A USER GUIDE FOR DONJON
VERSION 2.01 - DATE 00/05/15

By

E. Varin, A. Hébert, R. Roy and J. Koclas

IGE-208
École Polytechnique de Montréal
Institut de génie nucléaire
May 2000

A User Guide for DONJON
Version 2.01 - Date 00/05/15

by

E. Varin, A. Hébert, R. Roy and J. Koclas

Institut de Génie Nucléaire
École Polytechnique de Montréal
Montréal, Québec, H3C 3A7, CANADA

ABSTRACT

DONJON is the name of the software resulting from the linking of XSIMUL[1] and TRIVAC-3[2] around the GAN generalized driver.[3] It supports insertion of loops when calling modular parts in order to carry out sensitivity analysis, without any recoding. Users can define criticality procedures by varying essential core parameters, such as a poison concentration or controller levels for which material cross section data may be obtained from the usual cell codes. All the GAN generalized driver facilities are available in DONJON. Examples of full core procedures are given.

SUMMARY

The simulation code DONJON is based on static and kinetics diffusion equation resolution in a modular environment provided by the GAN generalized driver[3]. As a package of modules, DONJON provides users with flexibility in the input data. But a certain order must be respect if the user hopes to find a correct solution to his problem.

Different sequences of module calls will be given here to help the user to find the adequate modules he needs. Fuel map and devices can be used together in mixing the two lists of module calls given in the following.

From a simple problem definition, the following modules must be called to obtain a flux solution:

- **GEOD:** : Generate a geometry object
- **MACD:** or **CRE:** : Generate a macrolib object for nuclear properties of the problem
- **TRIVAT:** (or **BIVADT:** one for 2-D calculations) : Generate a "tracking" object for spatial discretization
- **TRIVAA:** (or **BIVACA:**) : Generate a system object for assembled system matrices
- **FLUD:** or **IQS:** : Generate a flux object containing the diffusion equation solutions.

When the problem to solve contains device movement or position modification, the process to obtain a flux solution is the same as before but it requires new objects and new access to them. The module calls must be as:

- **GEOD:** : Generate a geometry object
- **USPLIT:** : Generate an index object to access properties from geometry.
- **MACD:** or **CRE:** : Generate a macrolib object for nuclear properties of the problem
- **INIDEV:** : Generate a device object for device positioning and property reference.
- **INPROC:** : Generate a process object for device movement.
- **INIMAC:** : Generate an extended macrolib object over each region of the geometry.
- **LINKDS:** : Set device positions.
- **NEWMAC:** : Generate a macrolib object containing properties corrected by device positions.
- **TRIVAT:** (or **BIVADT:** one for 2-D calculations) : Generate a "tracking" object for spatial discretization
- **TRIVAA:** (or **BIVACA:**) : Generate a system object for assembled system matrices
- **FLUD:** or **IQS:** : Generate a flux object containing the diffusion equation solutions.

When representing fuel evolution in the core, a fuel map object is needed and its access to proper address fuel burnup-dependent properties. The standard module calls for such as case are:

- **GEOD:** : Generate a geometry object
- **MACD:** or **CRE:** : Generate a macrolib object for nuclear properties independent of burnup.
- **XSCONS:** : Generate a table object for fuel burnup dependent properties.
- **AFM:** : Generate a macrolib object for specific local parameter set for fuel as well as for moderator properties.
- **INIRES:** : Generate a fuel map object for fuel lattice information.
- **REFRES:** : Modify fuel map and index object to properly address fuel properties
- **CRE:** : Generate a macrolib object for fuel properties at specific burnup.
- **INIMAC:** : Generate an extended macrolib object over each region of the geometry.
- **TRIVAT:** (or **BIVADT:** one for 2-D calculations) : Generate a "tracking" object for spatial discretization
- **TRIVAA:** (or **BIVACA:**) : Generate a system object for assembled system matrices
- **FLUD:** or **IQS:** : Generate a flux object containing the diffusion equation solutions.

Data are transferred to modules in objects. These objects have a particular structure depending of their SIGNATURE. These structures are fixed and can be used to retrieve precise information. The contain of all the objects defined and used in DONJON is documented in Report IGE-226, Ref. [5].

CONTENTS

List of Figures

1. GENERAL SPECIFICATIONS OF THE DONJON INPUT

1.1 Syntactic rules for input data specifications

The input data to any module is read in free format using the subroutine `REDGET`. CLE-2000[4] variables can also be used. The rules for specifying the input data are given in this section. This user's 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 `KEY WORDS` used as key words;
- 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

DONJON is built around the GAN generalized driver.[3] The input data must follow the calling specifications of this driver which are given below:

Structure **(driver)**

[`PARAMETER` [[*STRNAME*]] ;]
[`LINKED_LIST` [[*STRNAME*]] ;]
[`XSM_FILE` [[*STRNAME*]] ;]
[`SEQ_BINARY` [[*STRNAME*]] ;]
[`SEQ_ASCII` [[*STRNAME*]] ;]
[`MODULE` [[*MODNAME*]] ;]
[`PROCEDURE` [[*PROCNAME*]] ;]
[[**(specif)**]]
`END: ;`

where

PARAMETER keyword to specify the list of parameters in a procedure file.[3]

STRNAME `Character*12` name of a DONJON data structure. The list of the DONJON data structure will be given in Section (1.3).

LINKED_LIST keyword to specify wich data structures will be stored on linked lists.

XSM_FILE keyword to specify wich data structures will be stored on XSM files.

SEQ_BINARY keyword to specify wich data structures will be stored on sequential BINARY files.

SEQ_ASCII keyword to specify wich data structures will be stored on sequential ASCII files.

MODULE keyword to specify the lists of modules to be used in this DONJON execution.

MODNAME `Character*12` name of a DONJON module.

PROCEDURE keyword to specify the lists of procedure files that will be used in this DONJON execution.

PROCNAME `Character*12` name of a procedure file.

(specif) Input specifications for a single module. Specifications for DONJON modules will be given in the following sections.

Distinction must be made between an input data file and its calling procedures. The input data always begin with the declaration of each linked list, XSM file, sequential (binary or ASCII) file that will be required by the following modules. This is followed by the declaration of the modules and the procedures actually used in the input data deck.

The procedure input data begin with the declaration of each calling parameter, that will be used in it, respecting the order given at the procedure call. Only linked lists, XSM files and CLE-2000 variables can be defined as procedure parameters. The following declarations are the same than those needed in the input data file. In both input data files and procedures, CLE-2000 variables can be declared. The following data describe module call specification, in the format of the GAN generalized driver. Modules and procedures communicate with each other through data structures whose specifications are given in Report IGE-226.[5]

The input data and procedures always end with a call to the `END:` module. Note that the file names are truncated to seven characters on MVS due to a constraint of this operating system.

1.3 The DONJON Data Structures

The transfer of informations between DONJON modules is allowed by well-defined data structures. These structures also called objects can be either created, only read or modified by DONJON modules and utlilties ones. Their definitions have been reported and can be obtained on request. A data structure is recognized by a module with its signature. Here we will give a short descritption of these structures.

GEOMETRY data structure containing geometry informations. Signature `L_GEOM`.

INDEX data structure containing mixture index references. Signature `L_INDEX`.

MACROLIB data structure containing multi-group macroscopic properties. Signature `L_MACROLIB`.

TABLE data structure containing multi-group macroscopic properties generally for fuel. These properties are stored as burnup dependent properties. Signature `L_TABLE`.

COMPO data structure containing multi-group nuclear properties as generated by the DRAGON transport code. Signature `L_COMPO`.

DEVICE data structure containing device positions and reference to its properties. Signature `L_DEVICE`.

PROCEDE data structure containing device informations about moving capabilities such as speed. Signature `L_PROCEDE`.

MAP data structure containing fuel bundle locations and states. Signature `L_MAP`.

DETECT data structure containing detector positions and responses. Signature `L_DETECT`.

VANADIUM data structure containing vanadium detector responses. Signature `L_VANAD`.

RRS data structure containing reactor regulating system informations Signature `L_SRR`.

SDS data structure containing shutdown system informations. Signature `L_SAU`.

TRACK data structure containing discretization informations. Signature `L_BIVAC` or `L_TRIVAC`.

SYSTEM data structure containing system matrices. Signature `L_SYSTEM`.

FLUX data structure containing the solution of a direct or adjoint diffusion equation or flux harmonics or generalized direct or adjoint solutions of fixed source eigenvalue problems. Signature `L_FLUX`.

IQS data structure containing a quasi-static solution based on the improved quasi-static method. Signature `L_IQS`.

SOURCE data structure containing source terms of a fixed source eigenvalue problem. Signature `L_GPT`.

2. OBJECT CREATION

In this section, input data for initialization will be given. These modules are important to define the topology consistent with a problem definition.

2.1 The GEOD: module

The GEOD: module is used to create or modify a geometry. The geometry definition module in DONJON 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 linked list or an XSM file under its given name. It is always possible to modify a given existing geometry or copy it into a neighboring linked list or XSM file under a new name. The calling specifications are:

Structure **(GEOD:)**

{ *GEOM* `:=` `GEOD:` `::` **(geod_data1)** |
GEOM `:=` `GEOD:` { *GEOM* | *OLDGEOM* } `::` **(geod_data2)** }

where

GEOM `character*12` name of the GEOMETRY linked list or the XSM file.

OLDGEOM `character*12` name of a GEOMETRY linked list or an XSM file containing the existing geometry. The type and all the characteristics of *OLDGEOM* will be copied onto *GEOM*.

(geod_data1) structure describing the characteristics of a new geometry (see section 2.1.1).

(geod_data2) structure describing the change to the characteristics of an existing geometry (see section 2.1.1).

2.1.1 Data input for module GEOD:

Structures **(geod_data1)** and **(geod_data2)** serve to define the principle components of a geometry (dimensions, materials, boundary conditions):

Structure **(geod_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)
;

Structure **(geod_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` key word used to set *iprint*.

iprint index used to control the printing in module `GEOD:`. =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:

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* ≥ 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 `MACD:` module.
`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.[6]
`ACYL`	the side under consideration has an arbitrary albedo to be specified with a circular correction applied on the Cartesian boundary. This option is only available in the $X-Y$ plane for `CAR2D` and `CAR3D` geometries.[6]
`S30`	hexagonal symmetry of one twelfth of an assembly (see Fig. 2).
`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).

`RADS` This key word is used to specify the cylindrical correction applied in the $X-Y$ plane for `CAR2D` and `CAR3D` geometries.

`ANG` This key word 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 key word `ANG` is present. *ang*(ir) $= \frac{\pi}{2}$ by default (i.e. the correction is applied at every angle).

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:

Structure **(descMC)**

```
MIX
  { (imix(i),i=1,lreg) |
  [[ PLAN 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` key word 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 array containing type of material mixture associated with a region. It is important that *imix*$\leq$*nmixt* where *nmixt* is defined in the `MACD:` or `CRE:` modules. If *imix*=0, the corresponding 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.

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

iplan plane number for which material mixture are input.

`SAME` key word 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` key word 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` key word to specify that the *lc* material mixture number of the current crown have the same value *jmix*.

`UPTO` key word 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:

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` key word for the mesh of the geometry along the X axis.

`MESHY` key word for the mesh of the geometry along the Y axis.

`MESHZ` key word for the mesh of the geometry along the Z axis.

`RADIUS` key word for the mesh of the geometry in the radial direction.

`SIDE` key word 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` key word for mesh splitting of the geometry along the X axis.

`SPLITY` key word for mesh splitting of the geometry along the Y axis.

`SPLITZ` key word for mesh splitting of the geometry along the Z axis.

`SPLITR` key word for mesh splitting of the geometry in the radial direction.

ispltx array containing the 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*(i)=1.

isplty array containing the 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*(i)=1.

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

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

where *maxpts* is set in `TRIVAT:` or `BIVADT:` modules (see Section 3.).

2.1.2 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 DONJON.

1. Slab geometry (see Fig. (plaque)):

```
GEOMETRY1 := GEOD: :: 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
 ;
```

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

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

2.2 The `USPLIT:` module

In DONJON when using device movement capabilities, cross reference index between the problem geometry, the material properties and the device definition have to be made. The `USPLIT:` module is used to create the index. It results that the initial geometry is analized and split if it is reuired. An other GEOMETRY linked list is produced with the following information:

- The mesh-splitting operations are performed.
- Material mixture indices are computed on the resulting geometry to provide a single number per volume.

The INDEX linked list is produced containing only material mixture indices as defined in the initial geometry. The calling specifications are:

Structure **(USPLIT:)**

NEWGEO INDEX `:=` `USPLIT:` *OLDGEO* `::` **(usplit_data)**

where

NEWGEO `character*12` name of the GEOMETRY linked list or the XSM file that will contain the new geometry.

INDEX `character*12` name of a INDEX linked list or an XSM file that will contain the material mixture indices.

OLDGEO `character*12` name of a GEOMETRY linked list or an XSM file containing the existing geometry, resulting from the `GEOD:` module.

(usplit_data) structure describing the data to the `USPLIT:` module.

2.2.1 Data input for module `USPLIT:`

Structure **(usplit_data)**

`MAXR` *maxpts*
;

where

`MAXR` key word to specify the maximum number of regions of the geometry.

maxpts maximum dimensions of the problem to be considered.

2.3 The `MACD:` module

In DONJON the macroscopic cross sections and diffusion coefficients can be read from the input data file using REDGET. The general format of the data for the `MACD:` module in DONJON is the following:

Structure **(MACD:)**

MACRO `:=` `MACD:` [*MACRO*] `::` **(macd_data)**

where

MACRO `character*12` name of the MACROLIB linked list or the XSM file that will contain the macroscopic cross sections and diffusion coefficients. If *MACRO* appears on both LHS and RHS, it is updated; otherwise, it is created. If *MACRO* is created, all macroscopic cross sections and diffusion coefficients are first initialized to zero.

(macd_data) structure containing the data to module `MACD:`.

2.3.1 Data input for module `MACD:`

Structure **(macd_data)**

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

where

`EDIT` key word used to set *iprint*.

iprint index used to control the printing in module `MACD:`. =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` key word used to define the number of energy groups. This data is given if and only if *MACRO* is created.

ngroup the number of energy groups used for the calculations in DONJON.

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

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

`DELN` key word 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` key word used to specify the maximum level of anisotropy permitted in the diffusion cross sections. This data is given only if *MACRO* is created.

naniso the maximum level of anisotropy. The default value is *naniso*=1, meaning isotropic data.

`ALBP` key word used for the input of the physical albedos.

nalbp the maximum number of physical albedos.

albedp physical albedos (real numbers).

`STEP` key word used to create a perturbation directory.

istep the index of the perturbation directory.

`READ` key word used to specify input of the cross section information from default input by REDGET.

(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.

The following options are used only to provide an easy way of converting old mixture specifications of TRIVAC-2.

`OLD` key word used to specify input of the cross section information from default input by REDGET in the TRIVAC-2 format. The nuclear data will be translated into DONJON format and printed on the listing. It will be the end of the execution.

(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` key word used to specify perturbed input of the cross section information from default input by REDGET in the TRIVAC-2 format. The perturbed nuclear data will be translated into DONJON format and printed on the listing. It will be the end of the execution.

(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.

2.3.2 Description of the nuclear data

Structure **(macxs)**

```
MIX matnum
  [ DIFFX (xdiffx(jg), jg=1,ngroup) ]
  [ DIFFY (xdiffy(jg), jg=1,ngroup) ]
  [ DIFFZ (xdiffz(jg), jg=1,ngroup) ]
  [ TOTAL (xssigt(jg), jg=1,ngroup) ]
  [ NUSIGF (xssigf(jg), jg=1,ngroup) ]
  [ CHI (chi(jg), jg=1,ngroup) ]
  [ CHDL ((chdl(i,jg), jg=1,ngroup), i=1,ndg) ]
  [ VELM (velm(jg), jg=1,ngroup) ]
  [ FIXE (xsfixe(jg), jg=1,ngroup) ]
  [ H-FACTORS (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` key word for input of the cross sections associated with a mixture.

matnum identifier for the material mixture to be read from the input data file. The maximum number of identifiers permitted is *nmixt* and the maximum value that *matnum* may have is *nmixt*.

`DIFFX` key word for input of the $X-$directed diffusion coefficient.

xdiffx array representing the multigroup $X-$directed diffusion coefficient (D_{x}^{g} in cm) for the mixture *matnum*.

`DIFFY` key word for input of the $Y-$directed diffusion coefficient. By default, *xdiffy* is taken equal to *xdiffx*.

xdiffy array representing the multigroup $Y-$directed diffusion coefficient (D_{y}^{g} in cm) for the mixture *matnum*.

`DIFFZ` key word for input of the $Z-$directed diffusion coefficient. By default, *xdiffz* is taken equal to *xdiffx*.

xdiffz array representing the multigroup $Z-$directed diffusion coefficient (D_{z}^{g} in cm) for the mixture *matnum*.

`TOTAL` key word for input of the macroscopic total cross sections.

xssigt array representing the multigroup macroscopic total cross sections (Σ^{g} in cm^{-1}) for the mixture *matnum*. For diffusion theory cases, *xssigt* can be set equal to the removal cross sections, provided that the within-group scattering cross sections are set to zero.

`NUSIGF` key word for input of the fission cross sections multiplied by the average number of neutrons per fission.

xssigf array representing the multigroup macroscopic fission cross sections multiplied by the average number of neutrons per fission ($\nu\Sigma_{\mathrm{f}}^{g}$ in cm^{-1}) for the material mixture *matnum*.

`CHI` key word for input of the fission spectrum. By default, a fission spectrum is set equal to 1.0 in the highest energy group and is set equal to zero in every other group.

chi array representing the multigroup fission spectrum (χ^g with $\sum_g \chi^g = 1$) for the material mixture *matnum*.

`CHDL` key word used to set *chdl*(i,jg).

chdl array representing the delayed neutron fission spectrum for the i^{th} precursor group ($\chi_{i,jg}$ with $\sum_{jg} \chi_{i,jg} = 1$) and for the material mixture *matnum*. By default, the delayed neutron fission spectrum is set equal to 1.0 in the highest energy group and is set equal to zero in every other group.

`VELM` key word for input of the average neutron velocities.

velm array representing the average neutron velocities for the material mixture *matnum* (v_g in cm s^{-1}).

`FIXE` key word for the input of the fixed neutron sources (S^g in $s^{-1}cm^{-3}$).

xsfixe array representing the multigroup fixed neutron sources for the material mixture *matnum*.

`H-FACTORS` key word for input of the power coefficients. The power coefficient (H-factor) is defined as the macroscopic fission cross sections multiplied by the average energy recovered per fission.

xhfact multigroup power coefficients for the mixture *matnum*.

`SCAT` key word for input of the macroscopic diffusion cross sections.

nbscat number of groups participating in the diffusion towards the group jg considered in mixture *matnum* for a given level of anisotropy.

ilastg index of the lowest thermal energy group participating in the diffusion towards the group jg considered in mixture *matnum* for a given level of anisotropy.

scat array representing the multigroup scattering cross section ($\Sigma_{sl}^{ig \to jg}$ in 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	$\Sigma_{s,l}^{1\to1}$	$\Sigma_{s,l}^{1\to2}$	$\Sigma_{s,l}^{2\to1}$	$\Sigma_{s,l}^{2\to2}$
0	0.50 cm^{-1}	0.20 cm^{-1}	0.03 cm^{-1}	0.40 cm^{-1}
1	0.05 cm^{-1}	0.00 cm^{-1}	0.00 cm^{-1}	0.04 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
```

The input data structure **(triv2)** is used when the user knows TRIVAC-2 cross section input and wants to know the correct input data in the structure **(macxs)**. It can only be used for one- or two-group cross sections. The input data structure **(trip2)** is given after and has almost the same contain.

Structure **(triv2)**

((*xdiffx*(jg) *xsigt*(jg) *xsigf*(jg) *xhfac*(jg) `T` *scat*(0,jg,ig) [`DY` *xdiffy*(jg)] [`DY` *xdiffy*(jg)] `*` , jg=1,*ngrp*) ,i= 1,*nmixt*)

Structure **(trip2)**

(*matnum* (*xdiffx*(jg) *xsigt*(jg) *xsigf*(jg) *xhfac*(jg) `T` *scat*(0,jg,ig) [`DY` *xdiffy*(jg)] [`DY` *xdiffy*(jg)] `*` , jg=1,*ngrp*) ,i= 1,*nmixt*)

2.3.3 Writing Powderpufs cross sections in a MACROLIB

Powderpufs is a four factors lattice code which computes a limited number of cross sections and diffusion coefficients for use in a "one and a half group" reactor calculation. The only quantities produced are

$$\begin{aligned} D(1),\ D(2) &= \text{fast and thermal diffusion coefficients} \\ \Sigma_{\rm r}(1),\ \Sigma_{\rm r}(2) &= \text{fast and thermal removal cross sections} \\ \Sigma_{\rm s}(2 \leftarrow 1) &= \text{slowing-down cross section} \\ \nu\Sigma_{\rm f}(2) &= \nu \text{ times the thermal fission cross section} \\ H(2) &= \text{thermal H-factor} \end{aligned}$$

The fast and thermal removal cross sections are respectively defined in term of the absorption ($\Sigma_{\rm a}(g)$) and slowing-down cross sections using:

$$\Sigma_{\rm r}(1) = \Sigma_{\rm a}(1) + \Sigma_{\rm s}(2 \leftarrow 1)$$

and

$$\Sigma_{\rm r}(2) = \Sigma_{\rm a}(2)$$

The removal cross sections can be written in place of the total cross sections on the MACROLIB, provided that the within-group scattering cross sections are set equal to zero. A user should therefore write its data as:

`'TOTAL'` $\Sigma_{\rm r}(1)$ $\Sigma_{\rm r}(2)$
`'DIFFX'` $D(1)$ $D(2)$
`'SCAT' 1 1 0.0 2 2 0.0` $\Sigma_{\rm s}(2 \leftarrow 1)$
`'NUSIGF' 0.0` $\nu\Sigma_{\rm f}(2)$
`'H-FACTORS' 0.0` $H(2)$

2.4 The XSCONS: module

The XSCONS: module is used to create a TABLE object containing burnup-dependent fuel nuclear properties from COMPO linked lists or XSM files. The calling specifications are:

Structure **(XSCONS:)**

TABFL `:=` `XSCONS:` [[*CPO*]] `::` **(xsc_data)** `;`

where

TABFL `character*12` name of the TABLE linked list or XSM file that wil contain the fuel properties.

CPO `character*12` name of the COMPO linked list or XSM file.

(xsc_data) structure containing the data to module `XSCONS:`.

2.4.1 Data input for module `XSCONS:`

Structure **(xsc_data)**

```
[ EDIT iprint ]
READ [[ COMPO CPO (int_data) ]]
;
```

where

`EDIT` key word used to set *iprint*.

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

`READ` key word used to read the TABLE specifications (burnup, neutron exposure, number densities) from the input data file.

`COMPO` key word used to select a COMPO and to set the interpolation information.

CPO `character*12` name of the selected COMPO linked list or XSM file.

(int_data) structure containing the interpolation and access data. This structure is defined as

Structure **(int_data)**

```
[[ TYPE NAMTAB HTYPE
[ MICRO { [[ HISO { conc | * } ]] | ALL } ]
[ UPS ] { BURNUP | N/KB } [ INTRPL dburn ] ENDTYP ]]
```

where

`TYPE` key word used to set *NAMTAB*.

NAMTAB name of the directory in the TABLE where fuel properties will be stored. In general this name depend on the COMPO file specified.

HTYPE name of the material mixture. Each name refers to a type of nuclear data that is stored on a directory in the COMPO linked list or XSM file. *HTYPE* is a `character*12` name built from the concatenation *HCOMPO//HIORD* where *HCOMPO* is an ASCII name with a maximum of 8 characters and *HIORD* is a four digit suffix with value `'   1'`, `'   2'`, `'   3'`, etc., indicating the material mixture index.

`MICRO` key word used to set the number densities of the extracted isotopes present in the COMPO linked list or XSM file. By default, the extracted isotopes are not added to the resulting TABLE.

HISO `character*12` name of an extracted isotope.

conc user-defined value of the number density (in 10^{24} particles per cm^3) of the extracted isotope.

`*` the value of the number density for isotope *HISO* is recovered from the COMPO.

`ALL` all the values for the number densities are recovered from the COMPO.

`UPS` key word used to compute properties with no up-scattering contribution.

`BURNUP` key word used to recover burnup information and to set it as an interpolation parameter. This is the default option.

`N/KB` key word used to recover neutron exposure information and to set it as an interpolation parameter. This option can be useful to compare results with other diffusion codes where neutron exposure is the only dependent parameter for fuel properties.

`INTRPL` keyword used to order a linear interpolation of fuel properties with respect to burnup.

dburn burnup increment used in the property interpolation. This value must be in MWd/t.

`ENDTYP` end of specification key word for the fuel type.

2.5 The `CRE:` module

The CRE: module is used to create or modify an extended MACROLIB containing set of interpolated nuclear properties from a COMPO linked list or XSM file. We have two sorts of utilization:

- Construction of an extended MACROLIB for few material regions directly from COMPO information.
- Construction of an extended MACROLIB for fuel properties directly from TABLE information with respect to an evolution parameter (burnup or neutron exposure) contained in the fuel map object.

The TABLE information are obtained by the `XSCONS:` module. The COMPO information are recovered from transport calculations. The calling specifications are:

Structure **(CRE:)**

{ *MACRO* `:=` `CRE:` [*MACRO*] [[*CPO*]] `::` **(cre_data1)** |
MACFL `:=` `CRE:` [*MACFL*] *TABFL MAPFL* `::` **(cre_data2)** }

where

MACRO `character*12` name of the extended MACROLIB. If *MACRO* appears on the RHS, the information previously stored in *MACRO* is kept.

CPO `character*12` name of the COMPO linked list or XSM file.

MACFL `character*12` name of the extended MACROLIB. If *MACRO* appears on the RHS, the information previously stored in *MACRO* is kept.

TABFL `character*12` name of the TABLE linked list or XSM file containing fuel properties with respect to burnup.

MAPFL `character*12` name of the MAP linked list or XSM file containing fuel regions description and burnup informations.

(cre_data1) structure containing the data to module `CRE:`.

(cre_data2) structure containing the data to module `CRE:`.

2.5.1 Data input for module CRE:

Structure **(cre_data1)**

```
[ EDIT iprint ]
[ NMIX nmixt ]
READ [[ COMPO CPO (intf_data) ]]
;
```

Structure **(cre_data2)**

```
[ EDIT iprint ]
READ [[ TABLE TABFL (intf_data) ]]
;
```

where

`EDIT` key word used to set *iprint*.

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

`NMIX` key word used to define the number of material mixtures. This data is given if and only if *MACRO* is created and *TABFL* is not specified.

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

`READ` key word used to read the MACROLIB specifications (burnup, neutron exposure, number densities) from the input data file.

`COMPO` key word used to select a COMPO and to set the interpolation information.

CPO `character*12` name of the selected COMPO.

`TABLE` key word used to select a TABLE and to set the interpolation information.

TABFL `character*12` name of the selected fuel TABLE.

(intf_data) structure containing the interpolation data. This structure is defined as

Structure **(intf_data)**

```
[[ MIX matnum HTYPE [ DERIV ] [ UPS ]
[ { { BURNUP burn | N/KB xnkb } [ { HOMG | TAVG | TAVGC | DIRECTC | DIRECT } ]
| T-BURNUP burn0 burn1 | T-N/KB xnkb0 xnkb1 } ]
[ MICRO { [[ HISO { conc | * } ]] | ALL } ]
ENDMIX ]]
```

where

`MIX` key word used to set *matnum*.

matnum identifier for the material mixture to be created. The maximum number of identifiers permitted is *nmixt* and the maximum value that *matnum* may have is *nmixt*.

HTYPE name of the material mixture. Each name refers to a type of nuclear data that is stored on a directory in the COMPO or TABLE linked list or XSM file. *HTYPE* is a `character*12` name built from the concatenation *HCOMPO//HIORD* where *HCOMPO* is an ASCII name with a maximum of 8 characters and *HIORD* is a four digit suffix with value `'   1'`, `'   2'`, `'   3'`, etc., indicating the material mixture index.

`DERIV` key word used to compute the derivative of the MACROLIB information with respect to *burn*, *xnkb*, *burn1* or *xnkb1*. By default, the MACROLIB information is not differentiated.

`UPS` key word used to compute properties with no up-scattering contribution. For TABLE interpolation, this option must have been set in the prevceeding call to `XSCONS:` module.

`BURNUP` key word used to perform a single interpolation and to set the burnup value *burn* in case of direct access to COMPO files. By default, the MACROLIB information is computed for $burn = xnkb = 0.0$. In case of TABLE interpolation, this key word is used to set burnup as the interpolation parameter.

burn value of the burnup in MW day per tonne of initial heavy elements. For TABLE interpolation, this value must be read but is not used.

`N/KB` key word used to perform a single interpolation and to set the neutron exposure value *xnkb* for direct access to COMPO files. In case of TABLE interpolation, this key word is used to set neutron exposure as the interpolation parameter.

xnkb value of the neutron exposure in neutron/kb. For TABLE interpolation, this value must be read but is not used.

`HOMG` key word used to specify an homogeneous interpolation of fuel properties according to burnup values stored in *MAPFL*. This option can not be used when COMPO information are recovered.

`TAVG` key word used to specify a time-average interpolation per combustion zone of fuel properties according to burnup values stored in *MAPFL*. This option can not be used when COMPO information are recovered.

`TAVGC` key word used to specify a time-average interpolation of fuel properties according to burnup values stored in *MAPFL*. This option can not be used when COMPO information are recovered.

`DIRECT` key word used to specify a single interpolation per combustion zone of fuel properties according to burnup values stored in *MAPFL*. This option can not be used when COMPO information are recovered.

`DIRECTC` key word used to specify a single interpolation of fuel properties according to burnup values stored in *MAPFL*. This option can not be used when COMPO information are recovered.

`T-BURNUP` key word used to perform a time averaged MACROLIB evaluation and to set the burnup values *burn0* and *burn1*. This option is not valid for TABLE interpolation.

burn0 initial value of the burnup in MW day per tonne of initial heavy elements.

burn1 final value of the burnup in MW day per tonne of initial heavy elements.

`T-N/KB` key word used to perform a time averaged MACROLIB evaluation and to set the neutron exposure values *xnkb0* and *xnkb1*. This option is not valid for TABLE interpolation.

xnkb0 initial value of the neutron exposure in neutron/kb.

xnkb1 final value of the neutron exposure in neutron/kb.

`MICRO` key word used to set the number densities of the extracted isotopes present in the COMPO linked list or XSM file. By default, the extracted isotopes are not added to the resulting MACROLIB. This option is invalid for TABLE interpolation. Extracted isotope contribution must be recovered in `XSCONS:` module, when TABLE objects are created.

HISO `character*12` name of an extracted isotope.

conc user-defined value of the number density (in 10^{24} particles per cm^3) of the extracted isotope.

`*` the value of the number density for isotope *HISO* is recovered from the COMPO.

`ALL` all the values for the number densities are recovered from the COMPO.

`ENDMIX` end of specification key word for the material mixture.

2.6 The `AFM:` module

The AFM: module is used to create an extended MACROLIB containing set of interpolated nuclear properties or a TABLE from a feedback model database.[7, 8] The DATABASE information are obtained by previous DRAGON calculations. [9] There are three possible utilizations:

- Construction of an extended MACROLIB for fuel properties directly from DATABASE information with respect to local parameters contained in the fuel map object or directly input.
- Construction of a TABLE for fuel properties with respect to burnup. Every other local parameter can be input.
- Construction of an extended MACROLIB containing only one set of cross sections derivated from the DATABASE information. Properties can be obtained for fuel or reflector.

The calling specifications are:

Structure **(AFM:)**

{ *MACRO* | *TABFL* } `:=` `AFM:` *DBASE* [*MAPFL*] `::` **(afm_data)** }

where

MACRO `character*12` name of the extended MACROLIB.

TABFL `character*12` name of the TABLE linked list or XSM file containing fuel properties with respect to burnup.

DBASE `character*12` name of the DATABASE linked list or XSM file containing fuel properties with respect to local parameters.

MAPFL `character*12` name of the MAP linked list or XSM file containing fuel regions description and burnup informations. This file is only required when a MACRO is created for fuel area.

(afm_data) structure containing the data to module `AFM:`.

2.6.1 Data input for module AFM:

Structure **(afm_data)**

{ TAB | MAP | MCR *mmix* } INFOR *NAMDB*
DNAME *ntyp* (*NAMTYP*(i), i=1,*ntyp*)
REFT (*imix*(i) *NAMTYP*(i), i=1,*ntyp*)
[NTAB (*NAMTAB*(i), i=1,*ntyp*]
[EDIT *iprint*]
[FIXP *pow*]
[TFUEL *tfuel*]
[TCOOL *tcool*]
[TMOD *tmod*]
[BORON *nB*]
[RDCL *dcool*]
[RDMD *dmod*]
[PUR *purity*]
[BURN *bval*]
[{ XENON *nXe* | XEREF }]
[{ NEP *nNp* | NREF }]
[SAM *nSm*]
[IMET *imet*]
;

where

TAB	keyword to specify that a TABLE for fuel properties will be computed.
MAP	keyword to specify that a MACROLIB for fuel properties will be computed.
MCR	keyword to specify that a MACROLIB containing only one non-zero mixture will be created.
mmix	maximum number of mixtures in the MACROLIB.
INFOR	keyword to specify the data base name.
NAMDB	character*72 title of the database as it has been created.
DNAME	keyword to specify the number of fuel types and their names as stored in the data base.
ntyp	number of fuel types. For MCR option, *ntyp* must be 1.
NAMTYP(i)	character*12 name of the directory where each fuel type information has been stored.
REFT	keyword to specify a number associated with a fuel type name.
imix(i)	fuel type index as specified for the fuel map or a non-zero mixture number for the single-property sc macrolib.
NTAB	keyword to specify the directory name where the fuel properties will be stored in a TABLE. It is required only if TAB is used.
NAMTAB	character*12 name of the directory in TABLE.
EDIT	key word used to set *iprint*.
iprint	index used to control the printing in module AFM:. =0 for no print(default value); =1 for minimum printing; larger values produce increasing amounts of output.

`FIXP` key word used to set *pow*.

pow uniform bundle power in kW. If this data is omitted, the reference value in the data base is used or the bundle powers present in a MAP. The reference value is 615 kW.

`TFUEL` key word used to set *tfuel*.

tfuel fuel temperature in K. If this data is omitted and the bundle powers present in a MAP, fuel temperatures are computed with respect to powers. If this data is omitted and there is no bundle power, the reference value in the data base is used, where it is 941.29 K.

`TCOOL` key word used to set *tcool*.

tcool coolant temperature in K. If this data is omitted, the reference value in the data base is used. The reference value is 560.66 K.

`TMOD` key word used to set *tmod*.

tmod moderator temperature in K. If this data is omitted, the reference value in the data base is used. The reference value is 345.66 K.

`BORON` key word used to set *nB*.

nB Boron concentration in ppm. If this data is omitted, the reference value in the data base is used. The reference value is 0.0 ppm.

`RDCL` key word used to set *dcool*.

dcool coolant density in g/cm^3. If this data is omitted, the reference value in the data base is used. The reference value is 0.81212 g/cm^3.

`RDMD` key word used to set *dmod*.

dmod moderator density in g/cm^3. If this data is omitted, the reference value in the data base is used. The reference value is 1.082885 g/cm^3.

`PUR` key word used to set *purity*.

purity moderator purity in atm%. If this data is omitted, the reference value in the data base is used. The reference value is 99.911 atm%.

`BURN` key word used to set *bval*. This option is valid only when `MCR` is used and can not be omitted.

bval fuel burnup in MWd/t. This value must be positive.

`XENON` key word used to set *nXe*.

nXe Xenon concentration in $10^{24} at/cm^3$. This concentration will be aplied to every bundle.

`XEREF` key word used to specify that the Xenon concentrations as computed with DRAGON will be taken. If this option is omitted and MAP contains bundle fluxes, new Xenon concentrations will be computed and used.

`NEP` key word used to set *nNp*.

nNp Neptunium concentration in $10^{24} at/cm^3$.

`XEREF` key word used to specify that the Neptunium concentrations as computed with DRAGON will be taken. If this option is omitted and MAP contains bundle fluxes, new Neptunium concentrations will be computed and used.

`SAM` key word used to set *nSm*.

nSm Samarium concentration in $10^{24} at/cm^3$. If this data is omitted, bundle concentrations as computed by DRAGON is used.

`IMET` key word used to set *imet*.

imet interpolation type for time-average calculations. *imet* = 1: using lagrange approximations; *imet* = 2: using spline approximations; *imet* = 3: using hermit approximations (default value).

2.7 The `INIMAC:` module

The `INIMAC:` module is used to perform a new MACROLIB linked list, in which properties are stored for each material region and each device description. To obtain this new MACROLIB, properties are taken from other MACROLIB objects. There are two types of recovery:

- All material properties of the problem, cell, reflector and device properties, are stored in the very same MACROLIB.
- Reflector and device properties, i.e. evolution independent properties are stored in a MACROLIB, and fuel evolution dependent properties are in an other MACROLIB.

The calling specifications are:

Structure **(INIMAC:)**

MACRO2 `:=` `INIMAC:` *INDEX MACRO* [*MACFL*]
;

where

MACRO2 `character*12` name of the MACROLIB linked list containing the new set of properties.

INDEX `character*12` name of the INDEX linked list or XSM file containing the material mixture indices to reference to either one (all indices are positive) or two MACROLIB (fuel indices are negative).

MACRO `character*12` name of a MACROLIB created by `MACD:` or `CRE:` modules.

MACFL `character*12` name of a MACROLIB created by `CRE:` module (see structure **(cre_data2)**). This linked list can only contain interpolated fuel properties.

2.8 The `INIDEV:` module

The `INIDEV:` module is used to read and store device specific informations. The INDEX object is modified to provide reference access to device properties in an extended MACROLIB created by the `INIMAC:` module.

In specific case where it appends a storage problem of this extended MACROLIB, i.e. Slowpoke-2 hexagonal core simulation, device property access is realized directly with the initial MACROLIB created by `MACD:` ot `CRE:` modules. In this particular case, device and geometry definitions are strongly linked in terms that mixture numbers of regions affected by a device must be all different and must differ from region numbers where the device is absent.

The calling specifications are the following:

Structure **(INIDEV:)**

DEV INDEX `:=` `INIDEV:` [*DEV*] *INDEX NEWGEO* `::` **(inidev_data)**

where

DEV `character*12` name of the DEVICE that will contain the device informations. If *DEV* appears on both LHS and RHS, it will be updated.

INDEX `character*12` name of the INDEX created by the `USPLIT:` module, that contains the material mixture indices. Device material mixtures are added to the vector containing the geometry indices, and device indices are modified in consequence.

NEWGEO `character*12` name of the GEOMETRY that contains the geometry specification, resulting from `USPLIT:` module.

(inidev_data) structure containing the data input for module `INIDEV:`.

2.8.1 Data input for module `INIDEV`

A device is a reactivity controller such as a stainless steel rod or a liquid zone control unit. Each device may be composed of different parts, called descriptions, for which the properties can differ, such as a full and an empty liquid zone controller. A device is composed of at least one description.

Structure **(inidev_data)**

```
[ HEXZ ]
NDEV ndev
[ DELTA | NODEL ]
NDESC ndesc [ MIXM ]
( (dev_info) , i=1,ndev )
;
```

where

`HEXZ` key word to specify that only hexagonal devices will be defined. If this key word is absent, only Cartesian and tube devices can be defined. Be aware that this option will initiate the "patch" for storage problem explained above.

`NDEV` key word to specify the number of devices.

ndev number of devices. This value must be greater than 0.

`DELTA` key word to specify that device properties will be obtained by the difference between the perturbed and the reference cross sections.

`NODEL` key word to specify that device properties will be obtained directly from incremental cross sections, only one mixture is used. This is the default option.

`NDESC` key word to specify the total number of device descriptions.

ndesc total number of device descriptions. This value must be greater than 0.

`MIXM` key word to specify that device mixtures will only be modified. This option is valid only if *DEV* is in modification mode.

(dev_info) structure describing the format used to read the devices from the input data file.

2.8.2 Description of the device data

Note that the input order must be respected. Any device is referred by its 3-D Cartesian coordinates. But tube or hexagonal device in 3-D can be defined as well as 2-D devices by its X-Y coordinates.

Structure **(dev_info)**

```
DEVICE NAMTYP NAMFAM NAMSEQ
[ NHEX nhex HEX ( ihex(i), i=1,nhex ) ]
[ ( pos_max(i), i=1,6 ) ] [ MOVDIR speed ( param(j), j=1,2 ) ]
[[ NAMDESC [ ( pos_cur(i), i=1,6 ) ] ( mixnb(n), n=1,2 ) ]]
ENDDEV
```

where

`DEVICE` key word for input of the information associated with a device.

NAMTYP `character*12` name of the device type. This means that the devices are reactivity mechanisms or other kind of perturbations of the core. (Ex: CONTROLLED, PERT, SHUTOFF)

NAMFAM `character*12` name of the device family. (Ex: ROD, LZC)

NAMSEQ `character*12` name of the device sequence. A sequence is a physical device present in the core. (Ex: SSAR#1, ZCU#1)

`NHEX` key word to set the number of hexagons affected by the device.

nhex number of hexagons.

`HEX` key word to set the hexagon numbers affected by the device.

ihex array containing the hexagon numbers where the device is present, as ordered in the geometry definition (See Section 2.1).

pos_max array containing the maximum allowed coordinates of the device. The positions must be read in the order: X− X+ Y− Y+ Z− Z+ . If a tube 3-D device is defined, the first two coordinates are the inner and the outer radii of the tube. For any device type other than Cartesian 3-D, the Y coordinates are not used but must be given. For 2-D device, Z coordinates must 0.0 and a value greater than 1.0. This vector is optional only with `MIXM` keyword.

MOVDIR `character*1` name of the movement direction of the device, must be 'X', 'Y' or 'Z'. A device can only be moved along one of the axis defined in the geometry. This value is optional only with `MIXM` keyword.

speed speed of the device in cm/s. LZC speed is positive when LZC are filled, and rod speed is positive when they are inserted in the core. This value is optional only with `MIXM` keyword.

param array containing other types of information such as time of displacement or isotope concentration. This information is not currently used. This vector is optional only with `MIXM` keyword.

NAMDESC `character*12` name of a device description. (Ex: SSAR#1A, EMPTY#1)

pos_cur array containing the current coordinates of the device description. The positions must be read in the order: X− X+ Y− Y+ Z− Z+ . Those coordinates must be the maximum allowed coordinates of the device part. If a tube 3-D device is defined, the first two

coordinates are the inner and the outer radii of the tube. For any device type other than Cartesian 3-D, the Y coordinates are not used but must be given. This vector is optional only with `MIXM` keyword.

mixnb array containing the material mixture indices of the device description properties as been defined in `MACD:` or `CRE:` modules. If `DELTA` option is active, the first and the second mixture indices must represent the perturbed and the reference cross sections respectively. If `NODEL` option is active, the first mixture index must represent the incremental cross sections of the device description and the second must be 0.

`ENDDEV` key word for ending the input of a device.

When a device is moved or placed at a specific position, its maximum length along its movement axis W_{max} is defined by the difference between its maximal coordinate and the orign of this axis. Its current length W_{cur} is defined in the same way.

For example, if a device moved along the 'Y' axis, i.e. $MOVDIR$ = 'Y', $W_{max} = pos\_max(4) - MESHY(1)$. $pos\_max(4)$ is the maximum coordinate along 'Y' axis, and $MESHY(1)$ is the first value of the array *MESHY* defined in `GEOD:` module (see section 2.1.1).

2.9 The `INPROC:` module

The `INPROC:` module is used to read and store information related to the devices like maximum filling time for LZC and to link device names and types with their numbering in the core. The calling specifications are:

Structure **(INPROC:)**

PROCESS `:= INPROC:` *DEVICE* `::` **(inpro_data)**

where

PROCESS `character*12` name of the PROCEDE containing the process information required to control device positions and speeds.

DEVICE `character*12` name of the DEVICE containing the device information.

(inpro_data) structure containing the data input for the module `INPROC:`.

2.9.1 Data input for module `INPROC:`

Structure **(inpro_data)**

```
EDIT iprt
[[ TYPE NAMTYP
END
[[ { LZC (lzc_data) | ROD (rod_data) } ]] ]]
;
```

where

`EDIT` key word used to set *iprt*.

iprt index used to control the printing in module `INPROC:`. =0,1 for no print(default value); =2 for minimum printing; =3 for printing controller sequence name; =4 for printing description names.

`TYPE` key word used to set *NAMTYP*.

NAMTYP Character*12 name of device type that contains device specifications in DEVICE.

`LZC` key word used to set **(lzc_data)**.

(lzc_data) structure containing the data input for LZC device.

`ROD` key word used to set **(rod_data)**.

(rod_data) structure containing the data input for rod device.

The data input of the structure **(lzc_data)** is:

Structure **(lzc_data)**

NAMFAL `NAME` *NAMSEQ* `NAMFUL` *NAMDESC* `NAMEMP` *NAMDESC*
[`NB` *nb*]
[`BIAS` *bias*]
[`YFULL` *yfull*]
[`TIME` (*trep*(i) ,i=1,2)]
[`ZFLOW` (*zflow*(i) ,i=1,2)]
`END`

where

NAMFAL Character*12 family name of liquid zone controllers as set in DEVICE.

`NAME` key word used to set *NAMSEQ*.

NAMSEQ Character*12 sequence name of a liquid zone controller as set in DEVICE.

`NAMFUL` key word used to set *NAMDESC*.

NAMDESC Character*12 description name of the full part of a liquid zone controller as set in DEVICE.

`NAMEMP` key word used to set *NAMDESC*.

NAMDESC Character*12 description name of the empty part of a liquid zone controller as set in DEVICE.

`NB` key word used to set *nb*.

nb number of a liquid zone controller, it must be between 1 and total number of zones, in general 14. This value is only used by `SRR:` module.

`BIAS` key word used to set *bias*.

bias the valve bias of a liquid zone controller in full filling fraction, i.e. the lift at which the in-flow equals the out-flow.

`YFULL` key word used to set *yfull*.

yfull the coordinate of full filling of the liquid zone controller.

`TIME` key word used to set *trep*.

trep array containing filling time of a liquid zone controller for the first value and 0.0 for the second.

`ZFLOW` key word used to set *zflow*.

zflow array containing the total demanded valve lift at current time of transient as the first value and the previous total demanded valve lift as the second one . These values are set by `SRR:` module.

The data input of the structure **(rod_data)** is:

Structure **(rod_data)**

NAMFAR `NAME` *NAMSEQ*
[`NB` *nb*]
[`NBANC` *nbanc*]
[`TIME` *tins*]
`END`

where

NAMFAR Character*12 family name of rods as set in DEVICE.

`NAME` key word used to set *NAMSEQ*.

NAMSEQ Character*12 sequence name of a rod as set in DEVICE.

`NB` key word used to set *nb*.

nb number of a rod, it must be between 1 and total number of rods of this family. This value is only used by `SRR:` module.

`NBANC` key word used to set *nbanc*.

nbanc number of the bank containing an adjuster rod. This value is only used by `SRR:` module.

`TIME` key word used to set *tins*.

tins maximum time in seconds for complete insertion or withdrawal of a rod.

`END` key word used to specify the end of a data block.

2.10 The `INIRES:` module

The `INIRES:` module is used to read fuel map informations defining the fuel lattice. The calling specifications are:

Structure **(INIRES:)**

MAPFL `:=` `INIRES:` [*MAPFL*] `::` **(ires_data)**

where

MAPFL `character*12` name of the MAP that will contain the fuel map informations. If *MAPFL* appears on both LHS and RHS, it will be updated.

(ires_data) structure containing the data to module `INIRES:`.

2.10.1 Data input for module INIRES:

The fuel map is composed of a 3-D geometry and of some other informations. Only Cartesian and hexagonal 3-D geometries are allowed. The number of non-virtual regions in the embedded geometry must be equal to the number of channels times by the number of bundles per channel.

Structure **(ires_data)**

[NBUND *nk*] [NCHAN *nch*] [NZONE *nzc*] [NGRP *ngrp*] [IMOD *imod*]
[::: GEOD: **(geo_data1)**]
[NXNAME (*NAMEX*(i),i=1,*nx*)] [NYNAME (*NAMEY*(i),i=1,*ny*)]
[ZONE (*izone*(i),i=1,*nch*)]
[BURN-ZC (*burnz*(i),i=1,*nzc*)]
[CPPF-ZONE (*izcppf*(i),i=1,*nch*)]
[CONTR-ZONE (*izpzc*(i),i=1,*nch*)]
[BURN-FUEL (*burnf*(i),i=1,*nch***nk*)]
[BUNDLE-PW (*power*(i),i=1,*nch***nk*)]
;

where

NBUND — key word to specify the number of bundles in a channel.

nk — number of bundles. This value must be greater than 0 and less than equal to the number of Z planes in the embedded geometry. This value is given if and only if *MAPFL* is created.

NCHAN — key word to specify the number of channels.

nch — number of channels. This value must be greater than 0 and less than equal to the number of non-virtual regions per Z plane in the embedded geometry. This value is given if and only if *MAPFL* is created.

NZONE — key word to specify the number of combustion zones.

nzc — number of combustion zones. This value must be greater than 0. This value is given if and only if *MAPFL* is created.

NGRP — key word to specify the number of energy groups. The number of energy groups must be specified when *MAPFL* is created.

ngrp — number of energy groups. This value must be greater than 0.

IMOD — key word to specify the type of interpolation performed fuel properties with respect to burnup. This information is required when *MAPFL* is created and that no burnup or power are provided.

imod — type of interpolation. This value must be greater than 0 and less than 6. If *imod* = 1, homogeneous model used ; = 2, time-average model per combustion zone; = 3, for complete time-average model; = 4, for instantaneous model;= 5, for instantaneous model per combustion zone.

::: — key word used to indicate the call of an embedded module.

GEOD: — key word used to call module GEOD: used to describe fuel map as a COARSE geometry.

(geo_data1) — structure containing the data to the module GEOD: (see section 2.1).

NXNAME — key word to specify the horizontal channel names.

NAMEX `character*4` array for horizontal channel names. The number of names must equal to the number of subdivisions in the X direction in the fuel map embedded geometry.

nx number of subdivisions in the X direction in the fuel map embedded geometry.

`NYNAME` key word to specify the vertical channel names.

NAMEY `character*4` array for vertical channel names. The number of names must equal to the number of subdivisions in the Y direction in the fuel map embedded geometry.

ny number of subdivisions in the Y direction in the fuel map embedded geometry.

`ZONE` key word to specify the combustion zone locations.

izone array containing combustion zone numbers of each channel.

`BURN-ZC` key word to specify the average exit burnup per combustion zone.

burnz array containing average exit burnups.

`CPPF-ZONE` key word to specify the channels of interest for CPPF measurement.

izcppf array containing index numbers, = 1 for interesting channels = 0 otherwise.

`CONTR-ZONE` key word to specify the channels of interest for zonal power calculations.

izpzc array containing index numbers of each zone. If 7 zones are defined, 14 overall zones will be used, divided bundle numbers per channel by 2.

`BURN-FUEL` key word to specify the instantaneous burnups per bundle.

burnf array containing instantaneous burnups, given in MWd/T or n/kb.

`BUNDLE-PW` key word to set the bundle powers.

power array containing bundle powers, given in kW.

2.11 The `REFRES:` module

The `REFRES:` module is used to compute cross references between the calculation geometry, the index of material mixtures and the fuel map geometry. The calling specifications are:

Structure **(REFRES:)**

INDEX MAPFL `:=` `REFRES:` *INDEX MAPFL NEWGEO* ;

where

INDEX `character*12` of the INDEX containing the material mixture indices.

MAPFL `character*12` of the MAP containing the fuel map informations.

NEWGEO `character*12` of the GEOMETRY containing the mesh-splitting geometry created by `USPLIT:` module.

2.12 The INIDET: module

The INIDET: module is used to read and store detector information. A detector is represented by a 2-D or 3-D Cartesian/Hexagonal geometry. The calling specifications are:

Structure **(INIDET:)**

DETEC := INIDET: [*DETEC*] :: **(inidet_data)**

where

DETEC character*12 name of the DETECT containing the detector informations. If *DETEC* appears on the RHS, it will be updated.

(inidet_data) structure containing the data to module INIDET:.

2.12.1 Data input for module INIDET:

Structure **(inidet_data)**

[EDIT *iprt*] [HEXZ] NGRP *ngrp*
[[TYPE *NAMTYP*
INFO *ndetect nrep* { SPECTRAL (*spec*(i), i=1,*ngrp*) | DEFAULT }
[INVCONST (*tinv*(i), i=1,*nrep*−2)] [FRACTION (*fract*(i), i=1,*nrep*−1)]
(**(det_data)**, i=1,*ndetect*)]]
;

where

EDIT key word used to set *iprt*.

iprt index used to control the printing in module INIDET:. =1,2 for no print(default value); =3 for printing the contents of the output DETECT.

HEXZ key word to specify that only hexagonal detectors will be defined. If this key word is absent, Cartesian detectors will be defined.

NGRP key word used to set *ngrp*.

ngrp number of energy groups in the calculation. It must be equal to the number set in the MACD: module or by the COMPO files.

TYPE key word to specify the detector type.

NAMTYP character*12 name of the detector type. To correspond to the actual detector response model encoded, the type of detector must be in this list:

- PLATN_REGUL
- PLATN_SAU
- VANAD_REGUL
- CHION_SAU
- CHION_REGUL

For other type names, only a fixed normalisation can be performed.

INFO key word to specify the information associated with the detector type.

ndetect number of detectors of the specified type.

nrep number of detector response components for the specified type. It must be greater or equal to 2, corresponding to a response in fraction and the reference flux value.

`SPECTRAL` key word to specify the energy spectral of a detector type.

spec array containing the energy spectral of a detector type.

`DEFAULT` key word to specify the energy spectral will be initialized as 1.0 for the highest energy group and 0.0 for other groups.

`INVCONST` key word to specify the inverse time constants of the detector type model. This option is only valid for platinum, (*NAMTYP*(1:5) = 'PLATN'), detector type.

tinv array containing the inverse time constants of the detector model.

`FRACTION` key word to specify the fractions corresponding to each delayed or prompt repons of the detector type model. This option is only valid for platinum, (*NAMTYP*(1:5) = 'PLATN'), detector type.

frac array containing the detector type model fractions.

(det_data) structure describing the format used to read detector information.

2.12.2 Description of the detector data

Note that the information input order must be respected.

Structure **(det_data)**

```
NAME NAMDET
[ NHEX nhex HEX ( ihex(i), i=1,nhex ) ]
  POSITION ( pos(i), i=1,6 )
  RESP ( rep(i), i=1,nrep )
ENDN
```

where

`NAME` key word to specify the detector name.

NAMDET `character*12` name of the detector. The different names in alphabetical order must fit their usual numbering in the core.(Ex: PLATN01, CHION01C)

`NHEX` key word to set the number of hexagons where the detector is placed.

nhex number of hexagons.

`HEX` key word to set the hexagon numbers corresponding to the detector position.

ihex array containing the hexagon numbers where the detector is present, as ordered in the geometry definition (See Section 2.1).

`POSITION` key word to specify the detector coordinates.

pos array containing the positions of the specified detector. The positions must be read as X− X+ Y− Y+ Z− Z+ . For 2-D geometry, Z coordinates must be 0.0 and a value greater than 1.0. For hexagonal geometry, only Z coordinates are used in 3-D representation.

`RESP` key word to specify the detector initial responses.

rep array containing the initial responses of the detector. To use the current detector models in DONJON, responses are given as

- For vanadium detectors: current response, last response.
- For platinum detectors: current response, reference flux, last detector slow responses.
- For ion chamber detectors: current logarithmic response, current log rate response, reference flux.

`ENDN` key word to specify the end of the detector informations.

3. FLUX CALCULATION

In this section, input data of flux calculation modules will be given.

3.1 The BIVADT: module

The BIVADT: module is used to perform a BIVAC-type "tracking" on a 2-D geometry.[10, 11, 12] The geometry is analyzed and a TRACK of type L_BIVAC is produced 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.
- Discretization is performed and the corresponding numbering is saved.
- The unit finite element matrices (mass, stiffness, etc.) are recovered if required.

The calling specifications are:

Structure **(BIVADT:)**

TRACK := BIVADT: [*TRACK*] *GEOM* :: **(bivadt_data)**

where

TRACK character*12 name of the TRACK (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 GEOMETRY file.

(bivadt_data) structure containing the data to module BIVADT:.

3.1.1 Data input for module BIVADT:

Structure **(bivadt_data)**

[EDIT *iprint*]
[TITL *TITLE*]
[MAXR *maxpts*]
[{ PRIM | DUAL } [*ielem icol* [*isplh*]]]
;

where

EDIT key word used to set *iprint*.

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

TITL key word which allows the calculation title to be set.

TITLE the title associated with the DONJON calculation. This title may contain up to 72 characters. The default when TITL is not specified is no title.

MAXR key word which permits the maximum number of regions to be considered during this DONJON calculation to be specified.

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

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

`DUAL` key word to set a mixed-dual finite element discretization.

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. Discretization of a hexagonal geometry is only available with *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 2-D 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.

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;

3.2 The TRIVAT: module

The TRIVAT: module is used to perform a TRIVAC-type "tracking" on a 3-D geometry.[12, 13, 14, 15, 16, 11] The geometry is analyzed and a TRACK of type L_TRIVAC is produced 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.
- 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:

Structure **(TRIVAT:)**

TRACK := TRIVAT: [*TRACK*] *GEOM* :: **(trivat_data)**

where

TRACK character*12 of the TRACK (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 GEOMETRY.

(trivat_data) structure containing the data to module TRIVAT:.

3.2.1 Data input for module TRIVAT:

Structure **(trivat_data)**

```
[ EDIT iprint ]
[ TITL TITLE ]
[ MAXR maxpts ]
[ { PRIM [ ielem [ isplh ] ] | DUAL [ ielem icol ] | MCFD [ ielem [ isplh ] ] | LUMP [ ielem ] } ]
[ VECT [ iseg ] [ PRTV impv ] ]
;
```

where

EDIT key word 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 key word which allows the calculation title to be set.

TITLE the title associated with the DONJON calculation. This title may contain up to 72 characters. The default when TITL is not specified is no title.

MAXR key word which permits the maximum number of regions to be considered during a DONJON 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 GEOD: module but this value is insufficient if symmetries or mesh-splitting are specified.

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

`DUAL` key word to set a mixed-dual finite element discretization. This option is not available for hexagonal geometries.

`MCFD` key word 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` key word 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. Discretization of a hexagonal geometry is only available with *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 2-D or 3-D 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.

`VECT` key word 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` key word 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 *icol* (see section 3.1).

3.3 The `NEWMAC:` module

The `NEWMAC:` module is used to create a MACROLIB. In this object, new properties of each region of the geometry are computed with respect to the device positions. The calling specifications are:

Structure **(NEWMAC:)**

MACRO3 `:=` `NEWMAC:` [*DEVICE*] *MACRO2 NEWGEO* `::` **(newm_data)**

where

MACRO3 `character*12` of the MACROLIB containing the exact properties of each region of the geometry.

DEVICE `character*12` of the DEVICE containing the device information. If *DEVICE* is not specified, the linked list *MACRO2* is copied into *MACRO3*.

MACRO2 `character*12` of MACROLIB containing the extended set of properties created by the `INIMAC:`, `MACD:` or `CRE:` modules.

NEWGEO `character*12` of the GEOMETRY containing the mesh-splitting geometry created by the `USPLIT:` module.

(newm_data) structure containing the data input for the module `NEWMAC:`.

3.3.1 Data input for module `NEWMAC:`

Structure **(newm_data)**

```
[ EDIT iprt ]
{ REFL-SIGF | NOSIGF }
[ TYPE CONTROLLED XFAC xfacc ]
[ TYPE PERT XFAC xfacp ]
;
```

where

`EDIT` key word used to set *iprt*.

iprt index used to control the printing in module `NEWMAC:`. $= 4$ zone limits and device description names are printed, ≥ 5 volume fraction are printed for each device description; for any smaller value, no print are performed. The default value is *iprt* =1 .

`REFL-SIGF` key word used to add explicitly device incremental fission cross sections to non-fissile properties. This option is available to compare with other reactor simulation codes in which incremental fission cross sections are put in the reflector.

`NOSIGF` key word used to set that no fission $\Delta\Sigma$ are added to non-fissile properties. It is the default option.

`TYPE` keyword to specify device type, `CONTROLLED` or `PERT`.

`CONTROLLED` keyword to set `XFAC` for mechanisms under `CONTROLLED` type definition in DEVICE

`PERT` keyword to set `XFAC` for perturbation of the core, which are defined under `PERT` type in DEVICE.

`XFAC` key word to specify the number of cells on which incremental cross sections were computed in the supercell code. For device type other than `CONTROLLED` or `PERT`, *xfac* value set for `CONTROLLED` type is used.

xfacc real for lattice number. For DRAGON code,[17] $xfac$ must be equal to 2.0 and for MULTICELL code, equal to 1.0 . The default value is 1.0.

xfacp real for device incremental cross section correction. For perturbation, it is usually set to 1.0 .By default, *xfacp* is equal to *xfacc*.

3.4 The BIVACA: module

The BIVACA: module is used to compute the finite element system matrices corresponding to a BIVAC TRACK and to a set of nuclear properties in a MACROLIB. The calling specifications are:

Structure **(BIVACA:)**

SYS := BIVACA: [*SYS*] *MACRO TRACK* :: **(bivaca_data)**

where

SYS character*12 name of the SYSTEM containing the system matrices. If *SYS* appears on the RHS, the system matrices previously stored in *SYS* are kept.

MACRO character*12 name of MACROLIB containing the macroscopic cross sections and diffusion coefficients.

TRACK character*12 name of the TRACK (type L_BIVAC) containing the BIVAC tracking.

(bivaca_data) structure containing the data to module BIVACA:.

3.4.1 Data input for module BIVACA:

Structure **(bivaca_data)**

[EDIT *iprint*]
;

where

EDIT key word 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.

3.5 The TRIVAA: module

The TRIVAA: module is used to compute the finite element system matrices corresponding to a TRIVAC TRACK and to a set of nuclear properties. The calling specifications are:

Structure **(TRIVAA:)**

SYS := TRIVAA: [*SYS*] *MACRO TRACK* [*DMACRO*] :: **(trivaa_data)**

where

SYS character*12 name of the SYSTEM containing the system matrices. If *SYS* appears on the RHS, the system matrices previously stored in *SYS* are kept.

MACRO character*12 name of MACROLIB containing the macroscopic cross sections and diffusion coefficients.

TRACK character*12 name of the TRACK (type L_TRIVAC) containing the TRIVAC tracking.

DMACRO character*12 name of the 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:.

3.5.1 Data input for module TRIVAA:

Structure **(trivaa_data)**

```
[ EDIT iprint ]
[ SKIP ] [{ DERI | PERT }] [ UNIT ] [ OVEL ]
;
```

where

`EDIT` key word 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` key word used to skip the system matrix assembly but to perform the $L - D - L^T$ factorization. Use the system matrices already present in *SYS*.

`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 as required by the improved quasistatic (IQS) method.

`UNIT` A system matrix corresponding to cross sections all set to 1.0 is computed. This matrix is useful in the improved quasistatic (IQS) method in cases where the neutron velocities are constant in each energy group.

`OVEL` The reciprocal neutron velocities for each material mixture are recovered from the input MACROLIB *MACRO* and used to compute the corresponding system matrices, as required by the improved quasistatic (IQS) method.

3.6 The FLUD: module

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

Structure **(FLUD:)**

```
FLUX := FLUD: [ FLUX ] SYS TRACK :: (flud_data)
```

where

FLUX `character*12` name of the FLUX containing the solution. If *FLUX* appears on the RHS, the flux solution previously stored in *FLUX* is used to initialize the new iterative process; otherwise, a uniform initial flux vector is used. The solution previously stored must be coherent with the new calculation.

SYS `character*12` name of the SYSTEM containing the system matrices.

TRACK `character*12` name of the TRACK (type `L_BIVAC` or `L_TRIVAC`) containing the tracking informations used to create *SYS*. TRACK must be a linked list.

(flud_data) structure containing the data to module `FLUD:`.

3.6.1 Data input for module FLUD:

Structure **(flud_data)**

```
[ EDIT iprint ]
[ HIST imph ]
[ MAX maxx0 ]
[ VAR1 icl1 icl2 ]
[ ADI nadi ]
[ PREC eps2 ]
[ ADJ ]
[ MONI lmod [ RAND ] ]
;
```

where

`EDIT` key word 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.

`HIST` key word used to set *imph*.

imph index used to control error calculation and convergence histogram storage. $imph = 0$ no histogram and error are printed; = 1 error calculation is obtained by comparing the most thermal flux to a reference one, stored in FLUX in a record REF; = 2 convergence histogram is computed and stored; = 3 ploting informations are stored; = 4 acceleration parameter histograms are stored.

`MAX` key word used to set *maxx0*.

maxx0 maximum number of outer iterations allowed. When this number is reached, a warning message is printed and values of the unknowns at the end of the $maxx0$-th iteration are subsequently used as the solution. The default value is $maxx0 = 250$.

`VAR1` key word used to set the parameters (*icl1* and *icl2*) of the symmetrical variational acceleration technique (SVAT).

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 $icl1 = maxx0$ and $icl2 = 0$.

`ADI` key word used to set *nadi*.

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.

`PREC` key word used to set *eps2*.

eps2 the outer iterations are stopped when the following criteria is reached:

$$\max_i |\Phi_i^{(k-1)} - \Phi_i^{(k)}| \ \leq \ eps2 \times \max_i |\Phi_i^{(k)}|$$

where $\vec{\Phi}^{(k)} = \text{col}\{\Phi_i^{(k)} \ ; \ i = 1, I\}$ is the unknown vector at the k-th outer iteration. The default value is $eps2 = 1.0 \times 10^{-4}$.

`ADJ` key word 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 (module DELTA:) or a quasistatic transient calculation (module IQS:). This option is incompatible with `MONI` option.

`MONI` key word 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 key word* `RAND` *should therefore be used.* This option is incompatible with `ADJ` option.

lmod the *lmod* first bi-orthonormalized harmonics of the solution are computed using the SVAT-accelerated preconditioned power method with a Hotelling deflation procedure.[18]

`RAND` key word 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.

3.7 The `IQS:` module

The IQS: module is used to compute the improved quasistatic (IQS) solution to a space-time kinetics problem corresponding to a set of system matrices.[19] The nodal collocation methods for the discretization are recommended. The calling specifications are:

Structure **(IQS:)**

FLUXP `:=` `IQS:` [*FLUXP*] *FLUX SYSP SYS TRACK DMACRO MACRO GEOM*
`::` **(iqs_data)**

where

FLUXP `character*12` name of the IQS containing the IQS solution. If *FLUXP* appears on the RHS, the solution previously stored in *FLUXP* is used to initialize the new iterative process; otherwise, a uniform unknown vector is used.

FLUX `character*12` name of the FLUX containing the unperturbed flux solution *obtained with the* `ADJ` *key word.*

SYSP `character*12` name of the SYSTEM containing one or many perturbation of the system matrices *obtained with the* `PERT` *and* `UNIT` *key words.* Each perturbation is stored on a `STEP` directory named `TEXT12`, embodying the macro time step index `ISTEP`, using

```
WRITE(TEXT12,'(4HSTEP,I8)') ISTEP
```

Should this directory be absent for a given macro time step, a null variation of the nuclear properties is assumed.

SYS `character*12` name of the SYSTEM containing the reference system matrices.

TRACK `character*12` name of the TRACK (type `L_TRIVAC`) containing the tracking informations. TRACK must be a linked list.

DMACRO `character*12` name of the MACROLIB containing the perturbation over cross sections obtained by `DMAC:` or `MACD:` modules.

MACRO `character*12` name of the MACROLIB containing the reference cross sections stored at the previous time step.

GEOM `character*12` name of the GEOMETRY containing the calculation geometry.

(iqs_data) structure containing the data to module `IQS:`.

When using `NEWMAC:` module to compute a MACROLIB, MACROLIB objects of two consecutive time steps must be stored, and perturbation over properties is obtained by `DMAC:` module.

3.7.1 Data input for module `IQS:`

Structure **(iqs_data)**

```
[ EDIT iprint ]
[ DELN ndg ]
[ TETA [ VTET theta ] ]
[ BETA (beta(i), i=1,ndg) ]
[ DECR (decr(i), i=1,ndg) ]
[ VEL (v(jg), jg=1,ngroup ]
[ CONV epsrho eps2 limm ]
[ KAPS [ DEL del ] [ HI hi ] [ METHOD { GRKA | GRKT } ] END ]
[ MAX maxx0 ]
[ CON2 eramp ercon rhopou ]
[ VAR1 icl1 icl2 ]
[ ADI nadi ]
[ RENO ] [ PKIN ] [ QSM ]
[ POWE pwrint ]
[ TMAC ]
BEGIN
[[ PERT ndiv { STEP | RAMP | CERB } dtmacr END
[ ::: OUT: (out_data) ]
[ REAC ] ]]
;
```

where

`EDIT` key word used to set *iprint*.

iprint index used to control the printing in module `IQS:`. =0 for minimum print; =1 for calucaltion history is printed (default value); =2 more informations on convergence is printed; =3 the solution is printed.

`DELN` key word used to set *ndg*. This data is given if and only if no delayed neutron spectrum $\vec{\chi}_i$ for each precursor group i were given in the module `MACD:` or if *FLUXP* is created. If not specified, the delayed neutron spectrum $\vec{\chi}_i$ for each precursor group i is equal to the fission spectrum $\vec{\chi}_p$.

ndg number of delayed neutron groups.

`TETA` key word used to set the semi-implicit method to compute the precursors concentrations.[20] By default, an analytical calculation is used. It is advisable to use this method only for macro time step *dtmacr* less than equal to 0.5 seconds.

`VTET` key word used to set *theta*.

theta θ parameter for the semi-implicit scheme. The default value is 0.50001, meaning, with the encoding of real value, the use of Crank-Nicholson scheme a little implicitly.

`BETA` key word used to set *beta*(i). These informations must be given if *FLUXP* is created. Otherwise, it can be omitted.

beta β_i delayed neutron fractions associated with group i.

`DECR` key word used to set *derc*(i). These informations must be given if *FLUXP* is created. Otherwise, it can be omitted.

decr λ_i delayed neutron decay rates associated with group i.

`VEL` key word used to set velocities v(jg) for each energy group. By default, the velocities are recovered from the MACROLIB on which *SYSP* is based.

v array of velocity associated with each group.

`CONV` key word used to set *epsrho* and *eps2*.

epsrho convergence criterion in milli-K for stopping the reactivity iterations. The default value is $epsrho = 0.01$ milli-K.

eps2 convergence criterion for stopping the shape distribution iterations. The default value is $eps2 = 1.0 \times 10^{-4}$.

limm maximum number of reactivity iterations. The default value is $limm = 5$.

`KAPS` key word used to set parameters relative to the point-kinetics solution by the Kaps-Rentrop method.[21]

`DEL` key word used to set *del*.

del tolerance value for the Kaps-Rentrop method. The default value is $del = 1.0 \times 10^{-6}$.

`HI` key word used to set *hi*.

hi length of the first integration step in seconds. The default value is $hi = 1.0 \times 10^{-3}$ s.

`METHOD` key word used to set the type of Kaps-Rentrop method.

`GRKA` key word used to set the most stable version of Kaps-Rentrop method.[22] This is the default value.

`GRKT` key word used to set a less stable version of Kaps-Rentrop method.

`END` key word used to terminate the Kaps-Rentrop related options.

`MAX` key word used to set *maxx0*.

maxx0 maximum number of outer iterations allowed for computing the shape distribution. When this number is reached, a warning message is printed and values of the unknowns at the end of the $maxx0$-th iteration are subsequently used as the solution. The default value is $maxx0 = 250$.

`CON2` key word used to set *eramp*, *ercon* and *rhopou*.

eramp convergence criterion for the point kinetics amplitudes. The default value is $eramp = 1.0 \times 10^{-4}$.

ercon convergence criterion for the constraint. The default value is $ercon = 1.0 \times 10^{-4}$.

rhopou criterion on $\Delta\rho^2/\rho$. The default value is $rhopou = 1.0$.

`VAR1` key word used to set the parameters (*icl1* and *icl2*) of the symmetrical variational acceleration technique (SVAT).

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 $icl1 = 250$ (or $icl1 = maxx0$) and $icl2 = 0$.

`ADI` key word used to set *nadi*.

nadi number of alternating direction implicit (ADI) inner iterations per outer iteration. The default value is $nadi = 1$. To set the optimal value of *nadi*, see section 3.6.

`RENO` key word used to force the code to use the constrained version of the IQS method. By default, the unconstrained version is used.

`PKIN` key word used to solve a pure point kinetics problem without spatial effects.

`QSM` key word used to set a generalized quasistatic algorithm (with group-dependent amplitudes). By default, the improved quasistatic option is used.

`POWE` key word used to set *pwrint*.

pwrint initial power of the reactor.To obtain a proper normalization of fluxes, the power must be given in Watts. The default value is $pwrint = 1.0$.

`TMAC` key word used to recover the beginning-of-transient time from *FLUXP* containing a previous IQS solution.

`BEGIN` key word used to begin the time-dependent data.

`PERT` key word used to set the data corresponding to a single macro time step.

ndiv number of equal-time subdivisions for this macro time step.

`STEP` key word used to set a step variation of the cross sections for this macro time step.

`RAMP` key word used to set a ramp variation of the cross sections for this macro time step.

`CERB` key word used to set a CERBERUS-type variation of the cross sections for this macro time step.

dtmacr macro time step in seconds.

`:::` key word used to indicate the call of an embedded module.

`OUT:` key word used to call module `OUT:` used to set the reaction rate calculation at the end of the macro time step.

(out_data) structure containing the data to module `OUT:` (see section 5.1.1).

`REAC` key word used to print additional information on the reactivity.

3.8 The SORKEF: module

The SORKEF: module is used to compute source terms based on a first order perturbation theory over diffusion equation. The direct diffusion equation for system matrix perturbations ΔA and ΔB can be written for a linear perturbation of the flux $\phi = \phi_o + \Delta\phi$:

$$(A_o - \lambda_o B_o)\Delta\phi = -(\Delta A - \lambda_o \Delta B - \Delta\lambda B_o)\phi_o \quad (1)$$

The direct source term is then simply $(\Delta A - \lambda_o \Delta B - \Delta\lambda B_o)\phi_o$ where $\Delta\lambda$ is the first order estimate of the eiganvalue variation, Rayleigh formulation.

The adjoint source terms are easily obtained from a similar expression of the ajoint diffusion equation. The calling specifications are:

Structure **(SORKEF:)**

GPT `:=` `SORKEF:` [*GPT*] *FLUX SYSP SYS TRACK* `::` **(sorkef_data)**

where

GPT `character*12` name of the SOURCE containing the source terms. If *GPT* appears on the RHS, the previous values will be updated.

FLUX `character*12` name of the FLUX containing the unperturbed flux, direct or adjoint.

SYSP `character*12` name of the SYSTEM containing the perturbation of the system matrices.

SYS `character*12` name of the SYSTEM containing the reference system matrices. SYSTEM must be a linked list.

TRACK `character*12` name of the TRACK (type `L_TRIVAC`) containing the tracking informations. TRACK must be a linked list.

(sorkef_data) structure containing the data to module `SORKEF:`.

3.8.1 Data input for module SORKEF:

Structure **(delta_data)**

[`EDIT` *iprint*]
[`ADJ`]
[`CLAS`]
[`NSRC` *isrc*]
[{ `CHAN` *ich* | `ZONE` *iz* }]
;

where

`ADJ` key word used to obtain the adjoint source term. If not present, direct source term will be computed.

`CLAS` key word used to obtain the first variation on the effective multiplication factor without storing the source term. (i.e. classical perturbation theory calculation).

`NSRC` key word to set the source number *isrc*.

isrc source number.This value allows storage of more than one source terms in the same object.

`CHAN` key word to set channel number *ich*. If this option is active, $\Delta\lambda$ for 380 different calls to this module can be stored.

ich channel number. It allows storage in the 380 vector at the correct place.

`ZONE` key word to set zone number *iz*. If this option is active, $\Delta\lambda$ for 14 different calls to this module can be stored.

iz zone number. It allows storage in the 14 vector at the correct place.

3.9 The `DELTA:` module

The DELTA: 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 S is the source term of the direct generalized adjoint equation, this module will solve:

$$(A_o - \lambda_o B_o)\Lambda = -S \tag{2}$$

The calling specifications are:

Structure **(DELTA:)**

DFLUX `:=` `DELTA:` [*DFLUX*] *GPT FLUX SYS TRACK* `::` **(delta_data)**

where

DFLUX `character*12` name of the FLUX containing the solution. If *DFLUX* appears on the RHS, the solution previously stored in *DFLUX* is used to initialize the new iterative process; otherwise, a uniform unknown vector is used.

GPT `character*12` name of the SOURCE containing the sources previously computed with `SORKEF:` module or any other source construction.

FLUX `character*12` name of the FLUX containing the unperturbed flux.

SYS `character*12` name of the SYSTEM containing the reference system matrices. SYSTEM must be a linked list.

TRACK `character*12` name of the TRACK (type `L_TRIVAC`) containing the tracking informations. TRACK must be a linked list.

(delta_data) structure containing the data to module `DELTA:`.

3.9.1 Data input for module `DELTA:`

Data input for module `DELTA:` shows many similarities with data input for module `FLUD:`. The common options are defined only in section 3.6.

Structure **(delta_data)**

```
[ EDIT iprint ]
[ HIST imph ]
[ MAX maxx0 ]
[ VAR1 icl1 icl2 ]
[ ADI nadi ]
[ PREC eps2 ]
[ ADJ ]
;
```

where

`ADJ`	key word used to obtain the solution of an adjoint fixed source eigenvalue problem.

4. DEVICE MOVEMENT AND REFUELLING

In the first part of this section, input data for device movement modules will be given. Then modules for core refuelling will be described. The user must undertand that these modules will only modify existing linked lists or XSM files. No linked list or XSM file will be created.

4.1 The LINKDS: module

The LINKDS: module is used to compute and set controller positions or speed. The controller positions are computed or given in fractional positions with respect to a "full" state of the controller. This means that :

- For LZC units, the positions are given in full filling fraction.
- For rods, one can use fraction of full insertion or fraction of full withdrawal

Whereas controller speeds are in reference to a maximum speed.

The LINKDS: module realizes different actions, in either of them some linked lists are modified and some are used in read-only mode. When device positions in fraction are computed with respect to their actual coordinates, only PROCEDE is modified, whereas DEVICE and GEOMETRY are just read. Though when device speed in cm/s are computed from fractional speed or when positions or speeds are set, both PROCEDE and DEVICE are modified, and GEOMETRY is just read. The calling specifications are:

Structure **(LINKDS:)**

PROCESS [*DEVICE*] := LINKDS: *PROCESS DEVICE GEOM* :: **(linkds_data)**

where

PROCESS character*12 name of the PROCEDE containing the process information required to control devices.

DEVICE character*12 name of the DEVICE containing the device information. If *DEVICE* appears on the LHS side, it must be DEVC option of LINKDS: module.

GEOM character*12 name of the GEOMETRY file containing the geometry information.

(linkds_data) structure containing the data input for the module LINKDS:.

4.1.1 Data input for module LINKDS:

Structure **(linkds_data)**

[EDIT *iprt*]
{ PROC | DEVC }
[[TYPE *NAMTYP*
{ **(proc_data)** | **(dev_data)** }
END]]
;

where

EDIT key word used to set *iprt*.

iprt index used to control the printing. =4 for printing controller names and positions; for any smaller value, no printing is performed. *iprt*=1 is the default value.

`DEVC` key word used to specify an update of controller speed in cm/s or to set a controller positions or speed.

`PROC` key word used to specify an update of the controller positions.

`TYPE` key word used to set *NAMTYP*.

NAMTYP Character*12 name of device type that contains liquid zone controllers and/or rods specifications.

(dev_data) structure describing the format used to read the access to the controllers and their new positions or speeds.

(proc_data) structure describing the format used to read the access to the controllers in PROCEDE that will be updated.

`END` key word used to specify the end of data input for a device type.

4.1.2 Description of the structures (proc_data) and (dev_data)

Structure **(proc_data)**

{ `LZC` *NAMFAL* | [[`ROD` *NAMFAR* { `INSR` | `EXTR` }]] }

Structure **(dev_data)**

{ `LZC` *NAMFAL* | [[`ROD` *NAMFAR*]] }
{ `POS` | `SPEED` [`TIME` *dt*] }
{ `ALL` *value* | [[`SOME` `NAME` *NAMSEQ value*]] `END` | `DIRECT` }

where

`LZC` key word used to set *NAMFAL* for liquid zone controllers.

NAMFAL Character*12 family name of liquid zone controllers as set in DEVICE.

`ROD` key word used to set *NAMFAR* for rods, i.e. adjuster, mechanical or shutdown system rods.

NAMFAR Character*12 family name of rods as set in DEVICE.

`INSR` key word to specify that the rod positions will be computed in fraction of full insertion. This is the default option.

`EXTR` key word to specify that the rod positions will be computed in fraction of full withdrawal.

`POS` key word to specify that the following *value* will be device fractional position.

`SPEED` key word to specify that the following *value* will be device fractional speed.

`TIME` key word used to set *dt*. This keyword is used only for `LZC` device type.

dt time step between two calls of `LINKDS:` module. Required only for `LZC` device type.

`ALL` key word to specify that all device positions or speed are set to the same *value*.

`SOME` key word to specify that only the following device positions or speed are modified.

`NAME` key word used to set *NAMSEQ*.

NAMSEQ Character*12 sequence name of a liquid zone controller or a rod as set in DEVICE.

value value of a device position or speed. For liquid zone controllers, it refers to a full filling fraction or a demand valve in fraction of full filling. This speed is very difficult to calculate because it refers to filling rate. To set a value, it will be better to know it from reactor regulation programs. This value must be between 0.0 and 1.0. If speed is positive, LZC are filled.

For rods, *value* is a rod position in fraction of full insertion or a speed in fraction of full speed. For withdrawal, speed must be negative and for insertion positive, between 0.0 and 1.0. The actual speed in cm/s will also be computed. The full speed of a rod is defined by the maximal time it takes to it to move from fully inserted to withdrawn, or vice-versa.

So the maximal speed of a rod moving along the Y-axis is:

$$v_{max} = \frac{W_{max}}{tins} \tag{3}$$

where W_{max} is the maximum length of the device (see section 2.8.1) and *tins* is the maximum time to a complete insertion or withdrawal of the device (see section 2.9.1).

`DIRECT` key word used to specify that device positions or speeds will be recovered directly from PROCEDE object. It is the case for regulating system transients.

`END` key word used to specify the end of input deck for some devices to be modify.

4.2 The `FLXAXC:` module

The `FLXAXC:` module is used to compute the average flux over bundles and axial flux shapes. Axial flux shapes are composed of the actual fluxes and the previously computed ones. If there are no previous fluxes, their contributions are set to zero. The calling specifications are:

Structure **(FLXAXC:)**

MAPFL `:=` `FLXAXC:` *MAPFL* { *MACRO* | *FLUX TRACK INDEX* } `::` **(flx_data) ;**

where

MAPFL `character*12` name of the MAP containing the fuel bundle locations.

MACRO `character*12` name of a MACROLIB. It is required if and only if a reference calculation will be performed, with no average flux calculation. In fact it is used to recover the number of energy groups.

FLUX `character*12` name of the FLUX or IQS containing the flux vectors.

TRACK `character*12` name of the TRACK containing the geometry tracking information.

INDEX `character*12` name of the INDEX containing the material mixture indices.

(flx_data) structure containing the data to module `FLXAXC:`.

4.2.1 Data input for module FLXAXC:

Structure **(flx_data)**

```
[ FLUX-AV ]
[ AXIAL { COMP | ZONE | REFR { COMP | ZONE } { FLAT | COS } } ]
```

where

FLUX-AV	key word to specify the bundle flux calculation. This option can be omitted if AXIAL option is set.
AXIAL	key word to specify the axial flux shape calculation.
COMP	key word to specify that the flux shapes are evaluated over channels.
ZONE	key word to specify that the flux shapes are evaluated over combustion zones.
REFR	key word used to indicate that an initialization of the flux shapes will be performed.
FLAT	key word used to indicate that the flux shape initialization will be a flat flux.
COS	key word used to indicate that the flux shape initialization will have a sines form.

4.3 The REFUEL: module

The REFUEL: module is used to compute fuel burnups with different approximations, to refuel or shuffle specified channels, and to follow up fuel burnups in time. The calling specifications are:

Structure **(REFUEL:)**

```
MAPFL := REFUEL: MAPFL [ TABFL ] :: (rfl_data) ;
```

where

MAPFL	character*12 name of the MAP containing the fuel flux information.
TABFL	character*12 name of the TABLE created by the XSCONS: module and containing fuel properties. This object is required only for time-average burnup calculations.
(rfl_data)	structure containing the data to module REFUEL:.

4.3.1 Data input for module REFUEL:

Structure **(rfl_data)**

```
[ EDIT iprt ] [ TAVGC [ NITZ nitz ] NS (ns(i), i=1,nzc) NTYP nt TYPE (ityp NAMTAB, i=1,nt) ]
[[ FOLLOW TIME time { SEC | HOUR | DAY | YEAR } WEIGHT mass ]]
[[ REFUEL { NOSHF | SHF } CHAN NAMCHA nsh ]]
[[ NEWFUEL { NOSHF | SHF } CHAN NAMCHA nsh { SOME ( imix(i), i=1,ABS(nsh) ) | ALL imix ]]
[[ SHUFF CHAN NMCHA1 TO { NMCHA2 | POOL } ]]
[ CELL ( ialch(i), i=1,nch) ] [ BEXIT ]
```

where

EDIT	key word used to set *iprt*.

iprt index used to control the printing in module `REFUEL:`. =0 for no print (default value); =3 for printing.

`TAVGC` key word to specify the time-averaged approach to compute burnup values.

`NITZ` key word to set *nitz*.

nitz number of iterations on axial flux shapes when computing burnup ranges. The default value is 4.

`NS` key word to set *ns*(i).

ns refuelling schemes. This value corresponds to the number of bundle shifts must be positive and less then the number of bundles per channel. A value per combustion zone (*nzc*) must be given.

`NTYP` key word used to set *nt*.

nt total number of fuel material types.

`TYPE` key word used to set *ityp*.

ityp identifier of the fuel material type as defined for material mixtures in the `GEOD:` module.

NAMTAB `character*12` directory name in the TABLE where fuel properties are stored.

`FOLLOW` key word to specify that the burnup of each bundle will be incremented over time in considering steady power.

`TIME` key word used to set *time*.

time time of the burnup increment calculation.

`SEC` key word to specify that the time is given in seconds.

`HOUR` key word to specify that the time is given in hours.

`DAY` key word to specify that the time is given in days.

`YEAR` key word to specify that the time is given in years.

`WEIGHT` key word to specify the heavy fissile element weight in bundles.

mass heavy fissile element weight in kg.

`REFUEL` key word to specify a channel refuelling.

`NOSHF` key word to specify the refuelled channel historical information will be stored, and can not be accessed.

`SHF` key word to specify the refuelled channel historical information will be stored and can be accessed. This is the default option.

`CHAN` key word to specify the refuelled channel information.

NAMCHA channel name as defined by *NXNAME* and *NYNAME*. *NAMCHA* is a character*4 variable, constructed as $WRITE(NAMCHA,'(A1,A3)')NYNAME(1:1),NXNAME(1:2)$.

nsh refueling scheme. The absolute value of *nsh* is the number of fuel bundles inserted in the channel *NAMCHA*. The sign of *nsh* define the refueling direction: positive direction is from the first to the *nk*th bundle and negative is from the *nk*th to the first bundle.

`NEWFUEL` key word to specify that a channel will be refuelled with a different type of fuel.

`SOME` key word to specify that the *nsh* values of fuel types can be different.

imix(**i**) index number of a fuel type with respect to the values defined in module `CRE:`.

`ALL` key word to specify that the *nsh* values of fuel types will be identical to *imix*.

`SHUFF` key word to specify that a specified channel will move into an other one or discharge into the pool.

`CHAN` key word to specify the moved channel name.

NMCHA1 channel name as defined by *NXNAME* and *NYNAME*. It is constructed as *NAMCHA*.

`TO` key word to specify the bundle destination.

NMCHA2 channel name as defined by *NXNAME* and *NYNAME*. It is constructed as *NAMCHA*.

`POOL` key word to specify that the channel referenced by *NMCHA1* is discharged into the pool.

`CELL` key word to specify that a patterned age distribution will be input and used to compute instantaneous bundle burnup.

ialch(**i**) refueling sequence number. This channel is refueled the *ialch*(i)th one. The channels are ordering from the top left to the bottom right of the core. The expression of the resulting bundle burnups are given in Ref. [23].

`BEXIT` key word to specify that the average exit burnup of every channel will be computed. This option must be used after a time-average calculation or after the specification in the MAP of the channel fuel scheme.

4.4 The `POWER:` module

The `POWER:` module is used to compute bundle and channel powers using bundle average fluxes obtained from a previous call to **FLXAXC:** module. Fluxes and powers are normalized to a total fission power given as input. The powers can be printed per Z plane, bundle, or as the sum over Z axis, channel. It can be used to obtain average exit burnup, channel and bundle power peaking factors. The calling specifications are:

Structure **(POWER:)**

MAPFL `:=` `POWER:` *MAPFL MACFL* [*MAPFL2*] `::` **(pwr_data)** `;`

where

MAPFL `character*12` name of the MAP containing the fuel flux information.

MACFL `character*12` name of the MACROLIB created by `CRE:` and containing only fuel properties.

MAPFL2 `character*12` name of the MAP containing time-averaged fuel powers. If not present, no channel or bundle power peaking factor calculation can be performed.

(pwr_data) structure containing the data to module `POWER:`.

4.4.1 Data input for module POWER:

Structure **(pwr_data)**

[EDIT *iprt*] [POWER *ptot*] [FSTH *fsth*] [CPPF] [TCOMB]

where

EDIT	key word used to set *iprt*.
iprt	index used to control the printing in module POWER:. =1 for minimum printing (default value); = 2 channel power map is printed ; > 3 bundle power maps are printed.
POWER	key word to specify the power calculation and input the total thermal power used to normalize.
ptot	total thermal power of the reactor in MW.
FSTH	key word to specify the thermal to fission power ratio.
fsth	thermal to fission power ratio. By default, this value is 1.0.
CPPF	key word to specify that the Channel and Bundle Power Peaking Factors are computed with reference to a time-averaged calculation.
TCOMB	key word to specify that a mean combustion rate for the core is computed with respect to the zonal combustion rates.

4.5 The SURVEY: module

The SURVEY: module is used to compute zonal powers and zonal fractions of total power in the interest zones specified in MAP. Calls to **FLXAXC:** and **POWER:** modules are previously required. The calling specifications are:

Structure **(SURVEY:)**

MAPFL := SURVEY: *MAPFL* :: **(surv_data)** ;

where

MAPFL	character*12 name of the MAP containing the fuel bundle powers.
(surv_data)	structure containing the data to module SURVEY:.

4.5.1 Data input for module SURVEY:

Structure **(surv_data)**

[EDIT *iprt*] [FSTH *fsth*] { POW-CONTR | POW-CHAN }

where

EDIT	key word used to set *iprt*.
iprt	index used to control the printing.
FSTH	key word to specify the thermal to fission power ratio.

fsth thermal to fission power ratio. By default, this value is 1.0.

`POW-CONTR` key word to specify that the zones of interest as specified in MAP will be expanded in order to divided by two each channel. This is the default option.

`POW-CHAN` key word to specify that the zones of interest as specified in MAP will be used directly.

4.6 The `MOVDEV:` module

The `MOVDEV:` module is used to compute new controller positions using speed set in `INIDEV:` or `LINKDS:` modules. The calling specifications are:

Structure **(MOVDEV:)**

DEVICE `:=` `MOVDEV:` *DEVICE PROCESS GEOM* `::` **(movd_data)** `;`

where

DEVICE `character*12` name of the DEVICE.

PROCESS `character*12` name of the PROCEDE file containing the process information to access correctly device information.

GEOM `character*12` name of the GEOMETRY file containing the geometry information, resulting from the `USPLIT:` module.

(movd_data) structure containing the data input for the module `MOVDEV:`.

4.6.1 Data input for module `MOVDEV:`

Structure **(movd_data)**

[`EDIT` *iprt*]
[[`TYPE` *NAMTYP* `TIME` *dt*
{ `LZC` *NAMFAL* | [[`ROD` *NAMFAR*]] }]]

where

`EDIT` key word used to set *iprt*.

iprt index used to control the printing. =4 for printing controller names and positions; =5 for description information; for any smaller value, no prints are performed. *iprt*=1 is the default value.

`TYPE` key word used to set *NAMTYP*.

NAMTYP Character*12 name of device type that contains liquid zone controller and adjuster rod specifications as set in DEVICE.

`TIME` key word used to set *dt*.

dt time step between two calls to `MOVDEV:` module.

`LZC` key word used to set *NAMFAL* for liquid zone controllers.

NAMFAL Character*12 family name of liquid zone controllers as set in DEVICE.

`ROD` key word used to set *NAMFAR* for rods.

NAMFAR Character*12 family name of rods as set in DEVICE.

4.7 The DETECT: module

The DETECT: module is used to compute the mean flux at each detector site and the response of each detector. The calling specifications are:

Structure **(DETECT:)**

DETEC := DETECT: *DETEC FLUX TRACK GEOM* :: **(detect_data)** ;

where

DETEC character*12 name of the DETECT containing the detector positions and responses.

FLUX character*12 name of the FLUX or IQS containing the flux solution computed by the FLUD: or IQS: modules.

TRACK character*12 name of the TRACK containing the TRIVAC tracking.

GEOM character*12 name of the GEOMETRY containing the mesh-splitting geometry created by the USPLIT: or GEOD: modules.

(detect_data) structure containing the data to module DETECT:.

4.7.1 Data input for module DETECT:

Structure **(detect_data)**

[EDIT *iprt*] TIME *dt* REF *kc*
[NORM *vnorm*] [SIMEX { SPLINE | PARAB }]

where

EDIT key word used to set *iprt*.

iprt index used to control the printing in module DETECT:. =0 for no print; =1 for minimum printing(default value); =4 for printing each detector name; =5 for finite element numbers and total number of finite elements for each detector.

TIME key word used to set *dt*.

dt time step between two calls to the DETECT: module.

REF key word used to set *kc*.

kc index used to control the type of calculation, =0 for reference calculation; =1 normal calculation. The reference responses are used to obtain detector current responses in full power fractions.

NORM key word used to set *vnorm*.

vnorm value used to normalized responses of all the detectors present in DETECT.

SIMEX key word used to specify that a polynomial interpolation of detector fluxes according to HQSIMEX method. This interpolation will be applied only for vanadium detector, under *NAMTYP* of value VANAD_REGUL.

SPLINE third order polynomial interpolation will be used at detector site with HQSIMEX method.

PARAB second order polynomial interpolation will be used at detector site with HQSIMEX method.

5. OTHER MODULES

In this section, input of output data treatment modules will be given.

5.1 The OUT: module

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

Structure **(OUT:)**

MACNEW `:=` `OUT:` *FLUX MACRO TRACK GEOM* `::` **(out_data)**

where

MACNEW `character*12` name of the extended MACROLIB.

FLUX `character*12` name of the FLUX containing a diffusion solution.

MACRO `character*12` name of the MACROLIB containing the properties used to compute *FLUX* solution.

TRACK `character*12` name of the TRACK containing the tracking informations.

GEOM `character*12` name of the GEOMETRY.

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

5.1.1 Data input for module OUT:

Structure **(out_data)**

[`EDIT` *iprint*]
[`POWR` *power*]
[`INTG` { `IN` | (*ihom*(i), i=1,nreg) }]
;

where

`EDIT` key word used to set *iprint*.

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

`POWR` key word used to set *power*.

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

`INTG` key word used to compute the reaction rates.

`IN` key word for computing the reaction rates on the same mesh previously used to define the geometry (see section 2.1) before mesh-splitting.

ihom index of the homogenized region corresponding to the each region of the geometry (see section 2.1) before mesh-splitting.

5.2 The FLUNRM: module

The FLUNRM: module is used to compute the total power over the used geometry and to normalize fluxes to this power. The calling specifications are:

Structure **(FLUNRM:)**

```
FLUX := FLUNRM: FLUX MACRO TRACK :: (fln_data)
```

where

FLUX character*12 name of the FLUX containing a diffusion solution.

MACRO character*12 name of the MACROLIB containing the properties used to compute *FLUX* solution.

TRACK character*12 name of the TRACK containing the tracking informations.

(fln_data) structure containing the data to module FLUNRM:.

5.2.1 Data input for module FLUNRM:

Structure **(fln_data)**

```
[ EDIT iprint ]
[ CALC ] [ POWER power ]
REF kc ;
```

where

EDIT key word used to set *iprint*.

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

CALC key word used to initiate denormalization of fluxes.

POWER key word used to set *power*. This option is required for $kc = 0$.

power value of the power used to normalize the flux in Watts.

REF key word used to set *kc*.

kc index used to control the type of calculation, =0 for normalization of fluxes to *power*; =1 power calculation using a steady normalization factor.

5.3 The ERROR: module

The ERROR: module is used to compare reaction rates contained into two extended MACROLIBS and to print statistics regarding the comparison. This module does not produce any object, it is used only for printing results.

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

$$P_i^{\text{quandry}} = \frac{\sum_i V_i}{V_i} \frac{P_i}{\sum_i P_i}$$

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:

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

and

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

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

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

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

$$T_{i,g}^{\text{norm}} = \frac{1}{\sum_i \sum_g T_{i,g}}\ T_{i,g}$$

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

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

and

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

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

The calling specifications are:

Structure **(ERROR:)**

```
ERROR: MACREF MACPER ;
```

where

MACREF `character*12` name of the extended MACROLIB used to compute the reference reaction rates with `OUT:` module.

MACPER `character*12` name of the extended MACROLIB used to compute the approximate reaction rates with `OUT:` module.

5.4 The DMAC: module

The DMAC: module is used to perform the differences between two MACROLIB or to add two MACROLIB. This action is possible only if the nuclear properties they contain have the same energy group and material mixture numbers. The calling specifications are:

Structure **(DMAC:)**

```
MACNEW := DMAC: MACRO1 MACRO2 :: (dmac_data)
```

where

MACNEW `character*12` name of the MACROLIB containing either the nuclear increments, from the calculation of *MACRO1* - *MACRO2* or the sum of properties from *MACRO1* + *MACRO2*. Be aware the order of MACROLIB is important, even for addition option.

MACRO1 `character*12` name of a MACROLIB.

MACRO2 `character*12` name of a MACROLIB. When addition is performed, it must contain incremental nuclear properties.

(dmac_data) structure containing the data to module DMAC:.

5.4.1 Data input for module DMAC:

Structure **(dmac_data)**

```
[ EDIT iprt ]
[ STEP ilev ]
[ { ADD | SUB } ] [ { DDIFF | NODIF } ] ;
```

where

`EDIT` key word used to set *iprt*.

iprt index used to control the printing. <= 2 minimum printing; > 3 macroscopic differences are printed.

`STEP` key word used to set *ilev*.

ilev number of the perturbed level in MACROLIB. In case of substraction of two MACROLIB: If a single set of increments is stored, it must be equal to 1. This step is used to later compute perturbation system matrices. If this information is absent, incremental cross sections are stored on root directory. In case of addition of two MACROLIB: *ilev* specifies the perturbed level where informations is stored in the two initial MACROLIB. If both MACROLIB have a perturbed level, it must the same. The resulting properties will be stored on root directory.

`ADD` keyword to specify that the two MACROLIB will be added.

`SUB` keyword to specify that the two MACROLIB will be substracted. This is the default option.

`DDIFF` keyword to specify a correct treatment of diffusion coefficients. If `SUB` is specified, the resulting incremental diffusion coefficient will be:

$$\Delta D = \frac{1}{\frac{1}{D_1} - \frac{1}{D_2}}$$

where D_1 is taken from the first MACROLIB and D_2 from the second. If `ADD` is specified, the resulting diffusion coefficient will be:

$$D = \frac{1}{\frac{1}{D_1} + \frac{1}{\Delta D_2}}$$

where D_1 is taken from the first MACROLIB and ΔD_2 from the second.

`NODIF` keyword to specify that no addition or substraction of diffusion coefficients will be done. This is the default option.

6. EXAMPLES OF INPUT DATA FILES

6.1 IAEA-2D benchmark

The IAEA-2D benchmark is defined in Refs. [10, 24] and its geometry is represented in Fig. 11. Here, it is solved using a parabolic variational collocation method without mesh splitting of the elements with BIVAC method:

Input data file `IAEA2D`

```
LINKED_LIST IAEA MACRO TRACK SYSTEM FLUX EDIT ;
MODULE      GEOD: MACD: BIVADT: BIVACA: FLUD: OUT: END: ;
SEQ_ASCII   FLU ;
*
IAEA := GEOD: :: 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 := MACD: ::
 EDIT 2 NGRO 2 NMIX 4
 READ
 MIX     1
     DIFFX  1.500E+00  4.0000E-01         TOTAL  3.012E-02  8.0032E-02
    NUSIGF  0.000E+00  1.3500E-01     H-FACTORS  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     2
     DIFFX  1.500E+00  4.0000E-01         TOTAL  3.012E-02  8.5032E-02
    NUSIGF  0.000E+00  1.3500E-01     H-FACTORS  0.000E+00  1.3500E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     3
     DIFFX  1.500E+00  4.00000E-01        TOTAL  3.012E-02  1.30032E-01
    NUSIGF  0.000E+00  1.35000E-01    H-FACTORS  0.000E+00  1.35000E-01
      SCAT  1 1 0.0 2 2 0.0 0.2E-01
 MIX     4
     DIFFX  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 := BIVADT: IAEA ::
      TITLE 'IAEA-2D BENCHMARK'
      MAXR 81 DUAL 1 2 ;
SYSTEM := BIVACA: MACRO TRACK ;
FLUX := FLUD: SYSTEM TRACK ::
      EDIT 2 ;
```

```
EDIT := OUT: FLUX MACRO TRACK IAEA ::
       EDIT 2 INTG
       3 2 2 2 3 2 2 1 0
         2 2 2 2 2 2 1 0
           2 2 2 2 1 1 0
             2 2 2 1 0 0
               3 1 1 0 0
                 1 0 0 0
                   0 0 0
                     0 0
                       0
       ;
FLU := FLUX ;
END: ;
```

6.2 Biblis-2D benchmark

The rods-withdrawn configuration of the Biblis-2D benchmark is defined in Ref. [10] and its geometry is represented in Fig. 12. Here, it is solved using a parabolic variational collocation method without mesh splitting of the elements. Then generalized adjoint vectors are computed in reference to perturbation input to **MACD:** module.

Input data file `BIBLIS`

```
LINKED_LIST BIBLIS MACRO TRACK SYSTEM FLUX EDIT DMACRO DSYSTEM DFLUX
            GPT ;
MODULE      GEOD: MACD: TRIVAT: TRIVAA: FLUD: OUT: DELTA:
             END: FREE: DELETE: UTL: SORKEF: ;
SEQ_ASCII  FLUX2 OUT ;
*
BIBLIS := GEOD: :: 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
                           0
       MESHX 0.0 23.1226 46.2452 69.3678 92.4904 115.613 138.7356
             161.8582 184.9808 208.1034
       ;
MACRO := MACD: ::
 EDIT 2 NGRO 2 NMIX 8
 READ
 MIX     1
     DIFFX  1.436000E+00  3.635000E-01
     TOTAL  2.725820E-02  7.505800E-02
    NUSIGF  5.870800E-03  9.606700E-02
 H-FACTORS  2.376800E-03  3.889400E-02
```

```
      SCAT  1 1 0.0 2 2 0.0  1.775400E-02
 MIX     2
     DIFFX  1.436600E+00  3.636000E-01
     TOTAL  2.729950E-02  7.843600E-02
    NUSIGF  6.190800E-03  1.035800E-01
 H-FACTORS  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.762100E-02
 MIX     3
     DIFFX  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
     DIFFX  1.438900E+00  3.638000E-01
     TOTAL  2.746400E-02  9.140800E-02
    NUSIGF  7.452700E-03  1.323600E-01
 H-FACTORS  3.017300E-03  5.358700E-02
      SCAT  1 1 0.0 2 2 0.0  1.710100E-02
 MIX     5
     DIFFX  1.438100E+00  3.665000E-01
     TOTAL  2.729300E-02  8.482800E-02
    NUSIGF  6.190800E-03  1.035800E-01
 H-FACTORS  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.729000E-02
 MIX     6
     DIFFX  1.438500E+00  3.665000E-01
     TOTAL  2.732400E-02  8.731400E-02
    NUSIGF  6.428500E-03  1.091100E-01
 H-FACTORS  2.602600E-03  4.417400E-02
      SCAT  1 1 0.0 2 2 0.0  1.719200E-02
 MIX     7
     DIFFX  1.438900E+00  3.679000E-01
     TOTAL  2.729000E-02  8.802400E-02
    NUSIGF  6.190800E-03  1.035800E-01
 H-FACTORS  2.506400E-03  4.193500E-02
      SCAT  1 1 0.0 2 2 0.0  1.712500E-02
 MIX     8
     DIFFX  1.439300E+00  3.680000E-01
     TOTAL  2.732100E-02  9.051000E-02
    NUSIGF  6.428500E-03  1.091100E-01
 H-FACTORS  2.602600E-03  4.417400E-02
      SCAT  1 1 0.0 2 2 0.0  1.702700E-02
       ;
DMACRO := MACD:  :: EDIT 2  NGRO 2 NMIX 8
  READ
  MIX 1
     TOTAL   1.500E-03   1.000E-03
  ;

TRACK := TRIVAT: BIBLIS ::
      TITLE 'BIBLIS BENCHMARK'
      EDIT 5 MAXR 81 PRIM 2 ;
SYSTEM := TRIVAA: MACRO TRACK ::
      EDIT 1 ;
```

```
FLUX := FLUD: SYSTEM TRACK ::
      EDIT 3 ADJ ;

DSYSTEM := TRIVAA: MACRO TRACK DMACRO ::
      EDIT 1 PERT ;

GPT  := SORKEF: FLUX DSYSTEM SYSTEM TRACK ::
      EDIT 5 ADJ ;

DFLUX := DELTA: GPT FLUX SYSTEM TRACK ::
      EDIT 1 ADJ ;

END: ;
```

6.3 LMW benchmark in 3-D

The LMW benchmark in 3-D is a space-time kinetics problem introduced by Greenman[25] and used by Monier[19]. A plane is represented in Fig. 13. Here, it is solved using a mesh centered finite difference method with mesh splitting of each element along z-axis:

Input data file `LMW43D`

```
LINKED_LIST LMW3D LMW3D2 TRACK MACLIB MACLIB2 MACLIB3 MACOLD
            SYSTEM FLUX DFLUX DEVICE INDEX DSYSTEM PROCEDE DMACRO ;
MODULE GEOD: MACD: TRIVAT: TRIVAA: FLUD: IQS: END:
       USPLIT: NEWMAC: FREE: DELETE: INIMAC:
       INIDEV: INPROC: LINKDS: DMAC: MOVDEV: ;
REAL TIMBC ;
REAL T DT ;
INTEGER I ;
EVALUATE I := 1 ;
EVALUATE DT := 0.5 ;

LMW3D := GEOD: :: CAR3D 6 6 20
      EDIT 1
      X- REFL X+ ZERO
      Y- REFL Y+ ZERO
      Z- ZERO Z+ ZERO
      MIX
          PLAN 1
          5  4  4  5  4  4  ! PLAN 1 Z=0.  A Z=20.
          4  4  4  4  4  4
          4  4  5  4  4  4
          5  4  4  4  4  4
          4  4  4  4  4  4
          4  4  4  4  4  0
          PLAN 2
          1  1  1  1  3  4  ! PLAN 2 Z=20. A Z = 27.5.
          1  1  1  1  3  4
          1  1  1  1  3  4
          1  1  1  3  3  4
          3  3  3  3  4  4
          4  4  4  4  4  0
```

```
          PLAN 3  SAME 2     ! PLAN 3  Z=27.5  A Z=35.
          PLAN 4  SAME 2     ! PLAN 4  Z=35.   A Z=42.5
          PLAN 5  SAME 2     ! PLAN 5  Z=42.5  A Z=48.75
          PLAN 6  SAME 2     ! PLAN 6  Z=48.75 A Z=55.
          PLAN 7  SAME 2     ! PLAN 7  Z=55.   A Z=62.5
          PLAN 8  SAME 2     ! PLAN 10 Z=62.5  A Z=70.
          PLAN 9  SAME 2     ! PLAN 10 Z=70.   A Z=77.5
          PLAN 10 SAME 2     ! PLAN 11 Z=77.5  A Z=85.
          PLAN 11 SAME 2     ! PLAN 12 Z=85.   A Z=92.5.
          PLAN 12 SAME 2     ! PLAN 13 Z=92.5  A Z=100.

          PLAN 13 SAME 2     ! PLAN 14 Z=100.  A Z=102.5
          PLAN 14 SAME 2     ! PLAN 15 Z=102.5 A Z=110.
          PLAN 15 SAME 2     ! PLAN 16 Z=110.  A Z=117.5
          PLAN 16 SAME 2     ! PLAN 17 Z=117.5 A Z=125.
          PLAN 17 SAME 2     ! PLAN 18 Z=125.  A Z=132.5
          PLAN 18 SAME 2     ! PLAN 19 Z=132.5 A Z=140.
          PLAN 19 SAME 2     ! PLAN 20 Z=140.  A Z=180.
          PLAN 20
          4  4  4  4  4  4  ! PLAN 20 Z=180.  A Z=200.
          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

      MESHX  0.0  10.  30.   50.  70.   90.  110.
      MESHY  0.0  10.  30.   50.  70.   90.  110.
      MESHZ  0.0  20.  27.5  35.  42.5  48.75 55.   62.5 70.  77.5
        85. 92.5 100. 102.5 110. 117.5 125.  132.5 140. 180. 200. ;

MACLIB := MACD: ::
    EDIT 1 NGRO 2 NMIX 5 READ

MIX     1
    DIFFX  1.423913E+00  3.563060E-01
    TOTAL  2.795756E-02  8.766217E-02
   NUSIGF  6.477691E-03  1.127328E-01
H-FACTORS  2.591070E-03  4.509310E-02
     SCAT  1 1 0.0 2 2 0.0  0.175555E-01
MIX     2
    TOTAL  5.500000E-04  3.800000E-03
MIX     3
    DIFFX  1.425611E+00  3.505740E-01
    TOTAL  2.817031E-02  9.925634E-02
   NUSIGF  7.503284E-03  1.378004E-01
H-FACTORS  3.001310E-03  5.512106E-02
     SCAT  1 1 0.0 2 2 0.0  0.1717768E-01
MIX     4
    DIFFX  1.634227E+00  2.640020E-01
    TOTAL  3.025750E-02  4.936351E-02
     SCAT  1 1 0.0 2 2 0.0  0.2759693E-01
MIX     5
```

```
    DIFFX  1.423913E+00  3.563060E-01
    TOTAL  2.795756E-02  8.766217E-02
   NUSIGF  6.477691E-03  1.127328E-01
H-FACTORS  2.591070E-03  4.509310E-02
     SCAT  1 1 0.0 2 2 0.0  0.175555E-01 ;

LMW3D2 INDEX := USPLIT: LMW3D ::
     MAXR 730 ;

DEVICE INDEX := INIDEV: INDEX LMW3D2 ::
 NDEV 4  NODEL  NDESC 4
DEVICE CONTROLLED ADJUSTER ADJ011
        50.  70.   0.  10.   0.  200.    Z  0.0  0.0  0.0
ADJ111  50.  70.   0.  10.   0.  200.    2    0
ENDDEV
DEVICE CONTROLLED ADJUSTER ADJ012
         0.  10.  50.  70.   0.  200.    Z  0.0  0.0  0.0
ADJ112   0.  10.  50.  70.   0.  200.    2    0
ENDDEV
DEVICE CONTROLLED ADJUSTER ADJ021
        30.  50.  30.  50.   0.  200.    Z  0.0  0.0  0.0
ADJ121  30.  50.  30.  50.   0.  200.    2    0
ENDDEV
DEVICE CONTROLLED ADJUSTER ADJ022
         0.  10.   0.  10.   0.  200.    Z  0.0  0.0  0.0
ADJ122   0.  10.   0.  10.   0.  200.    2    0
ENDDEV ;

MACLIB2 := INIMAC: INDEX MACLIB ;

EVALUATE TIMBC := 200. 3. / ;
PROCEDE := INPROC: DEVICE ::
 EDIT 2 TYPE CONTROLLED  END
 ROD  ADJUSTER  NAME  ADJ011   NB 1 NBANC 1 TIME <<TIMBC>> END
 ROD  ADJUSTER  NAME  ADJ012   NB 2 NBANC 1 TIME <<TIMBC>> END
 ROD  ADJUSTER  NAME  ADJ021   NB 1 NBANC 2 TIME <<TIMBC>> END
 ROD  ADJUSTER  NAME  ADJ022   NB 2 NBANC 2 TIME <<TIMBC>> END
  ;

PROCEDE := LINKDS: PROCEDE DEVICE LMW3D2 ::
   EDIT 4 PROC
   TYPE CONTROLLED
    ROD ADJUSTER INSR
   END ;
PROCEDE DEVICE := FREE: PROCEDE DEVICE ;

PROCEDE DEVICE := LINKDS: PROCEDE DEVICE LMW3D2 ::
   EDIT 4  DEVC
   TYPE CONTROLLED
    ROD ADJUSTER  POS
    SOME NAME ADJ011  0.5
    SOME NAME ADJ012  0.5
    SOME NAME ADJ021  0.1
```

```
      SOME NAME ADJ022  0.1 END
     END
     TYPE CONTROLLED
      ROD ADJUSTER SPEED
      ALL 0.0
     END ;

TRACK := TRIVAT: LMW3D2 ::
     EDIT 1
     TITLE 'LMW 3D TRANSIENT PROBLEM'
     MAXR 730 MCFD 1 ;

MACLIB3 := NEWMAC: DEVICE MACLIB2 LMW3D2 ::
      EDIT 5 REFL-SIGF
      TYPE CONTROLLED XFAC 1.0 ;

DEVICE MACLIB2 LMW3D2 := FREE: DEVICE MACLIB2 LMW3D2 ;
MACOLD := MACLIB3 ;
MACLIB3 := FREE: MACLIB3 ;
MACLIB3 := DELETE: MACLIB3 ;

SYSTEM := TRIVAA: MACOLD TRACK :: EDIT 1 ;

FLUX := FLUD: SYSTEM TRACK :: EDIT 2 ADI 2 PREC 1.E-6 ADJ ;
MACOLD TRACK SYSTEM := FREE: MACOLD TRACK SYSTEM ;

EVALUATE T := DT ;

REPEAT

  ECHO " TIME " T " ITERATION " I ;

  IF T 7.5 >  T 48.0 <  *   THEN
    PROCEDE DEVICE := LINKDS: PROCEDE DEVICE LMW3D2 ::
       EDIT 1 DEVC
       TYPE CONTROLLED
        ROD ADJUSTER  SPEED
        SOME NAME ADJ021 1.0
        SOME NAME ADJ022 1.0 END
       END ;
   ELSE
    PROCEDE DEVICE := LINKDS: PROCEDE DEVICE LMW3D2 ::
       EDIT 1 DEVC
       TYPE CONTROLLED
        ROD ADJUSTER  SPEED
        SOME NAME ADJ021 0.0
        SOME NAME ADJ022 0.0 END
       END ;
   ENDIF ;
   IF T 27.0 > THEN
     PROCEDE DEVICE := LINKDS: PROCEDE DEVICE LMW3D2 ::
       EDIT 1 DEVC
       TYPE CONTROLLED
```

```
          ROD ADJUSTER  SPEED
          SOME NAME ADJ011 0.0
          SOME NAME ADJ012 0.0 END
         END ;
     ELSE
       PROCEDE DEVICE := LINKDS: PROCEDE DEVICE LMW3D2 ::
         EDIT 1 DEVC
         TYPE CONTROLLED
           ROD ADJUSTER SPEED
           SOME NAME ADJ011 -1.0
           SOME NAME ADJ012 -1.0 END
         END ;
     ENDIF ;
     PROCEDE DEVICE := FREE: PROCEDE DEVICE ;

     DEVICE := MOVDEV: DEVICE PROCEDE LMW3D2 ::
         EDIT 6 TYPE CONTROLLED
         TIME  <<DT>>
         ROD  ADJUSTER ;
     PROCEDE DEVICE LMW3D2 := FREE: PROCEDE DEVICE LMW3D2 ;

    MACLIB3 := NEWMAC: DEVICE MACLIB2 LMW3D2 ::
       EDIT 5 REFL-SIGF
       TYPE CONTROLLED XFAC 1.0 ;
    DEVICE MACLIB2 LMW3D2 := FREE: DEVICE MACLIB2 LMW3D2 ;

    DMACRO := DMAC: MACLIB3 MACOLD ::
       EDIT 1 STEP 1 ;
    MACLIB3 MACOLD := FREE: MACLIB3 MACOLD ;

    DSYSTEM := TRIVAA: MACOLD TRACK DMACRO ::
       EDIT 1 PERT UNIT ;
    DMACRO TRACK MACOLD := FREE: DMACRO TRACK MACOLD ;

    IF I 1 = THEN
      DFLUX := IQS:  FLUX DSYSTEM SYSTEM TRACK DMACRO MACOLD LMW3D2 ::
       EDIT 0 DELN 6
*      TETA
       BETA 0.000247 0.0013845 0.001222
            0.0026455 0.000832 0.000169
       DECR 0.0127 0.0317 0.115  0.311  1.40  3.87
       VEL   1.25E7     2.5E5
       CONV  0.1E-1 0.5E-5 10 MAX 40
       CON2 .5E-4 5.E-6 1.
*      KAPS DEL 1.E-6  HI 1.E-3 METHOD GRKT END
       RENORM
       POWER 150.
       BEGIN
       PERT 1 RAMP  <<DT>> END
       ;
    ELSE
      DFLUX := IQS: DFLUX FLUX DSYSTEM SYSTEM TRACK
                              DMACRO MACOLD LMW3D2 ::
```

```
      EDIT 0
*      TETA
      VEL   1.25E7     2.5E5
      CONV  0.1E-1 0.5E-5 10 MAX 40
      CON2 .5E-4 5.E-6 1.
*      KAPS DEL 1.E-6  HI 1.E-3 METHOD GRKT END
      RENORM
      POWER 150.
      TMAC
      BEGIN
      PERT 1 RAMP  <<DT>>  END
      ;
   ENDIF ;
   FLUX DSYSTEM SYSTEM DMACRO MACOLD TRACK LMW3D2 := FREE:
      FLUX DSYSTEM SYSTEM DMACRO MACOLD TRACK LMW3D2 ;

   DSYSTEM SYSTEM DMACRO MACOLD := DELETE:
       DSYSTEM SYSTEM DMACRO MACOLD  ;

   MACOLD := MACLIB3 ;
   MACLIB3 := FREE: MACLIB3 ;
   MACLIB3 := DELETE: MACLIB3 ;

   SYSTEM := TRIVAA: MACOLD TRACK :: EDIT 1 ;

   IF I 30 = THEN
     EVALUATE DT := 2.0833 4. / ;
   ENDIF ;
   IF I 38 = THEN
     EVALUATE DT := 0.5 ;
   ENDIF ;
   IF I 63 = THEN
     EVALUATE DT := 0.834 ;
   ENDIF ;
   IF I 64 = THEN
     EVALUATE DT := 0.5 ;
   ENDIF ;

   ECHO " EVALUATE DT " DT ;

   EVALUATE  T := T DT + ;
   EVALUATE  I := I 1 + ;

 UNTIL T 1.0 > ;
* UNTIL T 60.5 > ;

 END: ;
```

6.4 Slowpoke-2 core

The following input file is used to represent HEU Slowpoke-2 core without its reflector in 2-D. Nuclear properties are recovered from COMPO files created by DRAGON for 6 energy groups. The resulting eigenvalue of this case is about 0.9507.

Input data file SLOW

```
LINKED_LIST  GEOMETRY TRACKS SYSTEM FLUX MACRO LCEL1 LCEL7 LCLUS ;
MODULE       GEOD: TRIVAT: CRE: TRIVAA: FLUD: END: ;
SEQ_ASCII    HCLUS  ;

GEOMETRY := GEOD: :: HEX 133
          HBC R120 ALBE 0.42
          SIDE 0.637181
          MIX
          3                                                 !COUR  1
          3 3                                               !COUR  2
          1 1 1 1                                           !COUR  3
          1 1 1 1 1 1                                       !COUR  4
          1 1 4 1 1 1 4 1                                   !COUR  5
          1 1 1 1 4 1 1 1 1 4                               !COUR  6
          1 1 4 1 1 1 1 1 4 1 1 1                           !COUR  7
          1 4 1 1 4 1 1 1 4 1 1 4 4 1                       !COUR  8
          1 1 1 4 1 1 1 4 1 1 1 4 1 1 1 4                   !COUR  9
          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                 !COUR 10
          1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2             !COUR 11
          2 2 2 2 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2 2 ; !COUR 12

! COMPO FILES FROM WIMSLIB WITH 6 ENERGY GROUPS
LCLUS := HCLUS ;

MACRO  := CRE: LCLUS ::
     EDIT 1 NMIX 4
     READ COMPO LCLUS  MIX 1  'ZCOMBU     1' ENDMIX !  COMBUSTIBLE
                       MIX 4  'ZCOMBU     2' ENDMIX !  SITE INOCCUPE
                       MIX 2  'PERIPH     1' ENDMIX !  REFLECTEUR
                       MIX 3  'ZCENTR     1' ENDMIX !  EAU CENTRALE
 ;
TRACKS := TRIVAT: GEOMETRY ::
    TITL 'HEU Model SLOWPOKE-2 Core (EPM) '
    MAXR 795  MCFD  1   1  ! Full Hexagons
     ;
!
SYSTEM := TRIVAA: MACRO TRACKS ;
!
FLUX := FLUD: SYSTEM TRACKS ::
      EDIT 2  ADI 2 ;
END: ;
```

6.5 CANDU Supercell Calculation

CANDU Supercell is defined in Ref. [23]. This supercell allows the calculation of the channel and bundle peaking factors in the reactor. A time-average representation of bundle fuel properties is first used with an axial flux shape convergence. Then a refueling sequence of the channels are input and used to obtain an instantaneous bundle burnup map. The Channel Power Peaking Factor is so computed. In this example, input procedures are used to ease the total input reading and insure proper use of the required modules. These procedures are compiled by the GAN kernel.[3] The resulting files `.o2m` can be stored and used for subsequent calculations, minimizing the execution time.

Input data file `CellC`

```
!*********************************************************************
!
!              TEST OF CELL 6*6 FOR TAV CALCULATION
!
!*********************************************************************

 PROCEDURE    PgeoCell PresCell Pmac Pmaktab PfluxC ;
 LINKED_LIST  G23D2 MACRO INDEX MACRO2 RESTAV SYSTEM
              FLUX TRACK RESEAU MACRES TAB ;
 MODULE       TRIVAT: TRIVAA: FLUD: LINKDS: FREE: DELETE: INIRES:
              POWER: END: INIMAC: GREP: CRE: FLXAXC: REFUEL: REFRES: ;
 STRING       NamCpo1 NamCpo2  Namtab Refl ;
 INTEGER      Int   ;
 REAL         Dburn Epsf B1 K1 Out ;
 REAL         fmean Xfacc Xfacp ;
 REAL         Precf := 1.E-4  ;
 CHARACTER    comb := "TAVGC" ;
! type = 1 axial flux error ; = 2 keff - 1
 INTEGER      iter iterf type edit := 0 0 1 2 ;
 REAL         epsillon := 1.E-4 ;
*
 G23D2 INDEX  := PgeoCell ;
 MACRO := Pmac ;
*
 EVALUATE B1 := 10000.  ;
 RESEAU := PresCell :: <<B1>> ;
*
 INDEX RESEAU := REFRES: INDEX RESEAU G23D2 ;
 INDEX RESEAU G23D2 := FREE: INDEX RESEAU G23D2 ;
*
 EVALUATE NamCpo1 NamCpo2 := "MIXTRXE" "1" ;
 EVALUATE Namtab := "NATURAL"   ;
 EVALUATE Int Dburn := 1  250.  ;
 TAB   := Pmaktab RESEAU :: <<NamCpo1>> <<NamCpo2>> <<Namtab>>
                       <<Int>>  "BURNUP" <<Dburn>> ;
 RESEAU := FREE: RESEAU ;
*
**********************************************************************
! TIME-AVERAGE CORE DEFINITION
 TRACK := TRIVAT: G23D2 ::  EDIT 1 TITLE '6*6 CELL CALCULATION'
                            MAXR 432 MCFD 1 ;
*
```

```
 RESEAU := FLXAXC: RESEAU MACRO ::  AXIAL  REFR COMP FLAT ;
 RESEAU MACRO := FREE: RESEAU MACRO ;
 ECHO "MAIN: EXIT BURNUP " B1 ;
*
 EVALUATE Epsf := epsillon 1. + ;
 EVALUATE iterf := 0 ;
*
* ITERATION OVER FLUX SHAPE CONVERGENCE
*------------
 WHILE Epsf epsillon > iterf  10 < * DO
*------------
  EVALUATE iterf := iterf 1 + ;
  RESEAU := REFUEL: RESEAU TAB :: TAVGC NS 2 NTYP 1 TYPE 1 <<Namtab>> ;
  RESEAU TAB := FREE: RESEAU TAB ;
*
 MACRES := CRE: TAB RESEAU :: READ
      TABLE TAB  MIX  1  <<Namtab>> BURNUP 0.0 <<comb>>  ENDMIX ;
 RESEAU  TAB := FREE: RESEAU  TAB ;
 MACRO2 := INIMAC: INDEX MACRO MACRES ;
 INDEX MACRO MACRES := FREE: INDEX MACRO MACRES ;
 SYSTEM := TRIVAA: MACRO2 TRACK :: EDIT 1 ;
 IF iterf 1 = THEN
   FLUX := FLUD: SYSTEM TRACK ::  EDIT 1  ADI 2 PREC <<Precf>>   ;
 ELSE
   FLUX := FLUD: FLUX SYSTEM TRACK :: EDIT 1  ADI 2 PREC <<Precf>> ;
 ENDIF ;
 MACRO2 SYSTEM TRACK := FREE: MACRO2 SYSTEM TRACK ;
 MACRO2 SYSTEM := DELETE: MACRO2 SYSTEM ;
 RESEAU := FLXAXC: RESEAU FLUX TRACK INDEX :: AXIAL COMP ;
 FLUX TRACK INDEX := FREE: FLUX TRACK INDEX ;
 RESEAU := POWER: RESEAU MACRES :: EDIT <<edit>> POWER 230.4  ;
 MACRES := FREE: MACRES ;
 MACRES := DELETE: MACRES ;
 GREP: RESEAU ::  GETVAL EPSF-AX 1 >>Out<< ;
 RESEAU := FREE: RESEAU ;
 EVALUATE Epsf := Out ;
 ECHO "MAIN: AXIAL FLUX ERROR " Epsf "ITERATIONS " iterf ;
*-----------
 ENDWHILE ;
*-----------
*
EVALUATE edit := 5 ;
RESEAU := REFUEL: RESEAU TAB :: TAVGC NS 2 NTYP 1 TYPE 1 <<Namtab>> ;
RESEAU TAB := FREE: RESEAU TAB ;
RESEAU := PfluxC RESEAU TAB INDEX MACRO G23D2 ::
    <<comb>> <<Namtab>> 432  <<Precf>> <<type>> 230.4 <<edit>> >>K1<<  ;
RESEAU INDEX MACRO G23D2 := FREE: RESEAU INDEX MACRO G23D2 ;
EVALUATE Epsf := K1 ;
ECHO "MAIN: AXIAL FLUX ERROR " Epsf "ITERATIONS " iterf ;
*
RESTAV := RESEAU ;
RESTAV RESEAU := FREE: RESTAV RESEAU ;
ECHO "MAIN: INSTANTANEOUS COMPUTATION FOR CELL 36" ;
```

```
RESEAU := REFUEL: RESEAU ::
   CELL   22 29  8 17 14 27
           7 16 23 34  9  4
          30 11 28 19 24 35
           3 20 33  2 21  6
          12 25 10 15 26 31
           5 36 13 32  1 18  ;
MACRES := CRE: TAB RESEAU ::  EDIT 1  READ
      TABLE TAB  MIX  1  <<Namtab>> BURNUP 0.0 DIRECTC  ENDMIX ;
RESEAU := FREE: RESEAU ;
MACRO2 := INIMAC: INDEX MACRO MACRES ;
SYSTEM := TRIVAA: MACRO2 TRACK  ;
FLUX := FLUD: FLUX SYSTEM TRACK ::     EDIT 1 PREC <<Precf>>   ;
RESEAU := FLXAXC: RESEAU FLUX TRACK INDEX ::  AXIAL COMP ;
FLUX TRACK := FREE: FLUX TRACK ;
RESEAU := POWER: RESEAU MACRES RESTAV :: EDIT 5 POWER 230.4 CPPF ;
RESEAU INDEX MACRO G23D2 MACRES := FREE:
RESEAU INDEX MACRO G23D2 MACRES ;
END: ;
```

Input procedure for geometry definition `PgeoCell.c2m`

```
!
! PROCEDURE:  PgeoCell
! USAGE:      Geometry construction for Cell 6*6
! AUTHOR:     E.Varin (97/08/11)
! CALL:
!
!  GEOM2 INDEX := PgeoCell   ;
!
!
! NOTES:
!
!       Cell 6*6 is a CAR3D geometry of 6*6*12 cells
!       used to reproduce CANDU-6 behavior in a smaller
!       number of regions
!
!       The boundary conditions are VOID and REFL
!       It is an infinite reflective cell
!         Mix  # 1 for the fuel
!
!
!
 PARAMETER   GEOM2 INDEX ::
 ::: LINKED_LIST GEOM2 INDEX ; ;
 LINKED_LIST GEOM  ;
 MODULE      END: GEOD: USPLIT: READ: ;
!  BASIC GEOMETRY DESCRIPTION
*
 GEOM := GEOD: :: CAR3D 6 6 12
     EDIT  1
                X- REFL      X+ REFL
                Y- REFL      Y+ REFL
                Z- VOID      Z+ VOID
     MIX
           PLAN    1

               1 1 1  1 1 1
               1 1 1  1 1 1
               1 1 1  1 1 1

               1 1 1  1 1 1
               1 1 1  1 1 1
               1 1 1  1 1 1

            PLAN 2  SAME 1
            PLAN 3  SAME 1
            PLAN 4  SAME 1
            PLAN 5  SAME 1
            PLAN 6  SAME 1
            PLAN 7  SAME 1
            PLAN 8  SAME 1
            PLAN 9  SAME 1
            PLAN 10 SAME 1
```

```
             PLAN 11 SAME 1
             PLAN 12 SAME 1


              MESHX  0.0   28.575 57.15  85.725 114.3  142.875 171.45
              MESHY  0.0   28.575 57.15  85.725 114.3  142.875 171.45
              MESHZ  0.0   49.53  99.06 148.59  198.12 247.65  297.18
                   346.71 396.24 445.77 495.30  544.83 594.36
   ;
  GEOM2 INDEX := USPLIT: GEOM :: MAXR 432 ;
END: ;
```

Input procedure for reflector property definition `Pmac.c2m`

```
!
! PROCEDURE:  Pmac
! USAGE:      Macrolib construction for a fixed material
! AUTHOR:     E.Varin (97/08/11)
! CALL:
!
!  MACRO := Pmac    ;
!
!
! NOTES:
!
!    All the sequential ASCII files are exported COMPO files
!    and must have the names and directory hierarchy as used
!    in the calling to CRE: module
!
!
!
 PARAMETER   MACRO ::
  ::: LINKED_LIST MACRO ;  ;
 LINKED_LIST NREFL ;
 SEQ_ASCII   REFLn ::
    FILE 'REFLn' ;
 MODULE      CRE: END: ;
*
 NREFL  := REFLn     ;
 MACRO := CRE: NREFL ::  NMIX 2  READ
       COMPO NREFL   MIX    2      'MIXTMOD    1'  ENDMIX ;
 END: ;
```

Input procedure for fuel map definition `PresCell.c2m`

```
!
! PROCEDURE:  PresCell
! USAGE:      Fuel map description for 6*6 Cell
! AUTHOR:     E.Varin (97/08/11)
! CALL:
!
!  RESEAU := PresCell :: B1    ;
!
! NOTES:
!         B1 = average exit burnup
!
!
!
 PARAMETER   RESEAU ::
  ::: LINKED_LIST RESEAU ; ;
 MODULE      END: INIRES:  ;
 REAL        B1  ;

 :: >>B1<<   ;

 RESEAU := INIRES: ::  NBUND  12  NCHAN  36  NZONE  1
     ::: GEOD:  CAR3D 6 6 12
        EDIT  5
                X- VOID      X+ VOID
                Y- VOID      Y+ VOID
                Z- VOID      Z+ VOID
        MIX
                PLAN    1
                   1 1 1  1 1 1
                   1 1 1  1 1 1
                   1 1 1  1 1 1

                   1 1 1  1 1 1
                   1 1 1  1 1 1
                   1 1 1  1 1 1

                  PLAN 2  SAME 1
                  PLAN 3  SAME 1
                  PLAN 4  SAME 1
                  PLAN 5  SAME 1
                  PLAN 6  SAME 1
                  PLAN 7  SAME 1
                  PLAN 8  SAME 1
                  PLAN 9  SAME 1
                  PLAN 10 SAME 1
                  PLAN 11 SAME 1
                  PLAN 12 SAME 1

             MESHX  0.0   28.575 57.15  85.725 114.3  142.875 171.45
             MESHY  0.0   28.575 57.15  85.725 114.3  142.875 171.45
             MESHZ  0.0   49.53  99.06 148.59  198.12 247.65  297.18
                  346.71 396.24 445.77 495.30  544.83 594.36  ;
```

```
              NXNAME  '1' '2' '3' '4' '5' '6'
              NYNAME  'A' 'B' 'C' 'D' 'E' 'F'
              ZONE
                       1 1 1  1 1 1
                       1 1 1  1 1 1
                       1 1 1  1 1 1

                       1 1 1  1 1 1
                       1 1 1  1 1 1
                       1 1 1  1 1 1

              CPPF-ZONE
                       1 1 1  1 1 1
                       1 1 1  1 1 1
                       1 1 1  1 1 1

                       1 1 1  1 1 1
                       1 1 1  1 1 1
                       1 1 1  1 1 1
              BURN-ZC  <<B1>>
  ;
END: ;
```

Input procedure for fuel property table `Pmaktab.c2m`

```
!
! PROCEDURE:  Pmaktab
! USAGE:      Table construction for Gentilly2 reactor from DRAGON
!             fuel properties
! AUTHOR:     E.Varin (96/02/12)
! CALL:
!
!  TAB := Pmaktab RESEAU ::   NamCpo   Namtab  Int  intTYP  Dburn ;
!
!
! NOTES:
!
!        NamCpo = directory name for fuel properties in compo file
!        Namtab = directory name for fuel properties in table TAB
!        Int = property interpolation with respect to burnup
!        IntTYP = { BURNUP | N/KB }
!        Dburn = burnup step for interpolation in MWd/t-U
!
!    All the sequential ASCII files are exported COMPO files
!    and must have the names and directory hierarchy as used
!    in the calling to CRE: module
!
!
!
 PARAMETER   TAB RESEAU ::  ::: LINKED_LIST TAB RESEAU ; ;
 LINKED_LIST NFUEL ;
 SEQ_ASCII   FUELn ::
     FILE 'FUELn' ;
 MODULE      XSCONS: END: ;
 STRING      NamCpo NamCpo1 NamCpo2  Namtab intTYP ;
 INTEGER     Int   ;
 REAL        Dburn ;
*
* READ INPUT PARAMETERS
 :: >>NamCpo1<<  >>NamCpo2<<  >>Namtab<<  >>Int<<  >>intTYP<<  ;
*
 NFUEL := FUELn ;
 EVALUATE NamCpo := NamCpo1 "    " NamCpo2 + + ;
 IF Int 0 =  THEN
     TAB := XSCONS: NFUEL :: EDIT 0 READ
       COMPO NFUEL   TYPE <<Namtab>>  <<NamCpo>>
                     <<intTYP>> MICRO ALL ENDTYP ;
 ELSE
*   INTERPOLATION OVER BURNUP
    :: >>Dburn<< ;
    TAB := XSCONS: NFUEL ::  EDIT 0 READ
       COMPO NFUEL   TYPE <<Namtab>>  <<NamCpo>>
               <<intTYP>> MICRO ALL INTRPL <<Dburn>> ENDTYP ;
 ENDIF ;
 END: ;
```

Input procedure for flux and power calculation `PfluxC.c2m`

```
!
! PROCEDURE:  PfluxC
! USAGE:      Flux calculation for Reactor without Device
! AUTHOR:     E.Varin (97/08/11)
! CALL:
!
!  RESEAU := Pflax RESEAU TAB INDEX MACRO G23D2 ::
!     Comb Namtab Maxreg Precf type edit   Out ;
!
!
! NOTES:
!
!        Comb   = { TAVGC | HOMG }
!        Namtab = directory name for fuel properties in table TAB
!        Maxreg = maximum number of matrial regions in geometry
!        Precf= Static flux calculation precision
!        edit = Edition level in module POWER:. (=2 for power maps)
!        type = = 1 axial flux error ; = 2 k-effective - 1.
!
!
!
 PARAMETER   RESEAU TAB INDEX MACRO G23D2 ::
  ::: LINKED_LIST TAB G23D2 RESEAU MACRO INDEX ; ;
 LINKED_LIST MACRES MACRO2  TRACK MACRO3  SYSTEM FLUX  ;
 REAL        Out Precf Pow ;
 CHARACTER   Namtab comb ;
 INTEGER     type edit Maxreg ;
 MODULE      TRIVAT: TRIVAA: NEWMAC: FLUD: FREE: END: GREP:
             INIMAC: CRE: REFUEL: FLXAXC: POWER: ;

 :: >>comb<< >>Namtab<< >>Maxreg<< >>Precf<< >>type<< >>Pow<<
    >>edit<<  ;

 IF comb 'TAVGC' <> comb 'HOMG' <> * THEN
   PRINT 'PROC: ILLEGAL KEYWORD ' comb ;
 ENDIF ;
 MACRES := CRE: TAB RESEAU ::   EDIT 1  READ
      TABLE TAB  MIX  1  <<Namtab>> BURNUP 0.0 <<comb>>  ENDMIX ;
 RESEAU := FREE: RESEAU ;
 MACRO2 := INIMAC: INDEX MACRO MACRES ;
 TRACK := TRIVAT: G23D2 ::  EDIT 1  TITLE 'FLUX CALCULATION'
                            MAXR <<Maxreg>> MCFD 1 ;
 SYSTEM := TRIVAA: MACRO2 TRACK :: EDIT 1 ;
 FLUX := FLUD: SYSTEM TRACK :: EDIT 1  PREC <<Precf>>   ;
 RESEAU := FLXAXC: RESEAU FLUX TRACK INDEX :: AXIAL COMP ;
 FLUX TRACK := FREE: FLUX TRACK ;
 RESEAU := POWER: RESEAU MACRES :: EDIT <<edit>> POWER <<Pow>> ;
 MACRES := FREE: MACRES ;
 IF type 1 = THEN
  GREP: RESEAU :: GETVAL EPSF-AX 1 >>Out<< ;
 ELSE
  IF type 2 = THEN
```

```
      GREP: FLUX :: GETVAL K-EFFECTIVE 1 >>Out<< ;
      EVALUATE Out := Out 1. - ;
     ELSE
      PRINT ' ERREUR DANS LA VALEUR DE type ' ;
      END: ;
     ENDIF ;
    ENDIF ;
    :: <<Out>> ;

    END: ;
```

7. REFERENCES

[1] E. Varin, S. Navarro, J. Koclas and R. Roy,“ Xsimul User's Guide - Version 1.0 ”, Report IGE-178, École Polytechnique de Montréal, Institut de Génie Nucléaire (July 1995).

[2] A. Hébert, “A User's Guide for TRIVAC-3,” Report IGE-161, École Polytechnique de Montréal, Institut de Génie Nucléaire (December 1994).

[3] R. Roy and A. Hébert, ‘The GAN Generalized Driver,” Report IGE-158, École Polytechnique de Montréal, Institut de Génie Nucléaire (March 2000).

[4] R. Roy, “The CLE-2000 Tool-box,” Report IGE-163, École Polytechnique de Montréal, Institut de Génie Nucléaire (December 1999).

[5] E. Varin, A. Hébert, “Linked List Specifications for DONJON,” Report IGE-226, École Polytechnique de Montréal, Institut de Génie Nucléaire (May 2000).

[6] R. Roy, D. Rozon, A. Hébert, G. Hotte,“Treatment of Circular Boundary Conditions in Neutron Diffusion Calculations,” 3rd Int. CNS/ANS Conf. on Simulation Methods in Nucl. Eng., Montréal (April 1990).

[7] M. T. Sissaoui and G. Marleau, “Application of the Feedback Model for the History Base Using DRAGON,” 19th CNS Simulation Symposium, Hamilton (October 1995).

[8] M. T. Sissaoui, G. Marleau, D. Rozon, “Accounting for Actinide Burnup History in CANDU Reactor Simulation,” Advances in Nucl. Fuel Management II, Myrtle Beach (March 1997).

[9] A. Hébert, G. Marleau and R. Roy, “A description of the DRAGON Data Structures - Version DRAGON_000331 Release 3.04,” Report IGE-232Rev3, École Polytechnique de Montréal, Institut de Génie Nucléaire (April 2000).

[10] A. Hébert, “Application of the Hermite Method for Finite Element Reactor Calculations,” *Nucl. Sci. Eng.*, **91**, 34 (1985).

[11] 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).

[12] A. Hébert, “Variational Principles and Convergence Acceleration Strategies for the Neutron Diffusion Equation,” *Nucl. Sci. Eng.*, **91**, 414 (1985).

[13] A. Hébert, “Preconditioning the Power Method for Reactor Calculations” *Nucl. Sci. Eng.*, **94**, 1 (1986).

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

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

[16] A. Hébert, “Application of a Dual Variational Formulation to Finite Element Reactor Calculations,” *Ann. nucl. Energy*, **20**, 823 (1993).

[17] R. Roy, G. Marleau, J. Tajmouati, and D. Rozon, “Modelling of Candu Reactivity Control Devices with the Lattice Code Dragon,” Ann. Nucl. Energy, Vol. 21,pp. 115-132 (1994).

[18] J. H. Wilkinson, “The Algebraic Eigenvalue Problem,” *Clarendon Press*, Oxford (1965).

[19] 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).

[20] J. Koclas, M. T. Sissaoui and A. Hébert, "Solution of The Improved and Generalized Quasistatic Methods using an Analytic Calculation or a Semi-Implicit Scheme to Compute the Precursor Equations, " Ann. Nucl. Energy, Vol. 23, no 11, pp 901-907 (1996).

[21] W. H. Press, S. A. Teukolski, W. T. Vetterling and B. P. Flannery, "Numerical Recipes in FORTRAN," Second Edition, Chapter 16, Cambridge University Press (1992).

[22] M. T. Sissaoui, J. Koclas and A. Hébert, "Solution of The Improved and Generalized Quasistatic Methods by Kaps and Rentrop Integration Scheme with Stepsize Control, " Ann. Nucl. Energy, Vol. 22, no 12, pp 763-774 (1995).

[23] J. Tajmouati, "Optimisation de la Gestion du Combustible Enrichi d'un Réacteur CANDU avec Prise en Compte des Paramètres Locaux", Ph. D. Thesis,École Polytechnique de Montréal, Institut de Génie Énergétique (May 1993).

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

[25] 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).

FIGURES

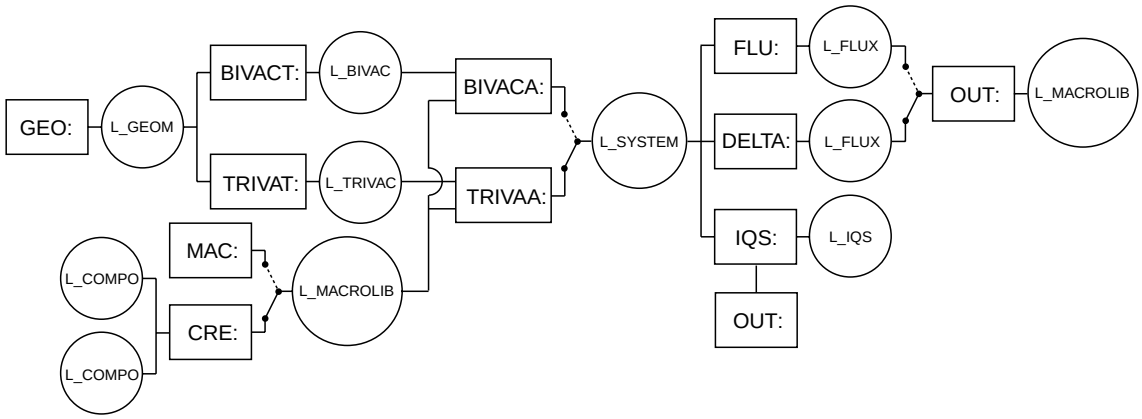

Figure 1: An example of the DONJON modular approach.

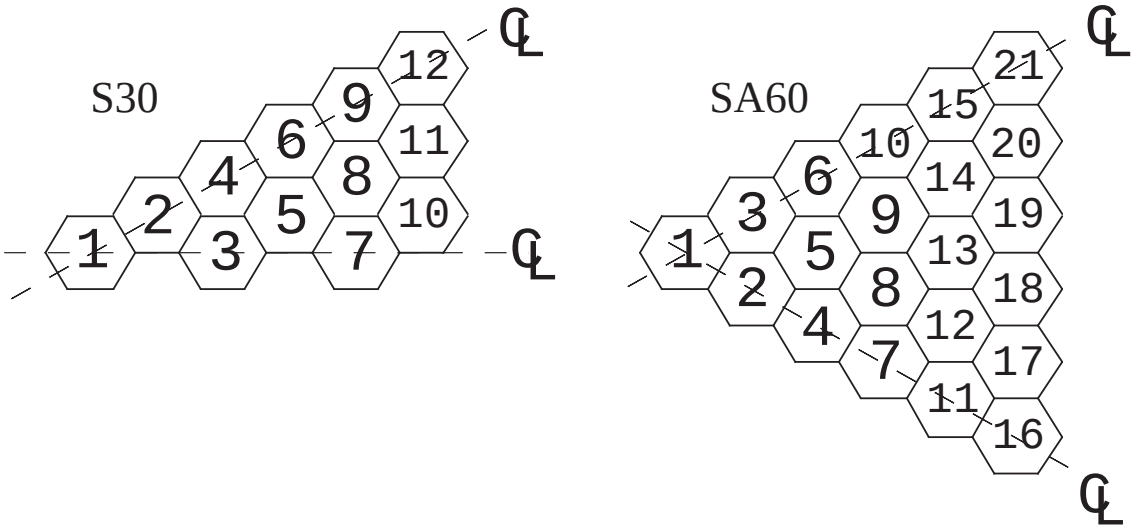

Figure 2: Hexagonal geometries of type S30 and SA60

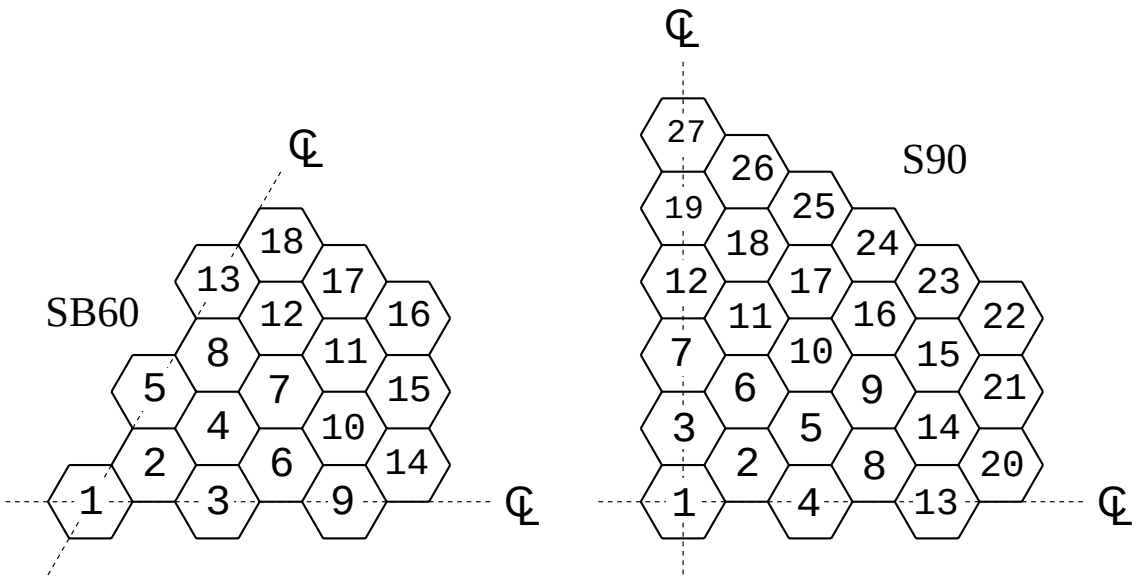

Figure 3: Hexagonal geometries of type SB60 and S90

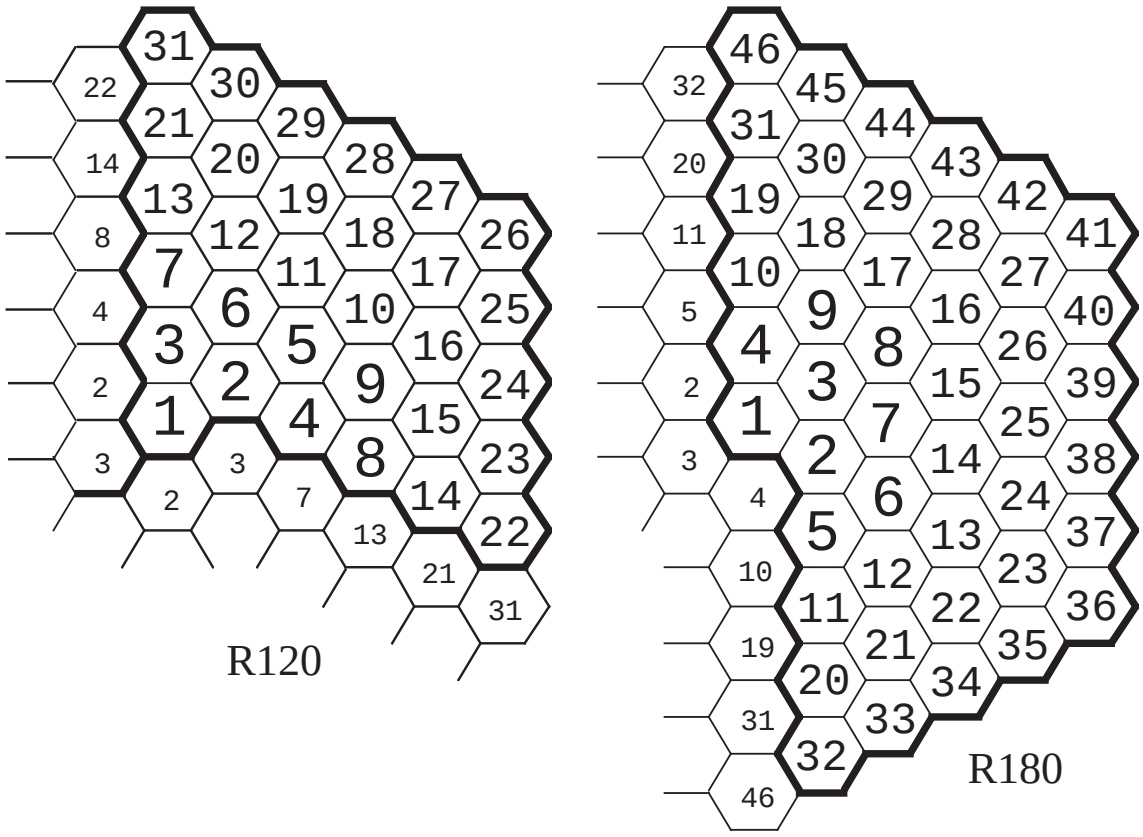

Figure 4: Hexagonal geometries of type R120 and R180

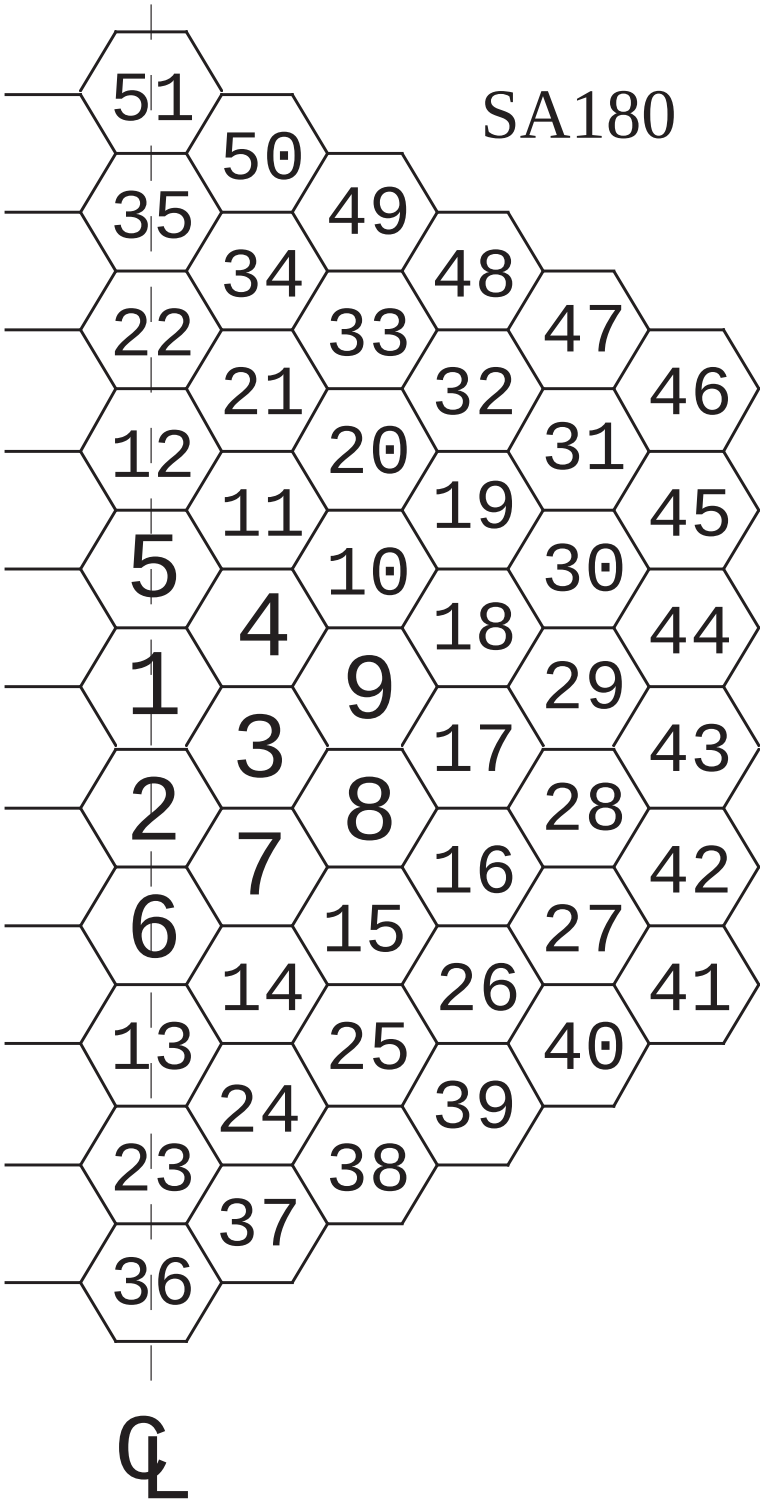

Figure 5: Hexagonal geometry of type SA180

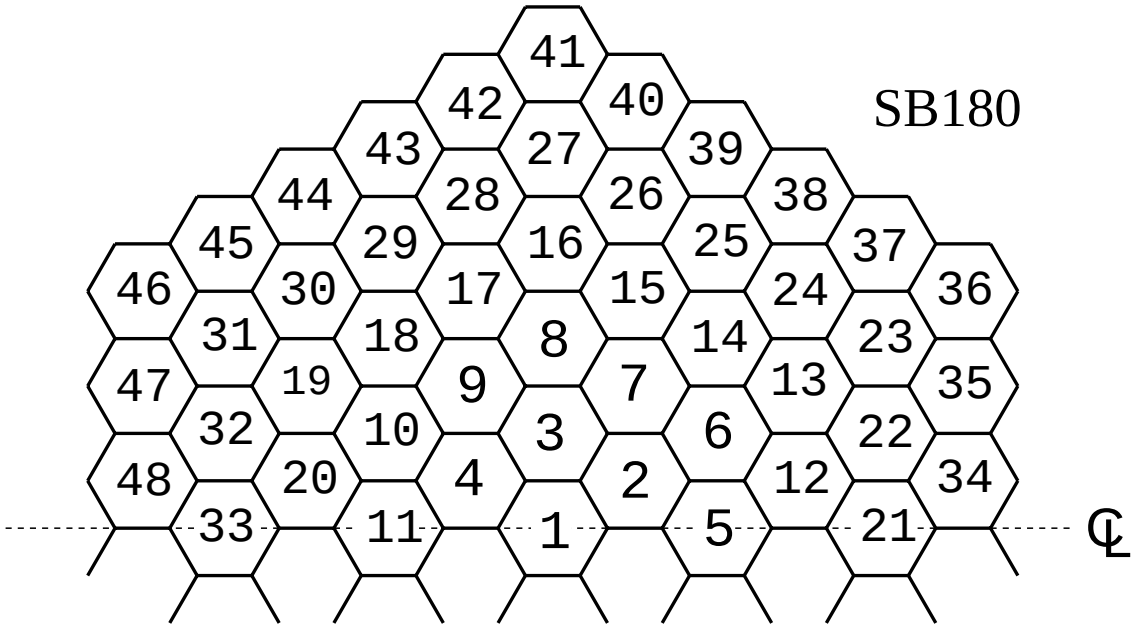

Figure 6: Hexagonal geometry of type SB180

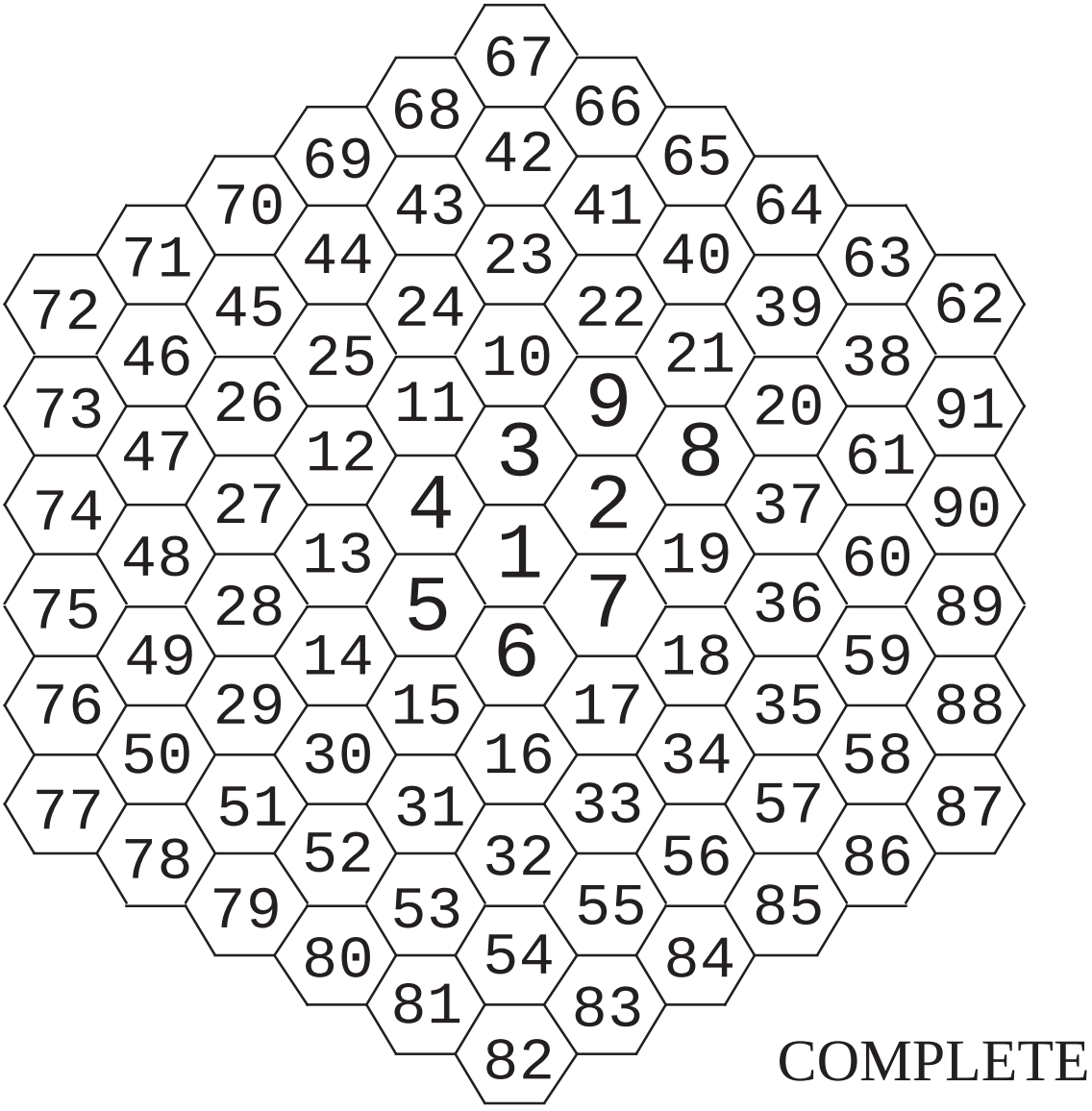

Figure 7: Hexagonal geometry of type COMPLETE

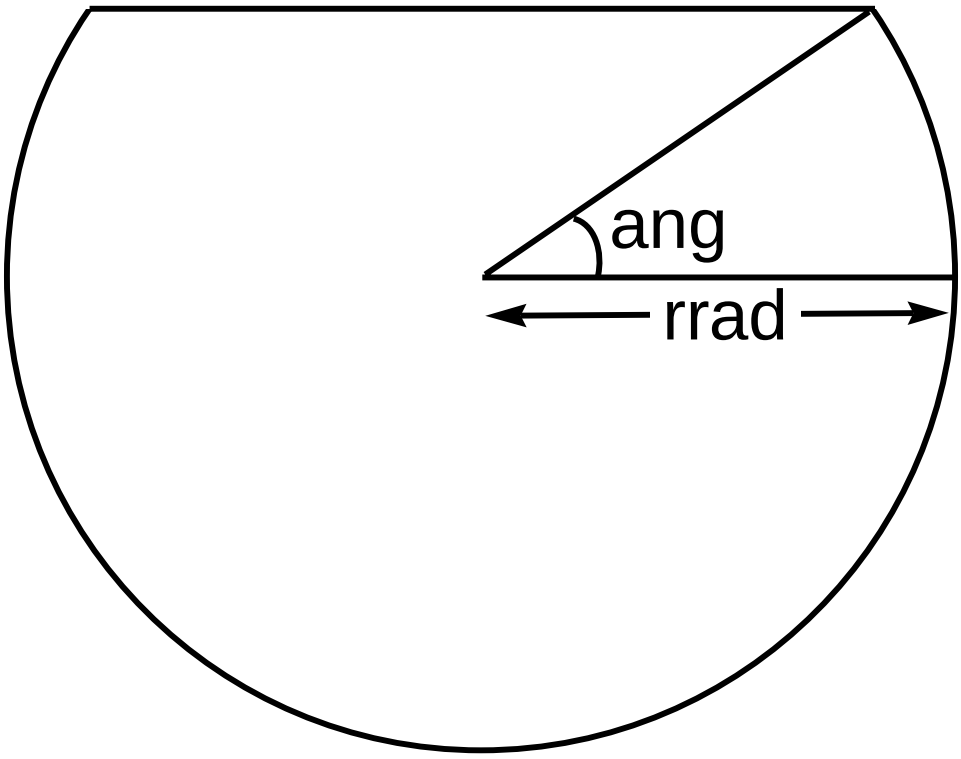

Figure 8: Cylindrical correction in Cartesian geometry

Figure 9: Slab geometry with mesh-splitting

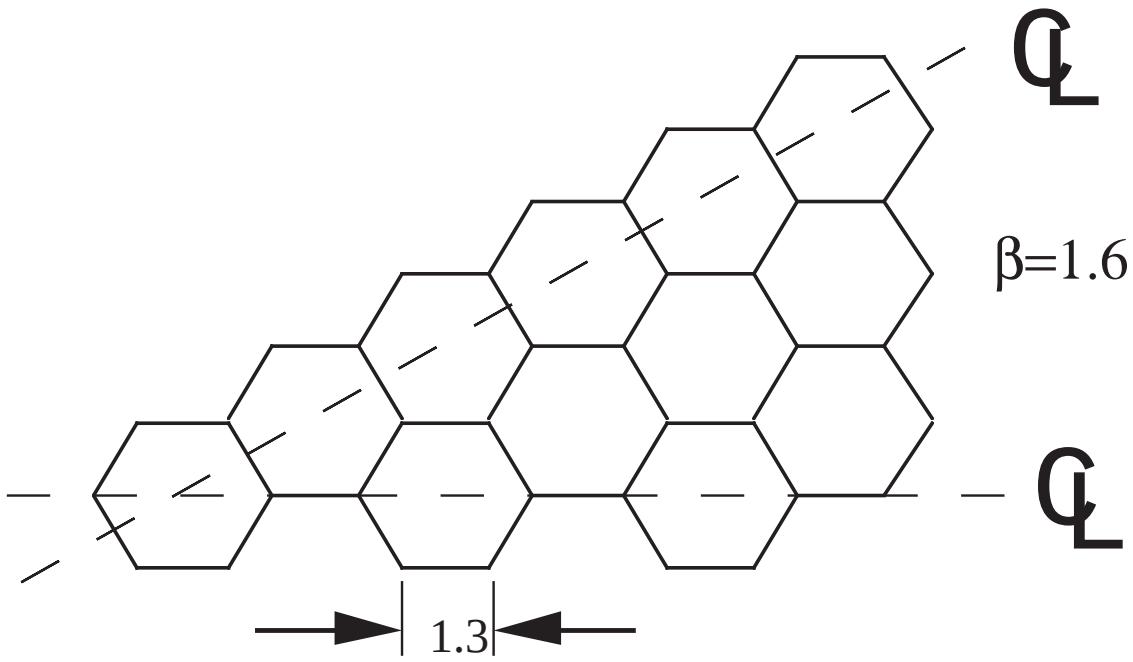

Figure 10: Two-dimensional hexagonal geometry

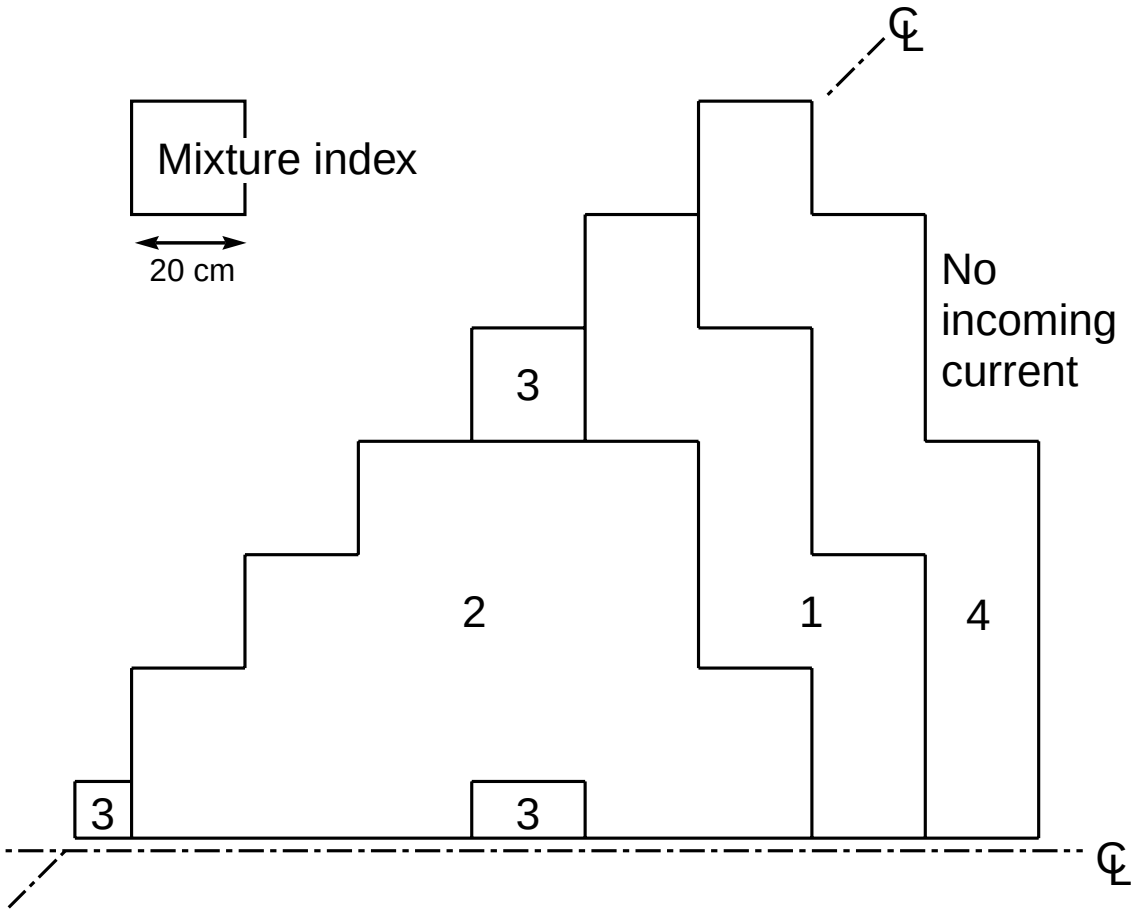

Figure 11: Description of the IAEA-2D benchmark.

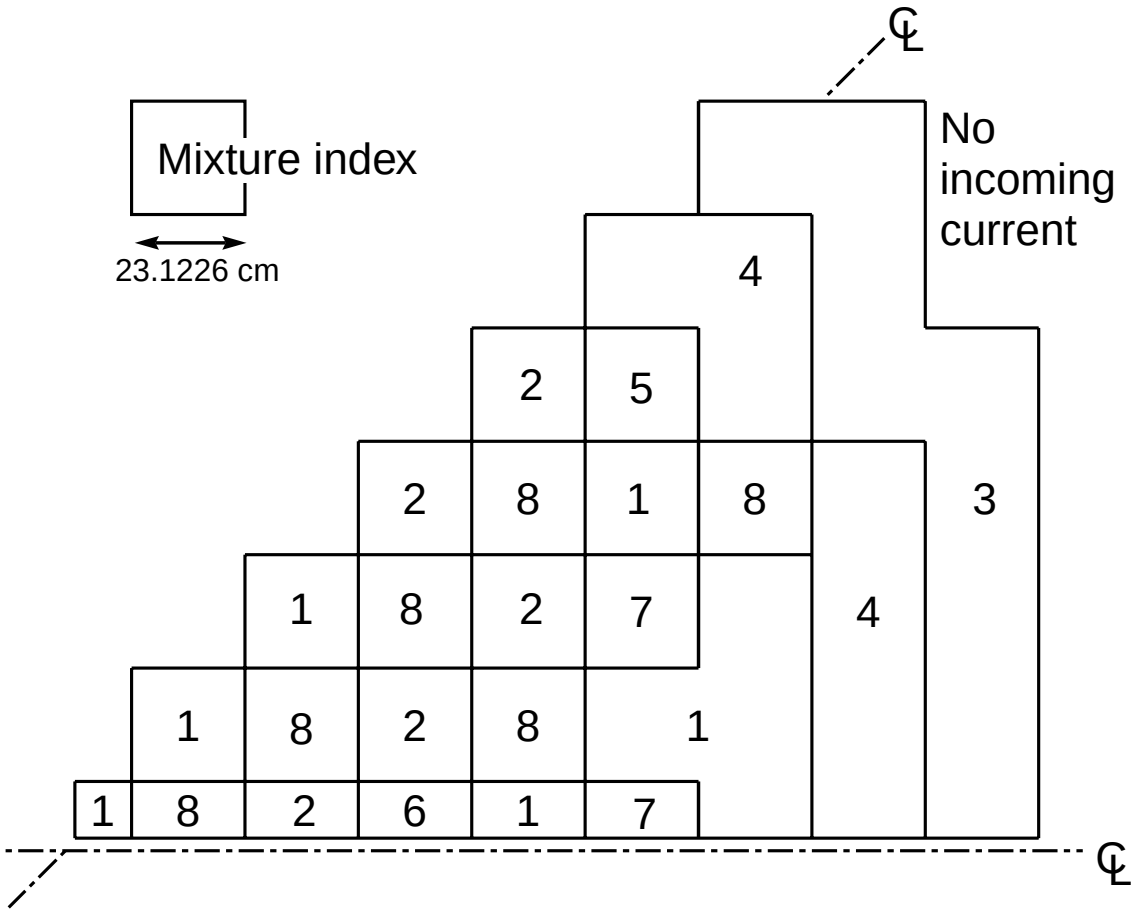

Figure 12: Description of the Biblis-2D benchmark, rods-withdrawn configuration.

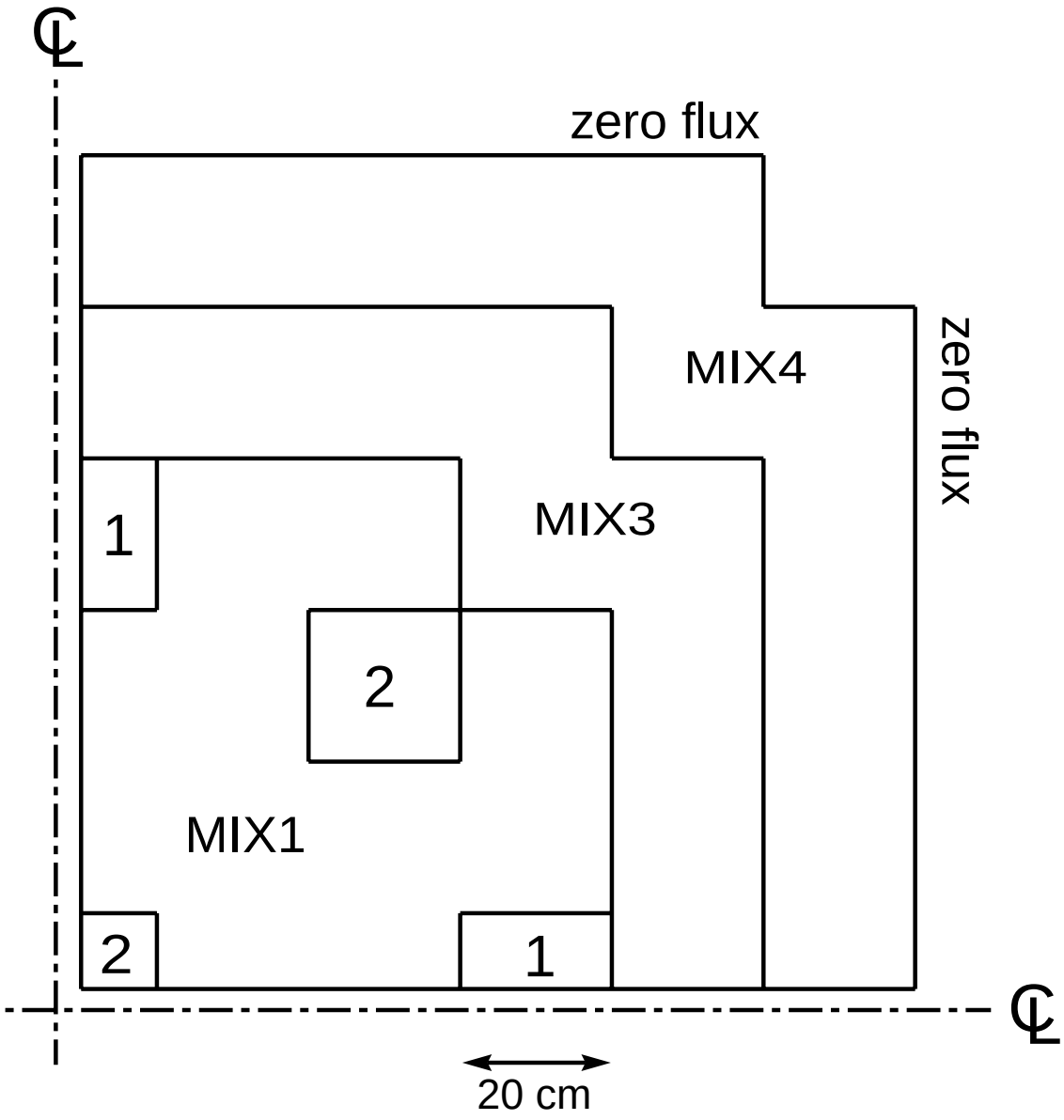

Figure 13: Description of the LMW benchmark in 3-D.

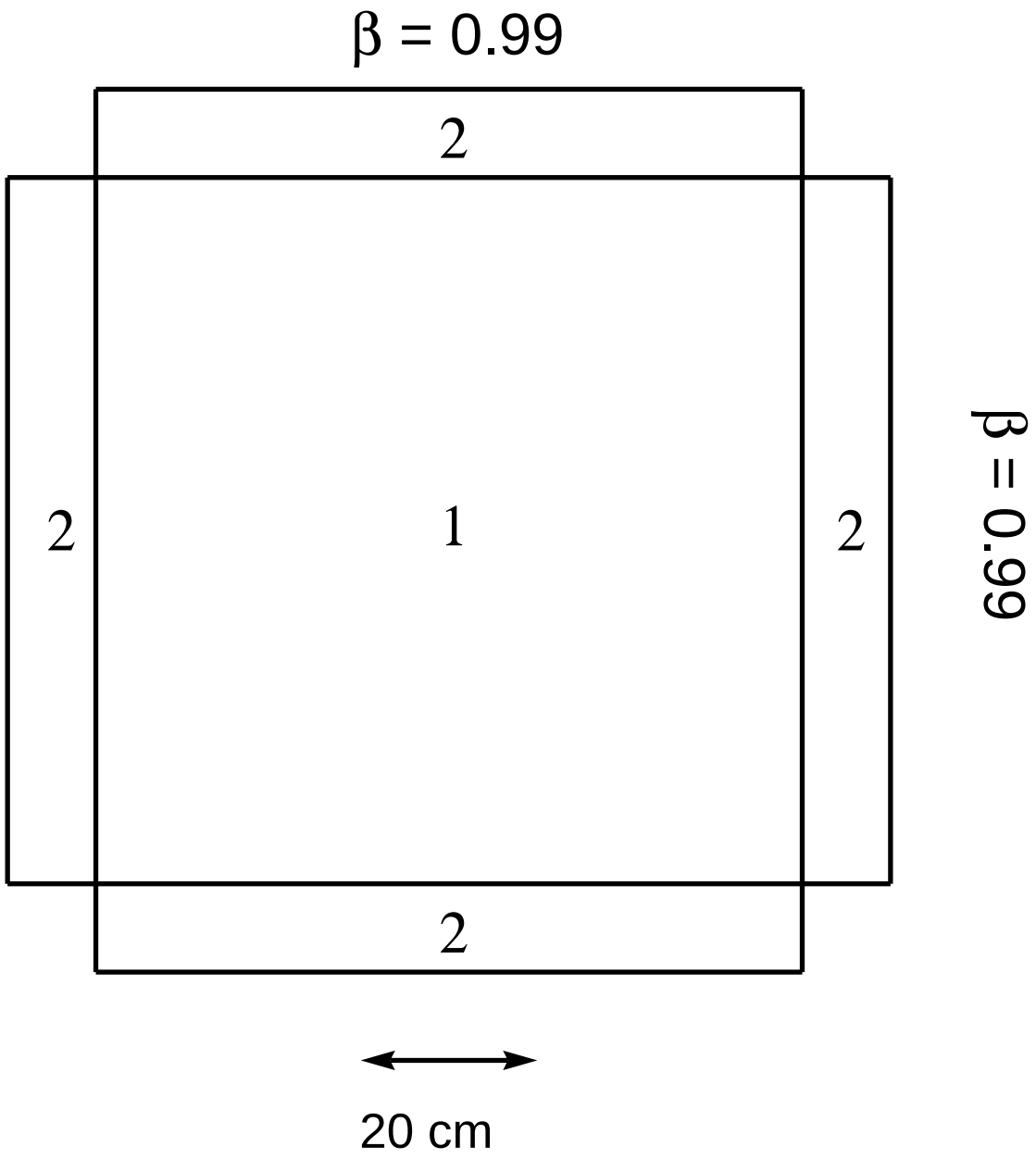

Figure 14: Description of a plane of the input geometry.