      DIFF  2.0E+00  3.0E-01
     TOTAL  4.0E-02  1.0E-02
      SCAT  1 1 0.0 2 2 0.0  0.4E-01
```

```
 ;
TRACK := TRIVAT: HEX ::
      TITLE 'S30 HEXAGONAL BENCHMARK IN 2-D.'
      EDIT 5 MAXR 50 MCFD (* IELEM= *) 1 (* ISPLH= *) 1 ;
SYSTEM := TRIVAA: MACRO TRACK ::
      EDIT 5 ;
FLUX := FLUD: SYSTEM ::
      EDIT 2 ;
EDIT := OUT: FLUX ::
       EDIT 2 INTG IN ;
END: ;
```

### 2.5 LMW benchmark in 2-D

The LMW benchmark in 2-D is a space-time kinetics problem introduced by Greenman[19] and used by Monier[13]. Its geometry is represented in Fig. 15. Here, it is solved using a parabolic nodal collocation method with 2×2 mesh splitting of each element. A reactivity transient is induced by the rapid withdrawal of the control rod in material mixture 6. The control rod is removed in 26.7 s, causing a negative ramp variation in total cross section.

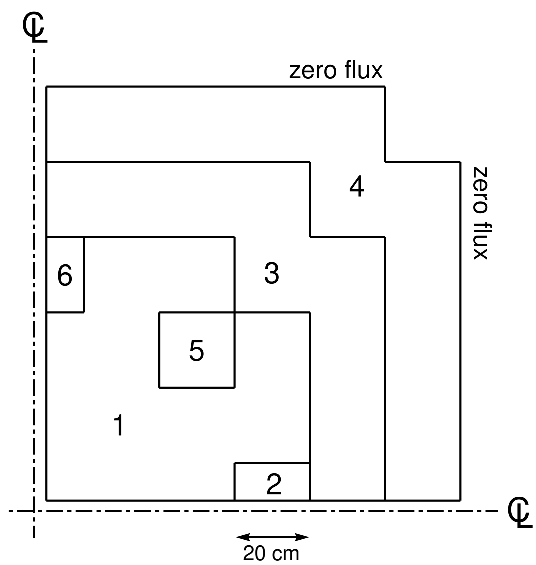


Figure 15: Description of the LMW benchmark in 2-D.

```
*----
*  TEST CASE LMW 2D
*
*  REF: G. Greenman, "A Quasi-Static Flux Synthesis Temporal Integration
*       Scheme for an Analytic Nodal Method," Nuclear Engineer's Thesis,
*       Massachusetts Institute of Technology, Department of Nuclear
*       Engineering (May 1980).
*
*----
*  Define STRUCTURES and MODULES used
*----
LINKED_LIST LMW TRACK MACRO1 SYSTEM1 MACRO2 SYSTEM2 FLUX KINET ;
MODULE GEO: MAC: TRIVAT: TRIVAA: FLUD: INIKIN: KINSOL: GREP: DELETE:
       END: ;
```

```
REAL fnorm sigt1 sigt2 ;
REAL TIME := 0.0 ;
PROCEDURE assertS assertS2 ;
*
LMW := GEO: :: CAR2D 6 6
      X- REFL X+ ZERO
      Y- REFL Y+ ZERO
      MIX 1 1 1 2 3 4
          1 1 1 1 3 4
          1 1 5 1 3 4
          6 1 1 3 3 4
          3 3 3 3 4 4
          4 4 4 4 4 0
      MESHX  0.0 10. 30. 50. 70. 90. 110.
      MESHY  0.0 10. 30. 50. 70. 90. 110.
      SPLITX 2 2 2 2 2 2
      SPLITY 2 2 2 2 2 2
      ;
MACRO1 := MAC: ::
 EDIT 0 NGRO 2 NMIX 6
 READ INPUT
 MIX     1
      DIFF  1.423910E+00  3.563060E-01
     TOTAL  2.795756E-02  8.766216E-02
    NUSIGF  6.477691E-03  1.127328E-01
  H-FACTOR  2.591070E-03  4.509310E-02
      SCAT  1 1 0.0 2 2 0.0  0.175555E-01
      OVERV 0.800E-07  4.000E-06
 MIX     2
      DIFF  1.423910E+00  3.563060E-01
     TOTAL  2.850756E-02  9.146219E-02
    NUSIGF  6.477691E-03  1.127328E-01
  H-FACTOR  2.591070E-03  4.509310E-02
      SCAT  1 1 0.0 2 2 0.0  0.175555E-01
      OVERV 0.800E-07  4.000E-06
 MIX     3
      DIFF  1.425610E+00  3.505740E-01
     TOTAL  2.817031E-02  9.925634E-02
    NUSIGF  7.503282E-03  1.378004E-01
  H-FACTOR  3.001310E-03  5.512106E-02
      SCAT  1 1 0.0 2 2 0.0  0.171777E-01
      OVERV 0.800E-07  4.000E-06
 MIX     4
      DIFF  1.634220E+00  2.640020E-01
     TOTAL  3.025750E-02  4.936351E-02
      SCAT  1 1 0.0 2 2 0.0  0.275969E-01
      OVERV 0.800E-07  4.000E-06
 MIX     5
      DIFF  1.423910E+00  3.563060E-01
     TOTAL  2.795756E-02  8.766216E-02
    NUSIGF  6.477691E-03  1.127328E-01
  H-FACTOR  2.591070E-03  4.509310E-02
      SCAT  1 1 0.0 2 2 0.0  0.175555E-01
      OVERV 0.800E-07  4.000E-06
 MIX     6
```

```
      DIFF  1.423910E+00  3.563060E-01
     TOTAL  2.850756E-02  9.146217E-02
    NUSIGF  6.477691E-03  1.127328E-01
  H-FACTOR  2.591070E-03  4.509310E-02
      SCAT  1 1 0.0 2 2 0.0  0.175555E-01
      OVERV 0.800E-07  4.000E-06
 ;
TRACK := TRIVAT: LMW ::
      TITLE 'LMW 2-D BENCHMARK'
      EDIT 1 MAXR 144 MCFD 2 ;
SYSTEM1 := TRIVAA: MACRO1 TRACK ::
      EDIT 1 UNIT ;
FLUX := FLUD: SYSTEM1 TRACK ::
      EDIT 1 EXTE 5.0E-7 ;
assertS FLUX :: 'K-EFFECTIVE' 1 1.014803 ;
*----
*  Crank-Nicholson space-time kinetics
*----
EVALUATE TIME := 0.0 ;
KINET := INIKIN: MACRO1 TRACK SYSTEM1 FLUX :: EDIT 1
      NDEL 6
      BETA   0.000247 0.0013845 0.001222 0.0026455 0.000832 0.000169
      LAMBDA 0.0127 0.0317 0.115 0.311 1.40 3.87
      CHID   1.0 1.0 1.0 1.0 1.0 1.0
             0.0 0.0 0.0 0.0 0.0 0.0
      NORM POWER-INI 1.0E4 ;
EVALUATE sigt1 := 2.850756E-02 ;
EVALUATE sigt2 := 9.146217E-02  ;
WHILE TIME 26.7 <= DO
  EVALUATE sigt1 := sigt1 5.5E-4 0.1 26.7 / * - ;
  EVALUATE sigt2 := sigt2 3.8E-3 0.1 26.7 / * - ;
  MACRO2 := MAC: MACRO1 ::
      EDIT 0
      READ INPUT
      MIX     6
         TOTAL  <<sigt1>>  <<sigt2>>
      ;
  SYSTEM2 := TRIVAA: MACRO2 TRACK ::
      EDIT 1 UNIT ;
  KINET := KINSOL: KINET MACRO2 TRACK SYSTEM2 MACRO1 SYSTEM1 ::
      EDIT 5  DELTA 0.1
      SCHEME  FLUX CRANK PREC CRANK EXTE 1.0E-6 ;
  GREP: KINET :: GETVAL 'TOTAL-TIME' 1 >>TIME<< ;
  ECHO "TIME=" TIME "S" "sigt=" sigt1 sigt2 ;
  IF TIME 1.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 1.986270E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.095509E-01 ;
    assertS2 KINET :: 'E-POW'     1 1.008753E+04 ;
  ELSEIF TIME 5.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 2.090369E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.097266E-01 ;
    assertS2 KINET :: 'E-POW'     1 1.063990E+04 ;
  ELSEIF TIME 10.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 2.305455E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.104699E-01 ;
```

```
    assertS2 KINET :: 'E-POW'     1 1.176902E+04 ;
  ELSEIF TIME 15.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 2.641221E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.121002E-01 ;
    assertS2 KINET :: 'E-POW'     1 1.352433E+04 ;
  ELSEIF TIME 20.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 3.157370E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.150681E-01 ;
    assertS2 KINET :: 'E-POW'     1 1.621938E+04 ;
  ELSEIF TIME 25.0 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 3.971426E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.200883E-01 ;
    assertS2 KINET :: 'E-POW'     1 2.047011E+04 ;
  ELSEIF TIME 26.7 - ABS 1.0E-3 < THEN
    assertS2 KINET :: 'CTRL-FLUX' 1 4.351272E+02 ;
    assertS2 KINET :: 'CTRL-PREC' 1 1.224600E-01 ;
    assertS2 KINET :: 'E-POW'     1 2.245449E+04 ;
  ENDIF ;
  MACRO1 SYSTEM1 := DELETE: MACRO1 SYSTEM1 ;
  MACRO1 := MACRO2 ;
  SYSTEM1 := SYSTEM2 ;
  MACRO2 SYSTEM2 := DELETE: MACRO2 SYSTEM2 ;
ENDWHILE ;
ECHO "test lmw2D completed" ;
```

## References

[1] A. HÉBERT, *Applied Reactor Physics*, Presses Internationales Polytechnique, ISBN 978-2-553-01436-9, 424 p., Montréal, 2009.

[2] A. HÉBERT, "A Programmer's Guide for the GAN Generalized Driver, FORTRAN-77 version," Report IGE-158, École Polytechnique de Montréal, Institut de Génie Nucléaire (December 1994).

[3] A. HÉBERT, "Application of the Hermite Method for Finite Element Reactor Calculations," *Nucl. Sci. Eng.*, **91**, 34 (1985).

[4] A. HÉBERT, "Variational Principles and Convergence Acceleration Strategies for the Neutron Diffusion Equation," *Nucl. Sci. Eng.*, **91**, 414 (1985).

[5] A. HÉBERT, "Preconditioning the Power Method for Reactor Calculations," *Nucl. Sci. Eng.*, **94**, 1 (1986).

[6] A. HÉBERT, "Development of the Nodal Collocation Method for Solving the Neutron Diffusion Equation," *Ann. Nucl. Energy*, **14**, 527 (1987).

[7] A. HÉBERT, "TRIVAC, A Modular Diffusion Code for Fuel Management and Design Applications," *Nucl. J. of Canada*, Vol. 1, No. 4, 325 (1987).

[8] A. HÉBERT, "Application of a Dual Variational Formulation to Finite Element Reactor Calculations," *Ann. nucl. Energy*, **20**, 823 (1993).

[9] A. HÉBERT, "The Search for Superconvergence in Spherical Harmonics Approximations," *Nucl. Sci. Eng.*, **154**, 134 (2006).

[10] A. HÉBERT, "Mixed-dual implementations of the of the simplified $P_n$ method," *Ann. nucl. Energy*, **37**, 498 (2010).

[11] J. H. WILKINSON, "The Algebraic Eigenvalue Problem," *Clarendon Press*, Oxford (1965).

[12] R. ROY, Private communication.

[13] A. MONIER, "Application of the Collocation Technique to the Spatial Discretization of the Generalized Quasistatic Method for Nuclear Reactors," Ph. D. Thesis, École Polytechnique de Montréal, Institut de Génie Énergétique (December 1991).

[14] A. BENABOUD, "Résolution de l'équation de la diffusion neutronique pour une géométrie hexagonale," Ph. D. Thesis, École Polytechnique de Montréal, Institut de Génie Énergétique (December 1992).

[15] A. HÉBERT, "A Raviart–Thomas–Schneider solution of the diffusion equation in hexagonal geometry", *Ann. nucl. Energy*, **35**, 363 (2008).

[16] W. H. PRESS, S. A. TEUKOLSKY, W. T. VETTERLING and B. P. FLANNERY, "Numerical Recipes in FORTRAN," Second Edition, Chapter 16, Cambridge University Press (1992).

[17] J. J. LAUTARD, S. LOUBIÈRE and C. FEDON-MAGNAUD, "CRONOS, a Computational Modular System for Neutronic Core Calculations," Proc. International Atomic Energy Agency Specialists Mtg. on Advanced Calculational Methods for Power Reactors, Cadarache, France, September 1990.

[18] "Argonne Code Center: Benchmark Problem Book," ANL-7416, Supp. 2, ID11-A2, Argonne National Laboratory (1977).

[19] G. GREENMAN, "A Quasi-Static Flux Synthesis Temporal Integration Scheme for an Analytic Nodal Method," Nuclear Engineer's Thesis, Massachusetts Institute of Technology, Department of Nuclear Engineering (May 1980).

# Index