Studsvik® *Scandpower*

USER MANUAL
AURORA

11 January 2008

DOCUMENT CONTROL

This document will be updated periodically, whenever revisions are made to the Manual. This front page, together with a page specifying the pages to be deleted and/or added, will be provided with each new addition. The User should check that the Manual corresponds to the Program Version in use and that all revisions are included.

INPUT MANUAL		PROGRAM		REMARKS	Rev'd	Appr'd
Rev. No	Date	Version	Date		by	by
0	18Oct93	AR931A aurora-1.0	11Oct93	Original release		
1	31Aug94	AR941A aurora-1.1	31Aug94	Minor changes		
1	-	aurora-1.2	-	Unchanged		
2	22Dec95	aurora-1.3	15Dec95	General restart option Represent gadolinia by 64000 Expand spectrum options		
3	01Oct96	aurora-1.4	05Jul96	Minor changes		
4	01Aug98	aurora-1.5	01Aug98	Manual converted from Macintosh Minor bugs removed Read Helios work/scratch-area sizes		
5	01Apr00	aurora-1.6	01Apr00	Minor bugs and typos removed Larger input SETs allowed Negative input buckling allowed		
6	31Oct01	aurora-1.7	31Oct01	Minor errors and typos removed New restart with unspecified burnup		
7	22Nov03	aurora-1.8	22Nov03	Minor errors and typos removed Changed resonance categories Zr as resonance absorber Example of OVSM use in TREE operator		
8	26Sep05	aurora-1.9	26Sep05	New RUN option for data base strategy Minor errors and typos removed Remarks on eliminated leakage-averaging		
9	11Jan08	aurora-1.10	11Jan08	Introduction of P_2 and P_3 output Introduction of 66000 and 68000 for input of Dy_2O_3 and Er_2O_3 Zr isotopes get resonance treatment Added rotation angle to CCS in STR Added new angular coupling options and changed existing options Minor errors and typos removed Equations converted from MathType		

REV. 1: 31AUG94

Minor changes

Pages replaced:

Entire document

REV. 2: 22DEC95

Introduce restart/dump option.
Represent gadolinia by identifier 64000.
Expand spectrum options (critical buckling, input buckling, infinite medium).

Pages replaced:

Cover page
Document Control .... pages 1-2
Contents .... pages 1-1
Introduction .... pages 1-5
MAT operator .... pages 1-6
OVLM operator .... pages 1-10
AREA operator .... pages 1-4
RUN operator .... pages 1-11

New pages

DBMAT operator .... pages 1-7
DUMP operator .... pages 1-3

REV. 3: 01OCT96

Minor changes.

Pages replaced:

Cover page
Document Control .... pages 1-2
Contents .... pages 1-1
DBMAT operator .... pages 1-7
STR operator .... pages 1-7
PATH operator .... pages 1-8
DUMP operator .... pages 1-3

REV. 4: 01AUG98

Converted from Macintosh with minor changes.
Sizes of Helios working areas and scratch area in memory can be changed by input.

Pages replaced:
Entire document

REV. 5: 01APR00

Minor bugs have been removed and typing errors corrected.
Larger input sets are allowed.
Negative input buckling is allowed.

Pages replaced:

Cover page
Document Control .... pages 1-3
Contents .... pages 1-1
Introduction .... pages 1-4
STR operator .... pages 1-7
PATH operator .... pages 1-8
RUN operator .... pages 1-11

REV. 6: 31OCT01

Minor errors have been removed and typing errors corrected.
New restart with unspecified burnup allows for branch-off calculations in a later case.

Pages replaced:

Cover page
Document Control .... pages 1-3
Contents .... pages 1-1
Introduction .... pages 1-4
DBMAT operator .... pages 1-10
STR operator .... pages 1-7
OVLM operator .... pages 1-10
OVSM operator .... pages 1-4
TREE operator .... pages 1-4
DUMP operator .... pages 1-3
RUN operator .... pages 1-11

REV. 7: 22NOV03

Minor errors have been removed and typing errors corrected.
Changed resonance categories and Zr as resonance absorber.
Explanation of beginner's wrong use of OVSM in TREE operator.

Pages replaced:

Cover page
Document Control .... pages 1-3
Contents.......................... pages 1-1
MAT operator .... pages 1-6
OVLM operator .... pages 1-10
OVSM operator .... pages 1-4
TREE operator .... pages 1-6
RUN operator .... pages 1-11

REV. 8: 26SEP05

New OM option in RUN operator for data base strategy.
Minor errors have been removed and typing errors corrected.
Remarks on abandoned leakage-averaging in MACRO and MICRO.

Pages replaced:

Cover page
Document Control .... pages 1-4
Contents.......................... pages 1-1
CASE operator .... pages 1-2
DBMAT operator .... pages 1-10
TREE operator .... pages 1-6

MACRO operator	pages 1-4
MICRO operator	pages 1-4
RUN operator	pages 1-11

REV. 9: 09JAN08

Neutron output of P2 and P3 matrices has been introduced.
Minor errors have been removed and typing errors corrected.
Introduction of 66000 and 68000 as identifiers for Dy_2O_3 and Er_2O_3.
Introduction of 40010 as identifier for natural Zr as resonance element (if it is not in the library).
Individual Zr isotopes included as separate resonance category.
Added CCS rotation angle in STR.
Added new angular coupling options and changed existing options, including removing duplicates.
Increased default memory parameters.
Minor errors have been removed and typing errors corrected.
Converted all equations from MathType for compatibility.

<u>Pages replaced:</u>

Cover page

Document Control	pages 1-4
Contents.........................	pages 1-1
Introduction.....................	pages 1-4
AREA operator	pages 1-4
CASE operator	pages 1-2
DBMAT operator	pages 1-10
ALB operator..................	pages 1-3
CUR operator.................	pages 1-2
DUMP operator	pages 1-3
FACE operator	pages 1-4
GROUP operator............	pages 1-2
MACRO operator	pages 1-5
MAT operator	pages 1-6
MICRO operator	pages 1-4
ISOX operator	pages 1-2
NEWK operator	pages 1-3
OVLDT operator.............	pages 1-5
OVLM operator...............	pages 1-10
BDRY operator...............	pages 1-5
OVSDT operator	pages 1-3
OVSM operator	pages 1-4
PAR operator..................	pages 1-11
PATH operator	pages 1-8
SET operator..................	pages 1-4
STAT operator................	pages 1-2
TREE operator	pages 1-6
CCS operator.................	pages 1-3
STR operator..................	pages 1-7
RUN operator	pages 1-11
CNX operator.................	pages 1-7

NOTE

In this loose-leaf manual, each odd page is identified in the header by

AURORA Input: "Chapter name"   "page nr"

while its footer is

"File name": "Date"   *Studsvik Scandpower*

Each even page is identified in the header by

"page nr"   AURORA Input: "Chapter name"

while its footer is

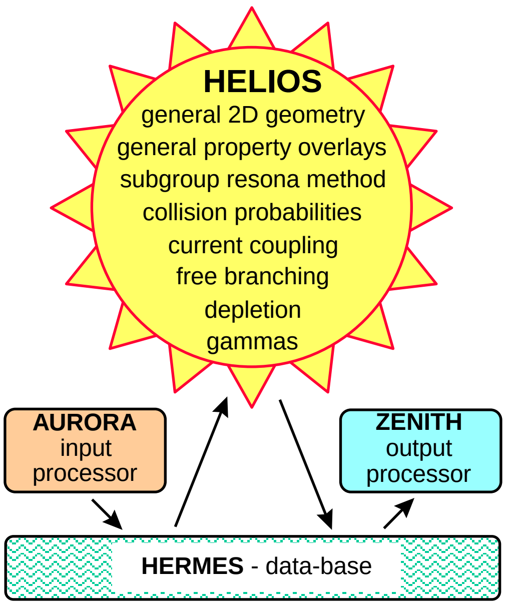
Studsvik Scandpower   *"File name": "Date"*

All the pages of a given chapter of this manual have the same date—the date when that chapter was updated or created. So the dates of the individual chapters do not necessarily agree with the date of issuance of the actual revision of this manual. Cover page, Document control, Note and Contents, however, always have the date of issuance of the actual revision of this manual as their date.

TABLE OF CONTENTS

INTRODUCTION

HELIOS is a neutron and gamma transport code for lattice burnup, in general two-dimensional geometry. Its input and output processors are the separate codes AURORA and ZENITH. The data flow between the three codes is via a data-base that is accessed and maintained by a subroutine package called HERMES. Fig. 1 shows this interrelationship.



The User's input to HELIOS is described in this manual. AURORA reads, processes and saves this input. The result is a number of "User arrays" that start with the letter "U". These arrays, information on their sizes, and the User's input, are written into a data-base for retrieval by HELIOS and/or ZENITH.

Another AURORA application is the management of an input data-base, as described in the section on the SET operator. Such a data-base contains sets of recurrent parts of the HELIOS and/or ZENITH input of a certain fuel-assembly design. These sets can be directly included in the User's input, thus reducing the work spent on input preparation.

Notation

Throughout this manual, a pair of brackets, "[...]", encloses one or more optional arguments and their delimiters. The ellipsis, "...", indicates that the contents of the preceding pair of brackets may be repeated any number of times.

General input rules

1) The input must be provided as an ASCII file, whose lines must not be more than 80 characters long.

2) The input is based on names, operators and their arguments. The following types of syntax are possible ("^" represents a delimiter):

   name = OPERATOR (argument [^ argument] ...)
   $*name* = PAR (argument [^ argument] ...)
   &*name* = SET (argument [^ argument] ...)

   In the application of the management of a SET data-base, ZENITH preprocessor instructions—starting on a "%"—can also be given as input; see the ZENITH manual.

3) The first nonblank character of the left-hand-side name of an operator must also be the first nonblank character on that line. However, it may be preceded by any number of comments provided that they begin and end on the same line (see Comment rules).

4) There are three types of names:
   - HELIOS names of operator types, such as CCS, CNX and CUR.
   - User names defined as the left-hand sides of an operator; they must be unique and represent the arguments of that operator.
   - User names assigned as identifiers to regions inside a CCS or STR operator; they do not have to be unique.

5) A <u>name</u> is a sequence of up to 80 arbitrary characters (except apostrophes) that must be enclosed in apostrophes, unless it starts with a letter and contains only alphanumeric characters.

6) A <u>parameter</u> name, $*name*, can only be defined by the PAR operator. It must start on a "$" sign that is part of the name, although it must be outside the possible apostrophes that enclose *name*. So, $'a $!' represents the five-character parameter name <u>$a $!</u>.

7) A <u>set</u> name, &*name*, can only be defined by the SET operator. It must start with the "&" sign that is part of the name, although it must be outside the possible apostrophes that enclose *name*. So, &'a $!' represents the five-character set name <u>&a $!</u>.

8) <u>Arguments</u> must be names—including parameter names, but excluding set names—numerical constants (see below), or arithmetic expressions which must be enclosed in quotes (see PAR operator).

9) <u>Blanks</u> inside and outside the operators are ignored—except blanks inside names in apostrophes, which are part of those names.

10) Each operator has a local syntax that is described in this manual. This syntax determines the types of arguments and their separating delimiters.

11) About half of the operators are mandatory; the others are optional. The mandatory operators are:

CASE	MAT	STR	CNX	BDRY	OVLM
OVSM	OVLD	OVSD	OVLT	OVST	STAT
PATH	RUN				

The optional operators are:

DBMAT	ALB	CCS	NEWK	ISOX	TREE
GROUP	AREA	FACE	MACRO	MICRO	CUR
DUMP	SET	PAR			

12) Each case must have one CASE and one RUN operator. They must be the first and last operator of that case and must have the same User name. There may be any number of operators of the other types.

13) The sequence of the operators is immaterial, with two exceptions:
    - The mutual sequence of the CNX and NEWK operators.
    - Per type ∗ (where ∗ is M, D or T), the mutual sequence of the overlay and overlay-set operators, OVL∗ and OVS∗.

Rules for numerical constants

1) Embedded blanks in numerical constants are ignored.

2) A numerical constant must start with a numerical character, a minus sign (if negative), an optional plus sign, or the decimal point. The optional plus sign is not always allowed—it is expressly forbidden in arithmetic expressions.

3) Apart from a possible initial sign, an <u>integer</u> constant must contain only numerical characters. It ends with the last numerical character before the first non-blank character, which must be a delimiter or the letter "E".

4) A <u>real</u> constant may be represented by one of the following:
    - An integer constant.
    - A fixed-point number.
    - A floating-point number, i.e. scientific notation (E-format).

5) Apart from a possible initial sign, a <u>fixed-point</u> number must contain only numerical characters and one decimal point. It ends with the last character (numerical or decimal point) before the first non-blank character, which must be a delimiter or the letter "E".

6) A <u>floating-point</u> number must consist of a basic part and an exponentiation factor, separated by the letter "E". The basic part must be an integer or a fixed point number; the exponentiation factor must be an integer.

   Observe that a floating-point number must have at least one numerical character before the "E".

Comment rules

1) A comment is any number of arbitrary characters—except exclamation marks—enclosed in exclamation marks.

2) Apart from the restriction mentioned in General input rule 3, comments may appear anywhere, except inside an alphanumeric name without enclosing apostrophes. They do not affect the input.

-	25,abc,!comment!/3	is equivalent with <u>25,abc,/3</u>;
-	25!comment!37	represents the number <u>2537</u>.
-	'na!comment!me'	represents the name <u>na!comment!me</u>;
-	na!comment!me	is not allowed.

Input processing

AURORA processes the input in four phases:

- It reads the User's input for all the cases, writes it to an output file, saves it in a data-base, and performs a global syntax test.
- It reads the User's input again, expands it with the included sets (see SET operator), writes this expanded input to the output file, and saves it in the data-base.
- It reads the expanded input, purges all blanks and comments, resolves parameters and arithmetics (see PAR operator), and saves this final input in the data-base.
- It reads the final input per case, interprets this input per operator, and writes the resulting "U" arrays into the data-base. This phase includes extensive testing—in all, there are more than 280 error messages.

How to run AURORA and HELIOS

1) The entire User's input must be preceded by one line that starts on:

   [+][T]*mode*

+	indicates that data in the data-base of an earlier case or set of the same name (see CASE and SET operators) will be overwritten.
T	generates—if *mode* is HEL—test output for all the cases, i.e. whatever goes to the data-base, and the arithmetic formulae with numerical constants instead of parameters.
mode	are the three letters HEL or SET, which indicate that the input is HELIOS or SET input (see SET operator).

2) The command line to execute AURORA has at most three parameters:

   aurora-version input-file-name [MMRY] [CMMRY]

   The first parameter, the name of the input file is mandatory. The other two allow run time definition of the sizes (in words) of the integer/real and the character working areas. Their defaults values are 1,000,000 and 2,000,000.

3) The command line to execute HELIOS has one (mandatory) parameter:

   helios-version hermes-database

   where the mandatory parameter is the name of the HERMES database. The OM option in the RUN operator allows changing the working-area sizes.

AURORA Input

The correct sequence of the enclosed *.pdf files is

:

ALB(), THE ALBEDO OPERATOR

Purpose

The ALB operator is used to define the matrix elements of neutron and gamma albedos. The BDRY operator may assign these albedos to some or all of the border segments of the system.

Usage

name = **ALB (** *t* **/** *a* [, *a*]… [; *a*, *a* …]… **[/** *a* [, *a*]… [; *a*, *a* …]…**])**

name	represents the albedo matrix specified here.	
t	specifies the type, $t = 1$:	diagonal, equal elements;
	$t = 2$:	diagonal, unequal elements;
	$t = 3$:	full albedo matrix.
a	specifies an albedo-matrix element for neutrons between the first two slashes, and for gammas between the second slash and the closing parenthesis of the operator.	

Rules

1) If the cross-section library contains gamma data, each ALB operator must contain data for neutrons and gammas, even if the actual case will not have gamma calculations. However, gamma albedos must not be specified if the library does not contain gamma data.
2) Wherever an albedo boundary condition is used, the albedo type *t* will be the same for neutrons and gammas.
3) For type $t=1$, one value of *a* must be specified for neutrons (and one for gammas).
4) For type $t=2$, *NOG* values of *a* must be specified for neutrons (and *NOGG* for gammas). These are the diagonal elements $a_{g \leftarrow g}$ in the neutron (and gamma) groups of the library. Their sequence must be $g=1,\ldots NOG(G)$, where the first group has the highest energy.
5) For type $t=3$, *NOG*NOG* values of *a* must be specified for neutrons (and *NOGG*NOGG* for gammas). These are the elements $a_{h \leftarrow g}$, given in the sequence (($g=1,\ldots NOG(G)$), $h=1,\ldots NOG(G)$), i.e. per column *h*.
6) For type $t=2$, commas must be used as delimiters between the diagonal elements.
7) For type $t=3$, commas must be used as delimiters between the elements in one column *h*, while the semicolon must be the delimiter between the col-

umns.

8) The condition of specular (mirror) reflection can only be specified in the BDRY operator; hence, it requires no ALB operator.

9) Albedo matrices that are not used may also be specified.

Remarks

The ALB operator specifies albedo matrices, which may or may not be used by the BDRY operator to assign reflective properties to border segments of the system. The albedo-matrix element $a_{h \leftarrow g}$ is the probability for a neutron (gamma) exiting through a segment in group g, with a neutron (gamma) returning through that segment in group *h*.

Although the albedo-matrix elements have the character of probabilities, no acceptance tests of the albedo data are made, except that their number agrees with the number of neutron (gamma) groups. In other words, the following tests are *not* made:

$$0 \le a_{h \leftarrow g} \le 1 \quad \text{and} \quad \sum_h a_{h \leftarrow g} \le 1$$

It is up to the User to decide if the albedo-matrix properties are physical or not. For instance, if per exiting particle more than one particle returns, the border is multiplicative. In certain circumstances, this could be physically correct. The significance of negative elements is harder to envisage; yet they are allowed, too. The User should be aware that straying too far from the above conditions may cause HELIOS to diverge, in particular during the B_1 calculation of the criticality spectrum (see *HELIOS Methods*, Chapter VI).

Angular dependence is not included in the HELIOS albedos. If the BDRY operator specifies a certain angular discretization k of the border currents, only the diagonal (in-group) reflection is "specular", i.e. particles enter through an angular sector that is specular with respect to the exiting sector. Off-diagonal reflection (to other groups) is isotropic, i.e., each sector gets an equal fraction of the reflected border current.

While the identity matrix may be specified, e.g. *alb* = ALB(1/1/1), it cannot be combined with exact current coupling, $k = 0$, to assign specular coupling in the BDRY operator; see the BDRY operator description.

The distinction between the three matrix types has been introduced to facilitate the input of the diagonal type $t = 1$, which is likely to be the most common. If the neutron and gamma albedos are of different types, the User may always upgrade type $t = 1$ to $t = 2$ or $t = 3$, and type $t = 2$ to $t = 3$.

Examples

1a) `total = ALB ( 1 / 1 / 1 )`

1b) `total = ALB ( 1 / 1 )`

```
1c)  total = ALB ( 2 / 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 )

2)   black = ALB ( 1 / 0 / 0 )

3)   grey  = ALB ( 2 / 0.5, 0.6, 0.7, 0.8, 0.9,   0.9, 0.9, 0.9, 0.1, 0.1,
                       0.3, 0.6, 0.9, 1.2, 1.5,   0.3, 0.3, 0.3, 0.3, 0.3,
                       0.5, 0.6, 0.7, 0.8, 0.9
                     / 0.9, 0.9, 0.9, 0.9, 0.9,   0.5, 0.6, 0.7, 0.8, 0.9,
                       0.5, 0.6, 0.7, 0.8, 0.9 )

4)   full  = ALB ( 3 / 0.5, 0.6, 0.7, 0.8, 0.9,   0.9, 0.9, 0.9, 0.1, 0.1;
                       .99, 0.6, 0.9, 1.0, 1.5,   .99, 0.3, .99, 0.3, 0.3;
                       0.3, 0.6, 0.9, 1.0, 0.9,   .99, 0.3, .99, .99, 0.3;
                        99, 0.6, 0.9, 1.0, 0.9,   0.3, 0.3, 0.3, .99, .99;
                        99, 0.6, 0.9, 1.0, 1.5,   0.3, 0.3, .99, .99, 0.3;
                       .99, 0.6, 0.9, 1.0, 0.9,   0.3, .99, 0.3, .99, .99;
                       0.3, 0.6, 0.9, 1.0, 1.5,   0.3, .99, .99, 0.3, 0.3;
                       .99, 0.6, 0.9, 1.0, 0.9,   .99, .99, .99, .99, .99;
                       0.3, 0.6, 0.9, 1.0, 0.9,   .99, .99, 0.3, 0.3, .99;
                       0.5, 0.6, 0.7, 0.8, 0.9,   0.9, 0.9, 0.9, 0.9, 0.9;
                       0.5, 0.6, 0.7, 0.8, 0.9,   0.5, 0.6, 0.7, 0.8, 0.9 )
```

The first example defines a diagonal albedo matrix with $a_{g \leftarrow g} = 1$ for all g. In the first variant the library includes gamma data; in the second and third variants, there are no gamma data in the library. The second variant is preferred to the third, as it does not require knowledge of the number of neutron groups. All three variants describe total reflection; its angular dependence will be specified in the BDRY operator as discussed above under "Remarks".

In the second example the albedo describes vacuum boundary conditions both for neutrons and gammas.

In the third example the reflection is still diagonal, i.e. no group-to-group coupling, but the diagonal elements vary from group to group. It is seen that there are 25 neutron groups and 15 gamma groups in the library.

The fourth example shows an example of a full matrix for use with a library with 10 neutron groups but without gamma data. It is obvious that the input of a full matrix is very unwieldy. There is no remedy for this situation, which can be compared to describing a region by its cross-section data in all groups.

AREA(), THE EDIT-REGION OPERATOR

Purpose

The AREA operator can be used to define one or more edit regions as one region combination or as a set of regions and/or region combinations. For the defined edit regions, the output specified by the MACRO and MICRO operators will be evaluated and saved in the data base.

Usage

name = **AREA (** [<]*aid* **[**, *aid***]** [>] … **)** [<…> are unnested pair(s)]

name	represents an area, i.e. a single combination of one or more regions, or a set of regions and/or region combinations.
aid	is the identifier of one or more regions of an area:

$aid =$ **[** $[<]ss -$ **[** $[<]ss -$ **]** $\ldots$ **]** $[<]st - [<]cc - [<]rg[>]$

which denotes one or more regions (*rg*) as the last link of a chain that passes through the subsystems (*ss*), if any, the structures (*st*) and the CCS-es (*cc*; for non-CCS regions $cc = 0$).

Variants:		
	$x = (x[-x]\ [,x[-x]]\ \ldots)$	$x = st$, cc or rg;
	$ss - [ss -] \ldots = *$	all subsystems;
	$cc - rg = **$	all regions of *st*;
	$x = *$	all *x* of preceding link.

<…> around one or more *aid*'s or inside individual *aid*'s causes the enclosed regions to be combined into one edit region.

Rules for *aid*'s before combination

(OVLM operator, Rules 2-14 for direct referencing)

1) If the regions specified by one or more *aid*'s are not combined, then the area (*name*) represents as many edit regions as the number of specified regions. Output for *name* will be saved in the data base for each of the regions.
2) The identifier (*aid*) must specify at least one region by a chain whose links are connected by hyphens, i.e. apart from <…>,

   $aid = [ss - [ss -] \ldots]\, st - cc - rg.$
3) The links represent the successive geometric components that contain the region(s). For each region, the chain passes through zero, one or more subsystems (*ss*), one structure (*st*), and one CCS (*cc*), to the region (*rg*). For non-CCS regions, $cc = 0$.
4) Each link must represent one or more components. If each link has only one component, *aid* defines one region; if not, it defines more regions.

5) A component is specified by referring to its representative entry in an operator that defines the component of the preceding link. The first link must refer to a component of the system itself.

6) The *ss* and *st* links must refer to entries in the *s* lists of the CNX operators that define the subsystems of the preceding *ss* links.

7) The *cc* link must refer to one or more CCS-es (*cname*) in the STR operator(s) that define the structure(s) of the preceding link.

8) The *rg* link must refer to one or more CCS regions (*cid*) in the CCS operator(s) that define the CCS(-es) of the *cc* link, or to one or more non-CCS regions in the STR operator(s) that define the structure(s) of the *st* link (their order is: first *bid*, then the *rid*'s, if any).

9) A component may be specified by giving the sequence <u>number</u> (>0) of its representative entry in the preceding operator. Alternatively, that number may be preceded by a colon and the component name. For partial CCS-es, observe CCS Rule 5 of the STR operator.

10) One or more components may be specified by their common <u>name</u>.

11) All the components of a certain type may be specified by an <u>asterisk</u>.
    (i) A double asterisk ($**$) denotes all the regions of a *st*.
    (ii) An asterisk denoting all the *ss*'s (there must be at least one *ss*) is the first link of all chains, even those without a *ss*.
    (iii) Thus, all the regions of a system with at least one *ss* in its CNX operator are given by $*-*-**$, otherwise by $*-**$.
    (iv) The construction $*-st$ number$-$ … must not be used.

12) Specifications by number and/or name can be combined as follows:

    $x = (x[-x]\ [,x[-x]]\ldots)$ with $x = st$, cc or rg.

    Each x inside the parentheses is a component number or name; a range of one or more numbers may be given by $x_i - x_j$ with $x_i \le x_j$.

13) Rule 12 defines multiple *aid*'s, each being a specification by number or name. If this construction occurs in more than one link, the *aid*'s are sequenced such that *rg* varies fastest and *st* slowest.

14) For dumps (see DUMP operator) the *ss*, *st* and *cc* components must be sequence numbers; they may be combined according to Rule 12.

Rules for combination

1) Provided they are unique, one or more of the regions specified by the *aid*'s may be combined into one edit region by enclosing them in <…>. For such an edit region, the output is first combined and then saved in the data base.

2) A region combination does not have to be contiguous, e.g. the combined fuel of all the pins of an assembly can be an edit region.

3) Each *name* may have none, one or more combinations. However, each <…> must either act on one or more *aid*'s or inside an *aid*, i.e. <…> cannot be more than one level deep.
4) The closing '>' of a combination must always come at the end of the *aid* inside—or the last *aid* on—which it acts, i.e. after the *rg*.
5) Owing to Rule 11 ii), an opening '<' before or after an asterisk that denotes all the *ss*'s acts in both cases on all the defined structures.
6) The output for a region combination is homogenized or summed. Averaged quantities (flux-averaged cross sections, data per cm^3 or per gramme) are homogenized; absolute quantities (weight or volume) are summed.
7) Because of Rules 1 for *aid*'s and for combination, an area with one or more combinations has fewer edit regions than the number of specified regions. If all the *aid*'s are enclosed by a single <…>, there is only one edit region, i.e. one region combination.

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data base (from where they can be retrieved by an output processor). With the AREA operator various areas can be defined. Their edit regions are homogenized regions and sets of regions and/or homogenized regions. Normally, output for these edit regions involves considerably fewer data than output for all the regions of the system.

Fuel-assembly homogenization is an example where the area is a single edit region made up of a region combination. Instead of saving certain cross sections per region they are saved per assembly, reducing the output considerably. Of course, these cross sections cannot be expanded back into data per region.

A map per fuel pin is a typical application where the area represents a set of edit regions, the fuel regions of the pins. If a pin has more than one fuel region, they must be combined (homogenized) by enclosing them in <…>; see also Examples.

The volumes of the (homogenized) edit regions of an area are always saved in the data base, while the fluxes are saved for each MACRO or MICRO operator in which the area occurs. If that area occurs in a MICRO operator, the relevant isotopic number densities in its edit regions are saved too.

Combining CCS regions with non-CCS regions is only possible if the involved regions have a unique name, as in Fig.1. The CCS *pin* contains fuel (*f*), clad (*cl*) and a coolant shell subdivided into the regions *a*, *b*, *c* and *d*. The sequence of the CCS regions is *f*, *cl*, *b*, *d*, *c* and *a* (CCS operator, Rules 3 and 5). The non-CCS regions have the same names as their CCS-counterparts; let their sequence be *a*, *b*, *c* and *d*. Assume that a data map is required for the combined regions *c* per *cell*. In Fig.1, the four edit regions of the area that represent the map are shaded. Observe that the system is directly made up of the four structures *cell*, i.e. there are no subsystems. The map *SWcool* can be defined as:

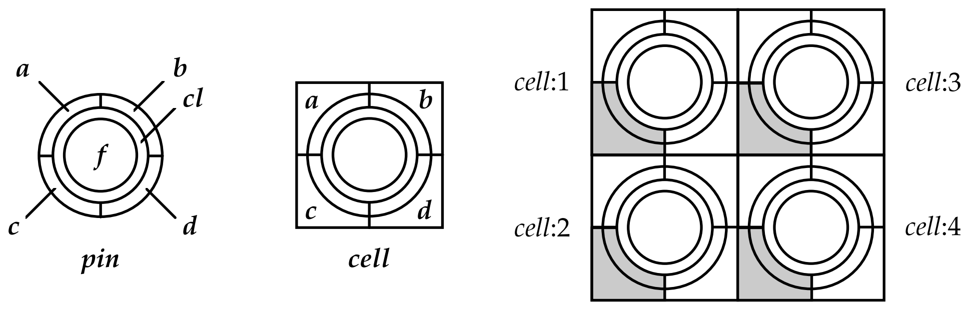


Fig. 1: Combining CCS and non-CCS regions.

```
SWcool = AREA( *-<(0,pin)-c> )
```

where the asterisk may be replaced by *cell* or (1-4) and *pin* by 1.

The following constructions are wrong:

```
SWcool = AREA( *-<(0,*)-c> )         ! violates Rule 12   !
SWcool = AREA( *-<0-c>, *-<*-c> )    ! eight edit regions !
SWcool = AREA( *-<0-3>, *-<*-5> )    ! eight edit regions !
SWcool = AREA(<*-<0-c>, *-<*-c>>)    ! multilevel <…>.    !
```

Observe that, as opposed to the *pid*'s of the overlay operators, there is no indirect referencing of the *aid*'s.

Examples

```
1) Allregs  = AREA(*-*-**)
2) Assembly = AREA(<*-*-**>)
3) Fuelmap  = AREA(*-*-*-<fuel>)
4) Allfuel  = AREA(<*-*-*-fuel, *-*-*-clad>)
```

The first example represents all the regions of the system. The area has as many edit regions as there are regions in the system—each region of the system is an edit region. The second example represents the entire system as a single edit region; it could be used for assembly-averaged output. The third example is an area whose edit regions are the CCS-es with one or more regions called *fuel*. The fourth example defines a single edit region that is made up of all the CCS-regions called *fuel* and *clad* combined; this is a non-coherent edit region.

BDRY(), THE BOUNDARY-CONDITION OPERATOR

Purpose

The BDRY operator is used to set the boundary conditions for a specific subsystem, thus defining the geometry of the system to be calculated by HELIOS.

Usage

name = **BDRY (** (*c*, *n*[) (*c*], *n*) *k* (*b*) **[/** (*c*, *n*[) (*c*], *n*) *k* (*b*) **]**… **)**

name	represents the geometric system, defined by a CNX operator.	
c	specifies a structure in *name* by its chain,	*c* = [*l*–]...*l*
n	specifies a node in that structure,	$n = n_i$
k	specifies an angular representation of border currents.	
b	specifies the boundary condition,	*b* = *aname*
	or	*b* = *c*, *n*[) (*c*], *n*

where:

l	is a link along the chain that defines the structure—the last link—whose nodes *n* are used to define the angular representation of the border currents;
n_i	is the sequence number of a User-specified node in that structure;
aname	is the name of a User-specified albedo (matrix), or the number 0 for specular reflection.

Rules for boundaries

1) The periphery of the system consists of the uncoupled peripheral segments of its structures. They form one or more closed surfaces, the boundaries. One is the external boundary, the others—if any—are the internal boundaries that describe holes in the system.
2) While more than one BDRY operator may be used, every boundary segment must get one and only one boundary condition.
3) A boundary condition must be assigned to one boundary section at a time. Each assignment is preceded by a slash.
4) A boundary section consists of one or more coherent segments and is defined by its begin and end point. When tracing such a section from begin to end point, the system must lie on the right.
5) If begin and end point are equal, an entire boundary is specified.
6) The node of a begin or end point is defined by the location *c* of its structure, and by its location *n* in that structure, enclosed in parentheses, i.e., (*c*, *n*).

7) The location of the structure is specified by a chain *c* whose links *l* are connected by hyphens. The links represent all the successive subsystems that contain the structure.
8) Each link must refer to an entry in the *s*-list of the CNX operator that defines the preceding link. The first link must refer to an entry in the *s*-list of the system itself. Only the last link can and must be a structure.
9) Each link must be specified as the sequence number of the subsystem or the structure in its defining *s*-list. The User may also use the name of the subsystem or the structure, immediately followed by a colon and its sequence number.
10) The location *n* in the structure is given by its sequence number n_i in the *n*-list of its STR operator.
11) If begin and end point belong to the same structure, their chains are identical. In that case, their specifications may be enclosed in one pair of parentheses instead of two, i.e. $(c, n)\ (c, n) \rightarrow (c, n, n)$.

Rules for boundary conditions

1) The angular representation of the boundary (interface) currents must not be exact coupling, i.e. the allowed *k*-values are $-5 \le k \le 18$ with $k \neq 0$.
2) There are three types of boundary condition: (i) specular (mirror) reflection, (ii) albedo reflection, and (iii) border coupling.
3) Specular reflection must be specified by $b = 0$, i.e., no albedo matrix is involved. Since there is no current coupling, any allowed value of *k* may be specified. Regardless of this value of *k*, the only possible current output will be the total in- and out-currents per segment.
4) Albedo reflection must be specified by the name of a User-provided albedo matrix. It describes what fraction of the border currents is reflected into which energy group.
5) Diagonal albedo reflection (to the same group) is specular between the angular sectors of the border currents. These sectors are determined by *k*. Off-diagonal reflection (to other groups) is isotropic, i.e. each sector gets the same fraction of the reflected border current.
6) Border coupling must be specified by giving a second boundary section that matches the first one exactly. This causes currents that exit through the first section to re-enter through the second section and conversely, thereby assigning boundary conditions to both sections.
7) Border coupling can describe periodic and rotational boundary conditions. In Fig. 1, the border coupling of sections 1-2 and 2-3 of system *A* with sections 4-3 and 1-4, respectively, results in periodicity. Rotational symmetry is obtained by coupling section 1-2 with 1-4.

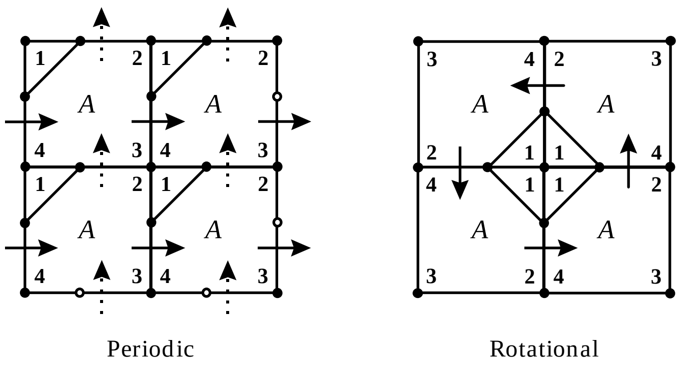


Fig. 1: Border coupling.

8) In border coupling, in deviation from Rule 4 for boundaries, but in agreement with Rule 2 for coupling in the CNX operator, the system must lie to the left of the second boundary section.

9) In border coupling, the two boundary sections must match node for node, i.e., all nodes must have been specified in the STR or CNX operator. Hence, in Fig. 1, the hollow nodes on the east and south faces of system *A* must have been specified in the STR operator.

10) Boundary conditions which, when applied repeatedly, fill the space in an unphysical way are accepted by HELIOS; see Remarks below.

11) Gamma calculations, if any, use the same boundary conditions as the neutronic calculations, i.e., types, angular-current representations and albedo names are the same. Only their albedo matrices may differ; see the ALB operator.

Remarks

Using the BDRY operator, only one subsystem can be completed with boundary conditions. This is the only system that can be calculated by HELIOS, i.e., the system itself.

It is possible to specify boundary conditions that fill the space in an unphysical way. Fig. 2 gives a subtle example. The original system is the triangle *a*. The side facing the dotted angle has specular reflection, the others have border coupling (rotation). Rotating triangle *a* yields triangle *b*. Repeated rotations will fill the top hexagon counterclockwise, as indicated by the arrow.

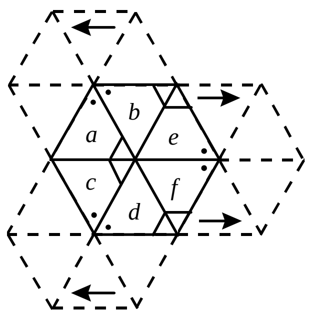


Fig. 2: Unphysical result of boundary conditions.

Reflecting *a* yields triangle *c* which, when rotated, yields triangle *d*. Further rotations will fill the bottom hexagon clockwise, as indicated by the arrow. Finally, reflecting the triangles *b* and *d* yields the triangles *e* and *f*, respectively.

Now, *f* cannot be obtained from *e* by rotation, nor can *e* from *f*; if rotated, each of them would fill the right hexagon, but in reverse directions.

Hence, an equilateral triangle with one specular side and the other two sides border-coupled cannot exist, unless the triangle has internal symmetry. On the other hand, it is readily verified that an equilateral triangle with only specular sides can exist. Also a right triangle with a specular hypotenuse whose other two sides are (rotationally) border-coupled can exist. HELIOS does not verify if the boundary conditions are physically realizable. In this respect, the User may specify and run unphysical cases. Therefore, caution must be exercised when choosing and defining boundary conditions.

Examples

```
B = CNX(A,A,A / (A:2,2,3)1(A:1,1,4) / (A:1,2,3)1(A:3,1,4) )
C = CNX(B,B,A,A / (1-2,3,4)1(3,2,1) / (1-3,3,4)1(  4,2,1)
                                    / (  4,3,4)1(2-3,2,1) )
```

1) `C = BDRY((B:1-A:2,1,1)1(0) / (B:1-A:1,4,4)1(alb) )`

2) `C = BDRY((  1-  2,1,1)1(0) / (  1-  1,4,4)1(alb) )`

3) `C = BDRY((  1-  1,4,4)1(0) / (  1-  2,1,1)1( 0 ) )`

4)
```
C = BDRY((2-2,4)(1-2,1)11(1-3,2)(1-2,1) / (1-3,2)(2-2,4)1(0)
                                        / (3,2,2)-3(alb) )
```

5)
```
C = BDRY((1-3,2)(2-2,4)1(0) / (1-2,1)(1-3,2)18(1-2,1)(2-2,4)
                            / (4,4,4)-3(alb) )
```

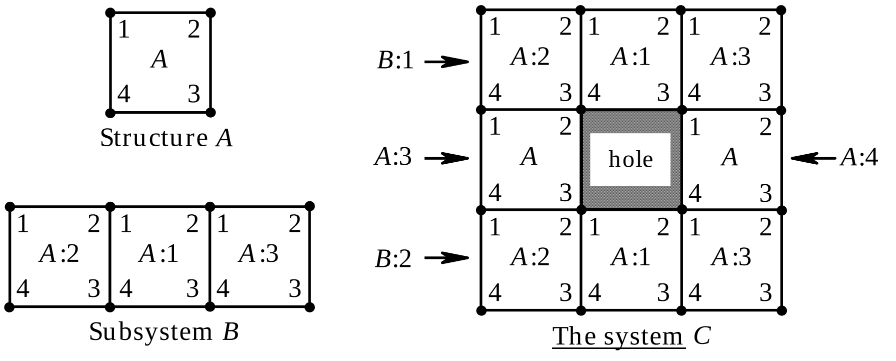


Fig. 3: Example of the BDRY operator.

All five examples refer to system *C* of Fig. 3. The first two are identical; one uses sequence numbers with prefixed names plus colon, the other uses only the sequence numbers. In these examples, the external boundary is specular all around. It has been specified by giving twice the northwestern corner node of the system. The inner boundary has albedo boundary conditions all around with isotropic reflection even for the diagonal elements ($k = 1$). Alternative ways to specify the entire inner boundary would have been (*A*:3, 2,2) or (*A*:4, 1,1), etc.

In the third example, both boundaries have specular reflection. This is unphysical, unless the structure *A* has internal symmetry with respect to its two diagonals. In that case, with only specular boundaries, system *C* could have been represented by one of the four triangles of structure *A*, formed by its midpoint and the two nodes at the extremes of each of its four sides.

The last two examples assign identical boundary conditions to the system, but they differ in the sequence and in the way their boundary sections are described. The eastern and southern sides of the external boundary are specular; the border coupling between the western and northern sides is rotational with six angular sectors per current (see CNX operator, Fig. 2 for $k = 18$). The internal boundary has albedo conditions with, for the diagonal elements, specular reflection between the nine angular sectors of the border currents ($k = -3$); the off-diagonal elements always have isotropic reflection.

CASE(), THE CASE OPERATOR

Purpose

The CASE operator signals the start of the input of a case to be run by HELIOS. It provides the case with an identifier and additional descriptive text, and specifies the cross-section library and the data-base file.

Usage

name = **CASE (** *x* **/** *d* **/** *t* [, *t* ...] **)**

name	specifies the name of the case whose input follows.
x	specifies the nuclear-data file (cross-section library).
d	specifies the data-base file with the input and the output.
t	text string of at most 80 characters.

Rules

1) *Name* is the case identifier. It will also appear as part of the second line of the three-line page heading of the output. This identifier must be identical to *name* of the RUN operator.
2) If the library *x*—specified in the CASE operator—does not exist, the case is not accepted by AURORA (nor by HELIOS).
3) If the data base *d*—specified in the CASE operator—does not exist, then *d* will be created by AURORA.
4) The text strings *t* may be used as additional text. The first *t* will appear in the third line of each output page; it must be specified by at least one character. Otherwise, the text strings are not used by HELIOS, though they are saved in the data base. However, any *t* may contain one or more data arrays for use by ZENITH. Such an array must have the structure

   << *x* = ([*i*∗] *r* [; [*i*∗] *r*… >>

   where *x* is the array's name, *i* is an integer multiplier and *r* is a real or integer constant.

Example

```
'SVEA-96' = CASE ( '~/library/lib-adj.34x'   !Library!           /
                   'BWR.hrf'                 !Output data base!  /
                   'Example of SVEA-96'                          ,
                   'Temps for ZENITH << T = (2*500; 960) >>'     )
```

The name of the case is *SVEA-96*. Thus, the page heading will contain this name in the second line of each output page; the text *Example of SVEA-96* will appear in the third line. Finally, an array *T* with the three components 500, 500 and 960 will be available to ZENITH.

CCS(), THE CIRCULAR-CYLINDRICAL-SYSTEM OPERATOR

Purpose

The CCS operator is used to define a pin system—like a fuel pin with its canning—which can be placed in a structure by the STR operator.

Usage

name = **CCS (** *r* [, *r*]… **/ [** *na* [(*nf*,*nl*)] **] / [***cid* [, *cid*]…**])**

name	represents a circular-cylindrical system that may contain radial zones, possibly subdivided in equal azimuthal sectors, thus defining its (flat-flux) regions or meshes.
r	specifies a radius (cm).
na	specifies the number of azimuthal regions per radial zone.
nf, *nl*	are the sequence numbers of the first and last radial zones in which *na* acts—if not specified, all the zones are taken.
cid	is the name of a CCS region.

Rules

1) The radii must be given in order of increasing size.
2) The first radius and the distance between two radii must at least be 10*AQRACY; the present default value is AQRACY = 1.0E-04.

   Two radii must at least differ by the factor RADFAC; at present, RADFAC = 1.0001.
3) The numbering of the radial regions goes from the centre outwards, like the first two regions in Fig. 1a.
4) *na*=0 and *na*=1 are equivalent, i.e., no azimuthal subdivision; they do not have to be specified. Otherwise, *na*<100.
5) If *na*>1, the azimuthal regions per radial zone are numbered in a clockwise direction from the north; see Fig. 1.
6) If the STR operator defines a partial CCS, it will be rotated such that its north pole lies on the radius that has the partial CCS on its right; see Fig. 1b.

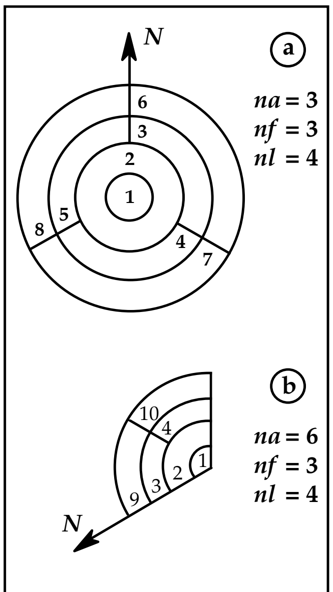


Fig. 1 CCS conventions.

7) The *cid*'s do not have to be unique, but they must always be given for the entire CCS.
8) All *cid*'s can be omitted, specifying nothing between the final slash and the final parenthesis. HELIOS then uses the region numbers of the generic (entire) CCS as names. For Fig. 1a they are: '*1*', '*2*', '*3*', '*4*', '*5*', '*6*', '*7*' and '*8*'; for Fig. 1b: '*1*', '*2*', '*3*', '*4*', '*9*' and '*10*'.
9) An individual *cid* can be omitted by specifying only its following delimiter, a comma or the closing parenthesis. HELIOS then uses its region number, according to Rule 9, as its name.
10) If any *cid* is specified, then all *cid*'s must be specified or explicitly omitted by only giving their delimiters.

Remarks

The CCS operator is used to describe a pin system, like a fuel pellet and its canning. With nothing else specified, fuel and canning define each a (flat-flux) region or mesh. Such a mesh may be too crude if the flux in these regions is important and not flat, i.e., in strongly absorbing regions with burnup or in regions with a strong scattering source. Then, a finer spatial mesh is needed for the flat-flux/source approximation of the collision probabilities.

For instance, to calculate the burnup of a burnable-absorber pin, the fuel should be radially subdivided into a number of fuel regions. For the same reason, a part of the coolant can have been included in the CCS as one or more annular regions. In rare cases, when the flux has a strong azimuthal variation, these outer regions may have to be subdivided in azimuthal sectors. In general, this is not needed for the central pin or its canning.

Later (e.g., in the overlay and output operators), the User may have to specify regions of a partial CCS with azimuthal subdivision. This can be done by name (User-defined or default), or by the sequence number in HELIOS. In the example of Fig. 1b, the default/HELIOS specifications are: '*1*'/1, '*2*'/2, '*3*'/3, '*4*'/4, '*9*'/5 and '*10*'/6.

Examples

```
1)  CCS( 0.5, 0.52, 0.6 / / '3.1% UO2+BA', gap, clad )
2)  CCC( 0.5, 0.6 / / fuel, 'clad+gap' )
3)  CCS( 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 / / BA1, BA2, BA3, BA4, BA5, )
4)  CCS( 0.5, 0.6, 0.68 / 4 (3,3) / fuel, , 'cool 1', 'cool 1',
                                         'cool 4', 'cool 4' )
```

In the first example, 3.1% enriched UO2 fuel with burnable absorber, a thin gap and its canning are specified.

In the second example, the same pin is described, but with the gap smeared into the canning. The fuel name is now *fuel*.

In the third example, the fuel pellet is divided in five radial regions with the User-specified names *BA*1 through *BA*5. The canning will get the default name '*6*'.

In the fourth example, a coolant layer, azimuthally subdivided into four regions, has been added to the pin. The six region names are: *fuel*, '2' and '*cool* 1', '*cool* 1', '*cool* 4' and '*cool* 4'.

CNX(), THE COUPLING OR CONNECTION OPERATOR

Purpose

The CNX operator is used to combine one or more structures into subsystems, and one or more subsystems and/or structures into the system, e.g. a fuel assembly. It is also used to specify the angular representation of the coupling currents across the structure interfaces.

Usage

name = **CNX (** *s* [,*s*]… **[/** (*c*, *n*[) (*c*], *n*) *k* (*c*, *n*[) (*c*], *n*)**]**… **)**

name	represents a (sub)system made up of one or more subsystems/structures, which are combined cumulatively.
s	specifies a subsystem or structure, *s* = *sname*
c	specifies a structure in *s* by its chain, *c* = [*l*–]...*l*
n	specifies a node in that structure, $n = n_i$, or $n = (x, y)$
k	specifies an angular representation of interface currents.

where:

sname	is the name of one of the subsystems/structures used to construct the present (sub)system;
l	is a link along the chain that defines the structure (the last link) whose node *n* is used in the coupling of one of the subsystems/structures *s* in the list (the first link);
n_i	is the sequence number of a node in that structure;
(x, y)	are the coordinates (cm), in parentheses, in that structure of a node that does not occur in its list of nodes.

Rules for subsystems/structures

1) The subsystem *name* is built up by a cumulative coupling of the subsystems/structures in the *s*-list.
2) Each subsystem/structure must occur as many times in the *s*-list as that subsystem/structure occurs in the new (sub)system *name*.
3) The information on each coupling of two subsystems/structures is preceded by a slash.
4) All the entries of the *s*-list must be used, i.e. there will be one more entry than couplings (slashes).
5) If the *s*-list has only one subsystem/structure as entry, there is no coupling information. The result must not appear in the *s*-list of another CNX operator; i.e. it can only be used as the final system, e.g. *system* = CNX(*structure*).

Rules for nodes

1) A node in a (sub)system/structure is specified by the location *c* of the structure in which it lies and by its sequence number *n* in the STR operator that defines the structure *c*, enclosed in parentheses, i.e., (*c*, *n*).
2) The location of a structure is specified by a chain *c* whose links *l* are connected by hyphens. The links represent all the successive subsystems—if any—that contain the structure, and the structure itself.
3) Each link must refer to an entry in the *s*-list of the CNX operator that defines the preceding link. The first link must refer to an entry in the *s*-list of the actual (sub)system. Only the last link can and must be a structure.
4) Each link must be specified as the sequence number of the subsystem or the structure in its defining *s*-list. The User may also use the name of the subsystem or structure, followed by a colon and its sequence number.
5) The node location *n* in a structure is given by its sequence number among the User-specified nodes, n_i, or by its coordinates in the local coordinate system of its STR operator, (*x*, *y*).

   Observe that nodes given as (*x*, *y*) must be preceded by a comma as delimiter, which is optional in the STR operator.
6) The sequence number refers to the *n*-list of the STR operator, supplemented—in their order of occurrence—with all the (*x*, *y*) nodes, specified in the present and all the preceding CNX operators, that were used to arrive at the structure in the actual (sub)system.
7) The coordinates must be given if and only if the node does not yet exist, i.e., it must not have been specified earlier, nor must it be an "anonymous" node created by HELIOS from STR or CNX-input—see the next three rules.
8) STR-input creates "anonymous" nodes where the radii of a partial CCS intersect with the periphery of the structure. It also creates anonymous nodes at each point where azimuthal segmentation of a CCS intersects the surrounding structure.
9) CNX-input creates "anonymous" nodes where the structure on the other side of a coupling interface has a node on that interface that does not correspond to a node of the actual structure.
10) Nodes that have been specified in the *n*-list of the STR operator of the structure are not "anonymous".

Rules for coupling

1) Two subsystems/structures are coupled by matching a pair of nodes on the periphery of one of them with a pair of nodes on the periphery of the other. This must not result in overlap.

2) When following the periphery from the first to the second node of the first pair, its subsystem/structure must lie on the right. The second pair of nodes must have its subsystem/structure on the left.
3) The two partial peripheries must match exactly, thereby becoming an interface between the two subsystems/structures. HELIOS will extend this interface to include all the segments that are in contact. If this interface is discontinuous, there are holes between the subsystems/structures (see BDRY operator, Fig. 3).
4) For each pair of nodes, both chains must have the same first link; the other links may differ. In fact, choosing the nodes far apart (e.g., in different structures), minimizes geometric rounding errors when coupling two subsystems/structures.
5) The first pair of nodes must belong to the (sub)system under construction. Hence, apart from the first coupling, both chains must have a first link that refers to an earlier-used subsystem/structure in the *s*-list.
6) The second pair of nodes must belong to a subsystem/structure that is being added, i.e., to an as yet unused entry in the *s*-list.
7) If both nodes of a pair belong to the same structure, their chains are identical. In that case, their specifications may be enclosed in one pair of parentheses instead of two, i.e., (c, n) $(c, n) \rightarrow (c, n, n)$.
8) The angular representation of the interface currents between two subsystems/structures is specified by an integer k between the node pairs describing their coupling (see Remarks). Presently, $-5 \le k \le 18$, where $k = 0$ is the exact but expensive option of no coupling at all.
9) Structures coupled by $k = 0$ will be treated throughout by collision probabilities, i.e., no current coupling across their interfaces, which thereby cease to be interfaces. Such fusioned structures must have no internal interfaces left; see the discussion of Example 2.
10) The angular representation of the interface currents across parts of an interface can still be modified by the NEWK operator.

Rules for partial CCS-es

1) Partial CCS-es must not be involved in exact coupling, $k = 0$. They may, however, be coupled by $k \neq 0$, i.e., by interface currents.
2) According to CCS Rule 6 of the STR operator, only a partial CCS can have nodes within its outer radius that can be used, namely: at its centre and at the intersections of its radii with the periphery of the structure. Therefore, a coupling that involves a partial CCS may generate only border nodes that coincide with its centre or its radii.

   This is shown for two unlikely situations in Fig. 1. In the left figure, shifting the lower structure to the right has generated two illegal nodes. In the right figure, shifting an inner line segment in the lower structure has generated one illegal node.

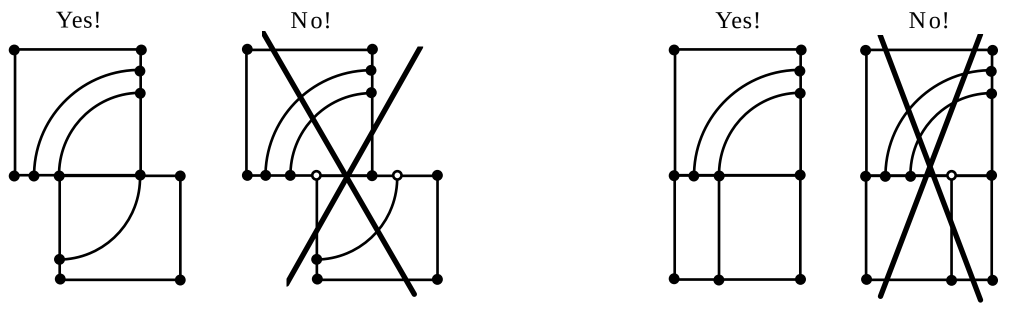


Fig. 1: Partial CCS couplings.

Remarks

The STR and CNX operators provide great flexibility to build up the final system. For example, defining *cell* as a pin-cell structure, one CNX operation could define the subsystem *row* as a row of cells, while another CNX operation connects these rows into the system, *assembly*. On the other hand, the CNX operator could define *qcell* as a subsystem consisting of 2x2 cells and continue building up *assembly* from there. It is also be possible—though impractical—to define the assembly as a structure, say *assbly*, and then define *assembly* = CNX(*assbly*). It is recommended to have as few subsystem levels as possible, e.g., building *assembly* directly from *cell* structures, skipping the *row* subsystem.

Owing to the cumulative build-up, the subsystems are coherent. However, they do not have to be solid, i.e., they may contain holes. It is therefore possible to define the gaps around a BWR assembly as a subsystem, *gaps*, the inside as another subsystem, '*box* + *fuel*', and combine them into the system, *assembly* = CNX(*gaps*, '*box* + *fuel*' / …).

Mathematically, the system is subdivided into space elements inside which collision probabilities are used. The space elements are coupled by interface currents. Within the structures collision probabilities are used, so no space element can be smaller than a structure. However, two or more structures can be fusioned into one space element by specifying $k = 0$ coupling between them (see Rules 8 and 9 for coupling).

The currents that couple the space elements are spatially constant along the interface segments between two successive nodes. More spatial detail can be obtained by using smaller line segments. This practice is only recommended if accompanied by a finer subdivision of the space elements into flat-flux regions, such that each interface node also belongs to an internal region boundary.

The angular dependence of the interface currents is discretized by a subdivision of the directional half-spheres into sectors of equal weight with respect to the currents. Inside each sector, the currents are assumed to be constant in the azimuthal angle and to vary with the cosine of the polar angle.

The integer *k* of Rule 8 specifies the angular discretization. If $k = 0$ there is no current coupling and the collision probabilities extend across the interface. All the positive values of *k* are shown in Fig. 2. There, the lines describe the azimuthal subdivisions and the arcs the polar ones. The five negative *k*-values correspond to the same number of azimuthal and polar subdivisions, as illustrated for $k = -3$. To allow finer discretizations is too costly; in that case, it would be better to use directly collision probabilities, i.e. $k = 0$.

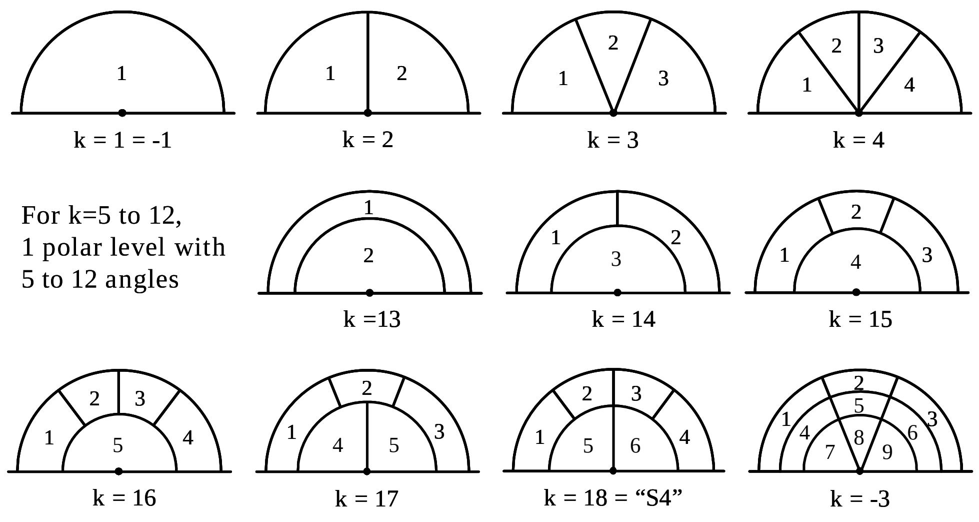


Fig. 2: Angular sectors for current coupling.

Examples

1)
```
C = CNX(A,B / (B:2,4,1)1(A:1,1,4) )
or
C = CNX(A,B / (2,4,1)1(1,1,4) )
or
C = CNX(A,B / (2,4,1)1(1,1)(1,4) )
or
C = CNX(A,B / (2,4)(B:2,1)1(A:1,1)(1,4) )
```

This example is the first step in the build-up of system *D* in Fig. 3. Observe the various ways to specify the node pairs. Because only structures are coupled, the two chains of each pair are equal and consist of one link. Hence, although each node may have its own chain, one set of parentheses suffices per pair. Notice that either a sequence number or "name:sequence number" is used to specify the chain links. The latter way of specifying the chain links is not commonly used.

2)
```
D = CNX(C,C,B,C / (C:4–A:1,3)(C:4–B:2,2)1(C:2–B:2,2(C:2–A:1,3)
       /(C:4-A:1,1,2)1(C:1–A:1,3,2)/(C:1–B:2,1,(0.8,0))-2(B:3,7,3) )
or
D = CNX(C,C,B,C / (4–1,3)(4–2,2)1(2–2,2)(2–1,3)
                / (4–1,1,2)1(1–1,3,2) / (1–2,1,(0.8,0))-2(3,7,3) )
or
```

```
D = CNX(C,C,B,C / (4–1,3,4)1(2–2,2,1)
                / (4–1,1,2)1(1–1,3,2) / (1–2,1,(0.8,0))-2(3,7,3) )
```

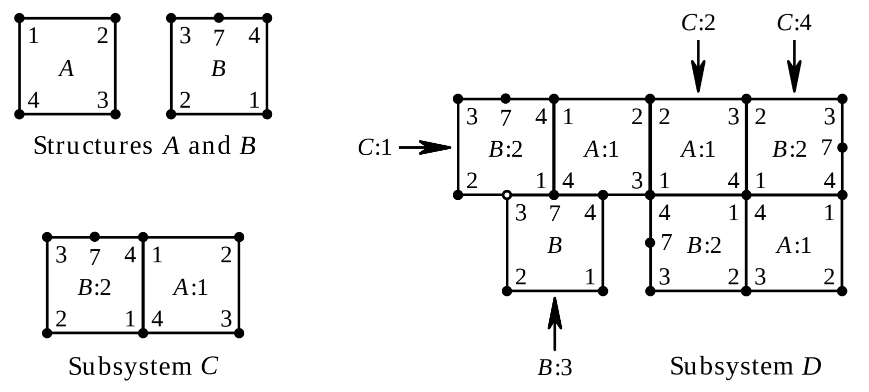


Fig. 3: Example of building up system *D* from the structures *A* and *B*.

This second example completes the build-up of system *D* in Fig. 3. Because subsystem *C* occurs three times in system *D*, it has three geometric identities, i.e., sequence numbers in the *s*-list of the CNX operator defining system *D*. Further, any entry of the *s*-list could have been taken as the nucleus of growth for the system; here it is *C*:4.

The last coupling, that of structure *B*:3 with subsystem *C*:1, uses only nodes of *B*. In *C*:1–*B*:2, it uses node 1 and the so far undefined (hollow) node that must now be defined by its coordinates; in *B*:3, it uses nodes 7 and 3 (though not shown here, *B* has also nodes 5 and 6). It should be observed that this is against the recommendation made in Rule 4 for coupling - it would have been better to use *B*:3's nodes 4 and 3, as they lie farther apart.

Finally, according to Rule 9 for coupling, the last interface coupling must be anything but $k=0$. On account of Rule 3 for coupling, segment (*B*:3, 4,7) would also get $k=0$. Thereby, structure *B*:3 would be fusioned with both the structures *B*:2 and *A*:1 of subsystem *C*:1 into a single space element. However, segment (*C*:1-*B*:2, 4,1) — or segment (*C*:1-*A*:1, 1,4) — would have become an internal interface, which is not allowed.

3)
```
Pin  = CCS( 0.3,0.4 / / fuel,clad)
Cell = STR((0,0) (0,1) (1,1) (1,0) / 4,cool / Pin(0.5,0.5) /)
Gaps = STR(( 0   ,0   )(-0.6,0.6)(4.6,0.6)(4,0  )             ! 1-4 !
           (-0.1,0.1)(-0.3,0.3)(0   ,0.1)(0,0.3)   (0   ,0.6) ! 5-9 !
           ( 1   ,0   )( 1   ,0.1)(1   ,0.3)(1,0.6)            !10-13!
           ( 2   ,0   )( 2   ,0.1)(2   ,0.3)(2,0.6)            !14-17!
           ( 3   ,0   )( 3   ,0.1)(3   ,0.3)(3,0.6)            !18-21!
           ( 4   ,0.1)( 4   ,0.3)(4   ,0.6)(4.1,0.1)(4.3,0.3) !22-26!
           / 4,gap / !No pins! /
                1, 5, 7,film;  1, 7,11,10,film; 10,11,15,14,film;
            14,15,19,18,film; 18,19,22, 4,film;     4,22,25,film;
             5, 6, 8, 7,box ;  7, 8,12,11,box ; 11,12,16,15,box ;
            15,16,20,19,box ; 19,20,23,22,box ; 22,23,26,25,box ;
             6, 2, 9, 8,gap ;  8, 9,13,12,gap ; 12,13,17,16,gap ;
            16,17,21,20,gap ; 20,21,24,23,gap !last reg default!)
```

```
Row  = CNX(Cell,Cell,Cell,Cell/ (1,3,4) 0 (2,2,1)
                              / (2,3,4) 0 (3,2,1)
                              / (3,3,4) 0 (4,2,1) !left-right!)
Fuel = CNX( Row, Row, Row, Row/ (1-1,2)(1-4,3) 0 (2-1,1)(2-4,4)
                              / (2-1,2)(2-4,3) 0 (3-1,1)(3-4,4)
                              / (3-1,2)(3-4,3) 0 (4-1,1)(4-4,4))
Rest = CNX(Gaps,Gaps,Gaps,Gaps/ (1,3,4) 1 (2,2,1)  !N+E!
                              / (2,3,4) 1 (3,2,1)  ! +S!
                              / (3,3,4) 1 (4,2,1)  ! +W!)
BWR  = CNX(Fuel,Rest/(1-1-1,1)(1-4-4,3) 1 (2-3,4)(2-1,4))
```

This example shows the build-up of the 4×4 BWR mini-assembly of Fig. 4. It includes also the CCS and STR operators. The pitch is 1 cm; the coolant film, box wall and outer gaps are 0.1, 0.2 and 0.3 cm thick.

Observe the build-up of the subassemblies. *Row* is built up from the left to the right, *Fuel* from the bottom to the top, and *Rest* in clockwise direction starting in the North. Finally, the system *BWR* is constructed by coupling *Fuel* and *Rest* along their periphery from the SW to the NE corners. Further, all the structures *Cell* in the subassembly *Fuel* have exact coupling ($k = 0$), while the other couplings—*Gaps* with *Gaps* and *Gaps* with *Fuel*—are by cosine currents ($k = 1$).

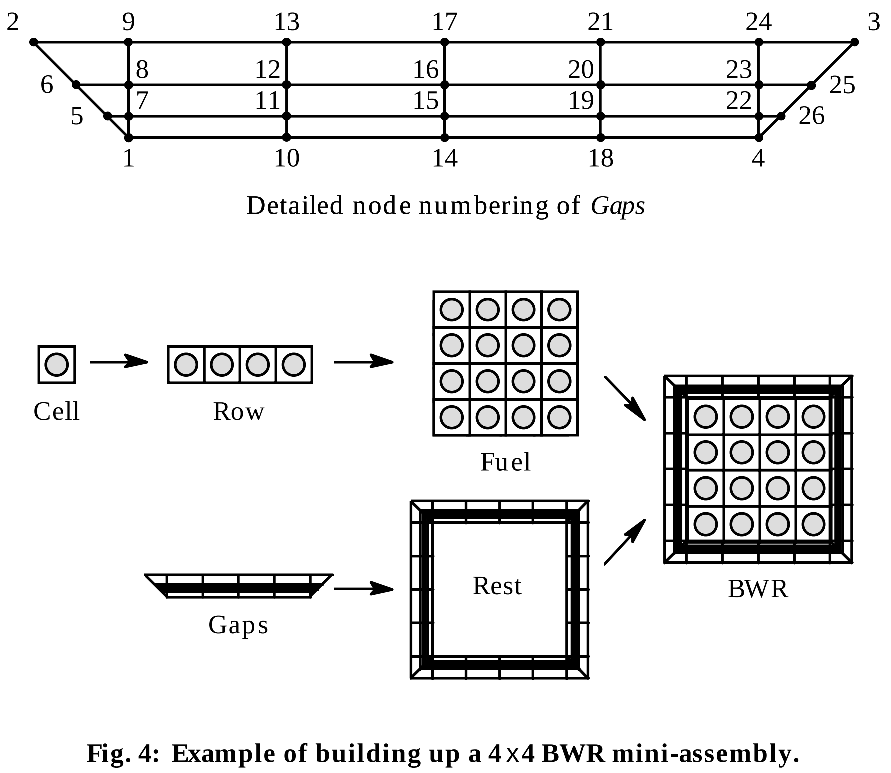


Fig. 4: Example of building up a 4×4 BWR mini-assembly.

CUR(), THE CURRENT-EDIT OPERATOR

Purpose

The CUR operator is used to specify the partial currents across one or more edit faces defined by one or more FACE operators in an edit-group structure defined by a GROUP operator. These currents are saved in the data base for later output processing.

Usage

name = **CUR (** [T/] *gr*, *fa* [, *fa*] … **)**

name	represents one or more output sets of partial (in and out) currents across one or more given faces in a given group structure.
T	is the letter T to also save time-averaged currents.
gr	is the name of an edit-group structure.
fa	is the name of an edit face.

Rules

1) The result of a CUR operator (*name*) specifies one or more sets of partial in- and out-currents across one or more given edit faces (*fa*) in a given edit-group structure (*gr*). The *fa*'s must be unique inside each CUR operator.
2) Time-averaged partial currents may also be saved by specifying 'T/' before *gr*. The average is over the preceding burnup or decay step in the PATH or TREE operator at which the actual output is produced.
3) The edit-group structure *gr* defines whether the partial currents are for neutrons or gammas—see the GROUP operator.
4) An edit face is a set of not necessarily coherent peripheral segments of one or more space elements—see the FACE operator. The combined length of these segments is always saved in the data base.
5) A space element is made up of one or more structures defined by the STR operator:

   One structure, if this structure is coupled with its surroundings by currents, $k \neq 0$ (see CNX and NEWK operators) and/or any type of boundary condition.

   More than one structure, if these structures are connected by collision probabilities, $k = 0$.

6) In-currents are partial currents into the structure(s) that define the peripheral segments (FACE operator, Rule 2). Out-currents are partial currents out of the structure(s) that define these segments.

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data base (from where they can be retrieved by an output processor). For instance, the CUR operator could be used to save only two-group in and out-currents of neutrons at the borders of a fuel assembly and about its four corners. These currents can be used to evaluate discontinuity factors of homogenized fuel assemblies, which correct for homogenization errors in 3D reactor calculations.

The in- and out-currents (j^- and j^+) can be used to define fluxes (ϕ), net currents (J) and albedos (a), according to:

$$\phi = 2(j^- + j^+),\ J = j^- - j^+ \text{ and } a = \frac{j^-}{j^+}. \tag{1}$$

Example

```
1) Borders =CUR(ng2, north,east,south,west)
```

In this example, in- and out-currents of neutrons across the four borders of a fuel assembly will be saved in the two-group structure *ng*2. Here it is assumed that *north, east, south* and *west* have been defined by four FACE operators as the collected boundary segments of each of the four sides of the assembly.

DBMAT(), THE RESTART-MATERIAL OPERATOR

Purpose

The DBMAT operator is used to retrieve number densities and related material data from a data-base. These materials have (or haven't) undergone depletion in an earlier case, in which their data have been saved by a DUMP operator. From this information the HELIOS input of the dumped materials is constructed. This input has the same format as the "U" arrays generated by the MAT operator. If possible, the DBMAT operator also generates a default material overlay.

There are two variants to retrieve number densities: (i) for one or more materials at a specified burnup, or (ii) for a set of materials—e.g., the fuel in the fuel pins—at all burnups along the PATH where the dumps were made. In the latter variant, the DUMP operator acts as a PATH operator and the construction of the default material overlay must be possible.

Usage

name = **DBMAT (** *df*; *ca*; *dname* / (*st*) [, *eb*] [/*l*] **)**

name	is the identifier (name) of a set of dumps of one or more material isotopics at a given burnup level.
df	is the file name of the (HERMES) data-base where the dumps reside.
ca	is the identifier (name) of the case whose dumps are requested.
dname	is the name of the DUMP operator in the input of *ca* that produced the isotopics dumps.
(*st*)	is the name of a state (on a path in the input of *ca*) whose isotopics are requested.
eb	if specified: the burnup of (*st*) at which its isotopics are requested; if not specified: isotopics of (*st*) are available at all the burnups where dumps have been made in the case *ca*.
l	is a possible addition/subtraction of a link to/from the chain(s) that point(s) at the region(s)—in an AREA operator in the *ca* input—whose dumped isotopics are requested:

$l > 0$	the explicit link number to be added;
$l = 0$	no action, i.e. the same as not specifying *l*;
$l < 0$	the first link must be deleted (the value of *l* is irrelevant).

Rules in general

1) In the case *ca* that produces a dump, the materials are specified by an AREA operator *ar*, which points at the regions in which these materials occur (see AREA and DUMP operator). The number of restart materials available is equal to the number of regions at which *ar* points, *NRA*.

2) Referring to Rule 1: *ar* points at regions by chains of the type

$$[ss-[ss-]\ldots]-st-cc-rg.$$

ss, *st*, *cc* and *rg* are subsystems (defined by the CNX operator), structures (defined by the STR operator), CCS-es (defined by the CCS operator) and regions (defined inside the STR and CCS operators).

3) All the links of the chains in *ar* are sequence numbers, except for the *rg* link, which also may be a name (indicating all regions of that name) or an asterisk (indicating all the regions). For regions outside CCS-es, $cc = 0$.

4) This operator generates the material overlay '*name-db*' if the chain(s) of *ar* in *ca*, possibly with an added/deleted first link (l), make sense in the actual case.

5) $l \neq 0$ always refers to a subsystem link. If $l < 0$ it deletes the first link of the chain(s) in *ar*, which must be a subsystem. The remaining chain(s) should, but don't have to, exist in the actual system. If $l > 0$ it should, but doesn't have to, refer to an existing link in the actual system. If not, '*name-db*' is not created.

6) An original burnable/non-burnable material with heavy isotopes remains burnable/non-burnable as a restart material. Any other restart material is burnable/non-burnable according to its isotopes (see MAT operator)—if it should be non-burnable, it must be redefined.

7) DBMATs with specified burnup and DBMATs with unspecified burnup may be used in the same run.

Rules for restart materials at a specified burnup

1) The DBMAT operator generates the *NRA* restart materials '*name-1*' through '*name-NRA*'. *NRA* is the number of materials dumped at each burnup by the dump *dname* in the case *ca*. This number is determined by the AREA operator in *ca* that produced the dumps.

2) The materials '*name-1*' through '*name-NRA*' can be used individually in the same way as materials defined by the MAT operator. But in an OVLM operator (see there) a restart material must be assigned to one region only.

3) The range of materials '*name-i*'–'*name-j*', with $i < j \leq NRA$, can be used in the same way as a material defined by the MAT operator. But in an OVLM operator (see there) it must be assigned to exactly $j-i+1$ different regions, and it must not be used in an indirect reference.

4) The entire set of *NRA* restart materials, *name*, can be used in the same way as materials defined by the MAT operator. However, in an OVLM operator (see there) it must be assigned to precisely *NRA* different regions.

Rules for restart materials at unspecified burnup

1) All DBMATs with unspecified burnup must assign a set of restart materials to the same regions. Their DUMPs must have the same AREAs *ar*, name for name and region for region, while their materials must have the same isotopes, isotope for isotope—though at different burnups. They must also reside in the same data-base file *df*.

2) The names of the individual restart materials of a DBMAT with unspecified burnup must not appear in any material overlay definition (see OVLM operator), nor the name of the DBMAT itself to specify all its materials.

3) The material overlay '*name-db*' that is generated by a DBMAT with unspecified burnup must exist (see Rules in general, 4), i.e. its AREA must exist and must have the same number of regions. If the restart materials have to be in the same positions as in the case *ca* where they were dumped, the build-up of that system (CCS, STR and CNX operators) must be exactly the same, apart from adding/deleting a possible link *l* (see Rules in general, 4 and 5).

4) If there are more DBMATs with unspecified burnup, their material overlays '*name-db*' assign the same materials to the same regions. So it is irrelevant which of these names is used.

5) Material overlays generated by DBMATs with unspecified burnup must not duplicate these materials. Thus, in view of Rule 4 above, there may only be one OVSM operator that contains a '*name-db*' of a DBMAT with unspecified burnup.

6) The name of a DBMAT with unspecified burnup must only be used in lieu of a path name to define the isotopics to be used in a TREE operator and/or the initial isotopics of a PATH operator.

   The states active in such a path or tree will use the isotopics of the dumped materials referred to by the DBMAT operator. All the other materials that are present (those not referred to in '*name-db*') will be at zero burnup.

7) The isotopics of a PATH operator are based on a DBMAT with unspecified burnup if its initial isotopics are given directly by that DBMAT, or if they are given by an earlier PATH whose initial isotopics are given by such a DBMAT.

   The isotopics of a TREE operator are based on a DBMAT with unspecified burnup if they are given directly by that DBMAT (in lieu of a path name) or by a PATH whose initial isotopics are based on such a DBMAT.

   A PATH or TREE operator whose isotopics are not based on a DBMAT with unspecified burnup must not contain a state that contains an OVSM that contains anywhere a '*name-db*' based on a DBMAT with unspecified burnup, directly or indirectly via a background OVSM, even if other OVLMs in that same OVSM entirely overwrite the '*name-db*'.

Remarks

In HELIOS, most branch-off calculations are part of the case in which the burnup calculations are done. They are based on the TREE operator, and on the definition of different states, using the STAT, ISOX and OVS∗ operators (∗ = M, D or T). Examples of the flexibility these operators offer are: (i) replacing bypass-gap water by control-blade materials and vice versa; (ii) replacing burnt fuel in one or more fuel pins by fresh fuel; and (iii) start burnup with four fresh assemblies and replace successively the burnt assemblies by fresh ones.

This list of examples can be expanded with many more. However, some of them are somewhat artificial and not very computer-efficient. For instance, control branch-offs in a BWR require all calculations to be done with the control-blade mesh in the bypass gaps. This unnecessarily increases computational time for the other calculations.

Also, calculations in cold and hot dimensions cannot be done in the same case. Of course, thermal expansion/contraction effects could be approximated by assigning density-correction factors to the various regions, using the OVLD and OVSD operators. These factors are the products of temperature differences and linear expansion coefficients of the regions. Otherwise, such effects must be found from different cases. This would require, amongst others, detailed number-density input for all the fuel regions at each burnup level to be examined. This is only feasible at zero burnup.

Input for fuel storage calculations, where the fuel pins of an assembly are placed in a different geometry, would not be trivial. Again, the fuel number densities must somehow be entered by MAT operators.

To facilitate calculations in which one or more materials with a certain depletion are to be re-used, the DUMP and DBMAT operators have been introduced. Such calculations are called restart calculations because their probably most frequent application is that of "restarting" the calculations with the same fuel pins, though in a (completely) different geometry.

The DUMP operator causes the isotopics of one or various materials at one or more burnups to be dumped in a data-base. Apart from the number densities, also the initial heavy-metal density (g/cm^3) and the physical burnup of each material are stored. The DBMAT operator retrieves these data and puts them as restart materials at the User's disposal.

The restart OVLM

The AREA operator *ar*, used with the DUMP operator *dname*, is one of the data sets that are stored in the restart data-base *df*. So the chain(s) that point at the regions of the dumped materials in the original case *ca* are available. They are utilized by automatically creating, if possible, the OVLM operator '*name–db*'. This operator assigns the *NRA* restart materials to regions by position identifiers (*pid*'s). These *pid*'s are the region identifiers of *ar* (*aid*'s), possibly with an *ss* link deleted or added,

'*name–db*' = **OVLM(***name* / [*ss*–] *aid*, **[** [*ss*–] *aid* **]** … **)** .

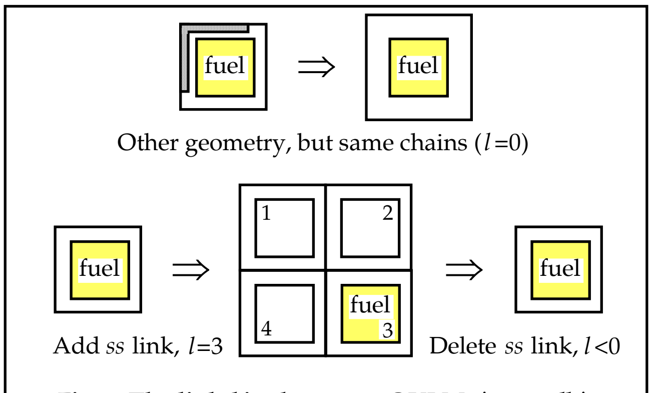


Fig. 1 The link *l* in the restart OVLM, *'name-db'*.

If in the restart OVLM no first link has been added/deleted to/from the *aid*'s, it can be used to easily move, e.g., spent fuel to a storage pool. Here it has been assumed that the *pid*'s of the fuel are the same in both systems. The top part of Fig. 1 illustrates this situation.

The addition/deletion of a first *ss* link to/from the *aid*'s facilitates the insertion of the fuel of an entire assembly into a four-assembly case. Likewise, the fuel of one of four assemblies can readily be used in a one-assembly case. These situations are shown in the bottom half of Fig. 1.

Direct references of restart materials in an OVLM

For DBMATs with specified burnup, there are three ways to refer directly to restart materials in an OVLM:

ov1 = OVLM(*name* / *pid*, … *pid*)

ov2 = OVLM('*name–i*' / *pid*)

ov3 = OVLM('*name-i*'–'*name-j*' / *pid*, … *pid*)

In *ov1*, the *pid*'s must specify *NRA* different regions; no region may be specified more than once. In *ov2*, *pid* must specify one single region. In *ov3*, finally, *pid*, … *pid* must specify $j-i+1$ different regions. However, the number of *pid*'s in *ov1* or *ov3* does not have to be equal to *NRA* or $j-i+1$, because one *pid* can specify more than one region, e.g. 2–1–*fuel*, or 4–2–1–$*$.

Indirect references of restart materials in an OVLM

For DBMATs with specified burnup, indirect referencing to restart materials in an OVLM is either to the entire set of *NRA* dumped materials, or to one or more individual restart materials:

ov4 = OVLM(*os–ov1–name* / *pid*, … *pid*)

ov5 = OVLM(*os–ov1–name*(p_m, p_n–p_r) / *pid*, … *pid*)

ov6 = OVLM(*os–ov2–*'*name–i*' / *pid*)

ov7 = OVLM(*os–ov2–*'*name–i*' (1) / *pid*)

ov8 = OVLM(*os–ov3–*'*name–i*' / *pid*)

$ov9$ = OVLM(os–$ov3$–'*name–i*' (1) / *pid*)

In *ov4*, the *pid*'s must specify *NRA* different regions—like in *ov1*. In *ov5*, the *pid*'s must specify as many different regions as are specified by those *pid*'s of *name* that are pointed at by p_m, p_n–p_r. This is nothing but an indirect reference to the restart materials in the regions specified by p_m, p_n–p_r.

Overlays *ov6* through *ov9* specify one region each. Indirect references to a single restart material are not necessary because each restart material belongs to a unique region. Therefore *p* must be 1 in *ov7* and *ov9*; any other value would be wrong. Let

$ov10$ = OVLM('*name-1*'–'*name-3*' / 1–1–$*$, 1–0–cool),

then the two *pid*'s must specify three regions, for instance, 1–1–1, 1–1–2 and 1–0–1. An indirect reference with *ov10*–'*name–2*' then specifies the chain 1–1–2.

Finally, the number of *pid*'s in *ov4* must be equal to the number of *pid*'s that belong to *name* in *ov1*. Likewise, the number of *pid*'s in *ov5* must be equal to the number of *pid*'s pointed at by p_m, and p_n–p_r. Besides, corresponding *pid*'s in *ov1* and *ov4*/*ov5* must define the same number of regions. For example, in

$ov11$ = OVLM(*name* / pid_1, pid_2, pid_3),

let $pid_1/pid_2/pid_3$ specify 7/8/3 regions. Then in

$ov12$ = OVLM(*os*–*ov11*–*name*(1,2) / pid_4, pid_5),

pid_4 and pid_5 correspond to pid_1 and pid_2, which are referred to in the indirect reference (1,2). Moreover, pid_4/pid_5 must specify 7/8 regions.

Examples

Examples 1 and 2 deal with the SVEA-96 assembly shown in Fig. 2. Both the entire assembly and one of its four identical mini bundles, run as an isolated case, have produced isotopic dumps. Let the fuel map and fuel assignment for a mini bundle of the original cases, *SVEA* and *Mini*, be as follows (a UO_2 pin has one fuel region, a BA pin has 6):

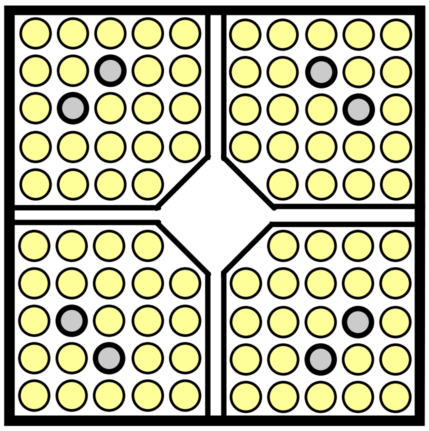


Fig. 2 SVEA-96 assembly of Examples 1 and 2.

```
$FuelPins = PAR(U198,U264,U317,U317,U264,
                U264,U350,G350,U400,U350,
                U317,G350,U350,U400,U400,
                U317,U400,U400,U400,U350,
                U264,U350,U400,U350     )

Fuel      = OVLM('UO2-1.98'   /[*-]U198-*-fuel/
                 'UO2-2.64'   /[*-]U264-*-fuel/
                 'UO2-3.17'   /[*-]U317-*-fuel/
                 'UO2-3.50'   /[*-]U350-*-fuel/
                 'UO2-4.00'   /[*-]U400-*-fuel/
                 'UO2-3.50+BA'/[*-]G350-*-fuel)
```

The extra link with an asterisk denotes the subsystems (mini bundles) of *SVEA*, NW/NE/SE/SW = 1/2/3/4—it does not exist in *Mini*.

The following dumps have been made from *Mini* and *SVEA*:

```
FuelArea   = AREA((1-24)-1-fuel)            ! 24 cells/mini !
FuelDump   = DUMP('dump.hrf'/FuelArea,Path)   ! All burnups !
FuelAreaNW = AREA(1-(1-24)-1-fuel)
FuelAreaNE = AREA(2-(1-24)-1-fuel)
FuelAreaSE = AREA(3-(1-24)-1-fuel)
FuelAreaSW = AREA(4-(1-24)-1-fuel)

FuelDumpNW = DUMP('dump.hrf'/FuelAreaNW,Path/500) !500 MWd/t!
FuelDumpNE = DUMP('dump.hrf'/FuelAreaNE,Path/500)
FuelDumpSE = DUMP('dump.hrf'/FuelAreaSE,Path/500)
FuelDumpSW = DUMP('dump.hrf'/FuelAreaSW,Path/500)
```

1) An example of how to run *Mini* with restart materials at a specified burnup uses the following DBMAT, OVLM and OVSM operators:

```
MiniDump = DBMAT('dump.hrf';Mini;FuelDump  /(State),500)
SVEADump = DBMAT('dump.hrf';SVEA;FuelDumpSW/(State),500/-9)

Fuel     = OVLM( MiniDump                  /( 1-24)-1-fuel )
Fuelx    = OVLM('MiniDump-1' -'MiniDump-7' /( 1- 7)-1-fuel /
                'MiniDump-8' -'MiniDump-13'/     8 -1-fuel /
                'MiniDump-14'              /     9 -1-1    /
                'MiniDump-15'              /    10 -1-1    /
                'SVEADump-16'              /    11 -1-fuel /
                'SVEADump-17'-'SVEADump-22'/    12 -1-(1-6)/
                'SVEADump-23'-'SVEADump-34'/(13-24)-1-1    )

Osm1     = OVSM(NonFuel, Fuel)
Osm2     = OVSM(NonFuel, Fuelx)
Osm3     = OVSM(NonFuel,'MiniDump-db')
Osm4     = OVSM(NonFuel,'SVEADump-db')
```

In this example, *Osm1* gets its fuel materials from *Fuel* according to Rule 4 for restart materials at a specified burnup. *Osm2* gets its materials from *Fuelx* according to Rules 2 and 3 for restart materials at a specified burnup. Observe that the 8th and 12th pins have six fuel regions each, and that materials from two dumps are used (with the first link removed from the second dump). *Osm3* and *Osm4* use the restart OVLMs of the two dumps used.

2) An example of how to run *SVEA* with restart materials at a <u>specified burn-up</u> uses the following DBMAT, OVLM and OVSM operators:

```
FDump    = DBMAT('dump.hrf';Mini;FuelDump  /(State),500/4)
FDumpNW = DBMAT('dump.hrf';SVEA;FuelDumpNW/(State),500)
FDumpNE = DBMAT('dump.hrf';SVEA;FuelDumpNE/(State),500)
FDumpSE = DBMAT('dump.hrf';SVEA;FuelDumpSE/(State),500)
FDumpSW = DBMAT('dump.hrf';SVEA;FuelDumpSW/(State),500/-1)

Fuel     = OVLM('UO2-1.98'   /[*-]U198-*-fuel/
                'UO2-2.64'   /[*-]U264-*-fuel/
                'UO2-3.17'   /[*-]U317-*-fuel/
                'UO2-3.50'   /[*-]U350-*-fuel/
                'UO2-4.00'   /[*-]U400-*-fuel/
                'UO2-3.50+BA'/[*-]G350-*-fuel)
MiniSE   = OVLM( FDump                   /3-(1-24)-1-fuel)
MiniSW   = OVLM('FDump-1'  -'FDump-34'   /4-(1-24)-1-fuel)
SVEANE   = OVLM( FDumpSW                 /2-(1-24)-1-fuel)
SVEASE   = OVLM('FDumpSE-1'-'FDumpSE-34'/3-(1-24)-1-fuel)

Osm1     = OVSM(NonFuel, Fuel)
Osm2     = OVSM(Osm1   / MiniSE)
Osm3     = OVSM(Osm1   / MiniSW)
Osm4     = OVSM(Osm1   /'FDumpNE-db')
Osm5     = OVSM(NonFuel,'FDumpNW-db',SVEANE,MiniSE,'FDump-db')
```

In this example, *Osm1* gets its fuel materials from *Fuel* without any restart materials. In *Osm2* and *Osm3*, the SE and SW bundles have got the restart compositions of *Mini* at 500 MWd/t, which have been defined according Rules 4 and 3, respectively. In *Osm4* the restart OVLM of the NE bundle of the original *SVEA* case provides the material compositions in the actual NE bundle. *Osm5*, finally, has restart materials of the NW/SW bundles of *SVEA* in its NW/NE bundles (note the definition of *SVEANE*), and the same restart materials of *Mini* in its SE and SW bundles. The restart overlay '*FDump-db*' could be created because of the addition of a first link, $l = 4$, which assigns its data to the 4th (SW) bundle. (However, the restart OVLM '*FDumpSW-db*' cannot be created, because the first link has been deleted.)

3) In this example, the use of restart materials with <u>unspecified burnup</u> to do branch-off calculations is illustrated in an 8x8 BWR lattice with four BA pins and two water tubes; see Fig. 3. Let the pin-cell structures be numbered from the

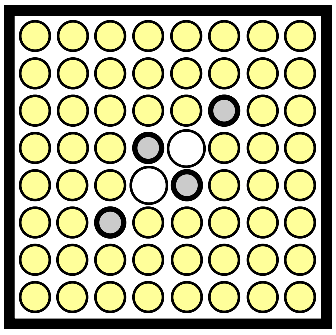


Fig. 3 BWR assembly of Example 3.

left to the right and from the top down. Thus, structures 29 and 36 contain the water tubes. Three burnup calculations are done: at 0, 40 and 70% steam void in the coolant. From each of these burnup paths one set of branch-off calculations is made, varying the void from 0 to 40%, 40 to 70% and 70 to 0%, respectively.

The branch-off calculations can be done in the same case where the burnup is done. The relevant operators are listed below. Let the operator *fuels* assign the fresh fuel materials to the fuel regions of the pins, and let the three water materials be called *H2Ov1*, *H2Ov2* and *H2Ov3*. The overlay *out1* assigns *H2Ov1* not only to all coolant regions but also to the remaining (water) regions in the outside gaps and the water tubes, which are always filled with non-voided water. The three burnup paths are *B0*, *B4* and *B7*. Finally, the three sets of branch-off calculations are done using the three TREE operators *T40*, *T74* and *T07*.

The last four operators are not necessary because the branch-off calculations are done in the same case. However, they are needed in this example to demonstrate how the same branch-off calculations can be done in a next case using the DBMAT operator. They store (dump) the isotopics and other material information in the data-base '*x.hrf*'. Observe also the structure numbering in the AREA operator, where the cells with the water tubes are skipped because they have no *fuel* regions.

```
fuels = OVLM($fuels)
out1  = OVLM(H2Ov1/*-**/
                Zirc2/*-0-box,*-*-clad)          !  0% void !
out2  = OVLM(H2Ov2/*-0-cool)                     ! 40% void !
out3  = OVLM(H2Ov3/*-0-cool)                     ! 70% void !

osm00 = OVSM(out1,fuels)                         !  0% void !
osm40 = OVSM(osm00/out2)                         ! 40% void !
osm70 = OVSM(osm00/out3)                         ! 70% void !

v0    = STAT(osm00,osd,ost,$pow)                 !  0% void !
v4    = STAT(osm40,osd,ost,$pow)                 ! 40% void !
v7    = STAT(osm70,osd,ost,$pow)                 ! 70% void !

B0    = PATH(/C,(v0),10000/10,12000,15000,35000/4)    !  0% !
B4    = PATH(/C,(v4),10000/10,12000,15000,35000/4)    ! 40% !
B7    = PATH(/C,(v7),10000/10,12000,15000,35000/4)    ! 70% !

T40   = TREE(B0/(v4)/15000/5, 35000/2)
T74   = TREE(B4/(v7)/15000/5, 35000/2)
T07   = TREE(B7/(v0)/15000/5, 35000/2)

arfu  = AREA((1-28,30-35,37-64)-1-fuel)
dump0 = DUMP('x.hrf'/arfu,B0)
dump4 = DUMP('x.hrf'/arfu,B4)
dump7 = DUMP('x.hrf'/arfu,B7)
```

To do the branch-off calculation in a next case, that case needs the operators shown below. The three DBMATs with unspecified burnup make it possible to use the isotopics of the previous case, which was called *dump*, as restart materials in the present case. The DBMATs have the same names as the PATHs from which their isotopics were dumped in the data-base. This is not necessary, but it causes the TREE operators of the present case to be identical to those of the first case. Notice that instead of the material overlay *fuels* of the previous case, here the automatically-generated material overlay *'B7-db'* is used (see Rules in general, 4). By virtue of Rule 4 for restart materials without specified burnup, *'B0-db'* or *'B4-db'* could equally well have been used.

```
B0    = DBMAT('x.hrf';dump;dump0/(v0))
B4    = DBMAT('x.hrf';dump;dump4/(v4))
B7    = DBMAT('x.hrf';dump;dump7/(v7))

out1  = OVLM(H2Ov1/*-**/
             Zirc2/*-0-box,*-*-clad)            !  0% void !
out2  = OVLM(H2Ov2/*-0-cool)                    ! 40% void !
out3  = OVLM(H2Ov3/*-0-cool)                    ! 70% void !

osm00 = OVSM(out1,'B7-db')                      !  0% void !
osm40 = OVSM(osm00/out2)                        ! 40% void !
osm70 = OVSM(osm00/out3)                        ! 70% void !

v0    = STAT(osm00,osd,ost,$pow)
v4    = STAT(osm40,osd,ost,$pow)
v7    = STAT(osm70,osd,ost,$pow)

T40   = TREE(B0/(v4)/15000/5, 35000/2)
T74   = TREE(B4/(v7)/15000/5, 35000/2)
T07   = TREE(B7/(v0)/15000/5, 35000/2)
```

DUMP(), THE RESTART-DUMP OPERATOR

Purpose

The DUMP operator is used to dump number densities and related data of regions (materials) in a HERMES data-base. This can be done at one or more burnup levels of a PATH operator. The regions are defined by an AREA operator. The DBMAT operator (see there) can access these data to construct material input to HELIOS in the form of "U" arrays that have the same format as those generated by the MAT operator. If possible, the DBMAT operator also generates a default material overlay—see OVLM operator.

Usage

name = **DUMP(** *df* / *ar*, *bname* **[** / *eb* [/ *n*] [, *eb* [/ *n*]] … **])**

name	represents a set of one or more isotopics to be dumped.
df	is the name of the data-base where the dumps are made.
ar	is the name of the area that defines which regions (materials) are to be dumped.
bname	is the name of the path along which dumps are to be made.
eb[/*n*]	specifies one [or *n*] burnups along *bname* where dumps must be made (MWd/t)—if not given, all the burnups are assumed.

Rules

1) The dump *name*, represents data sets for one or more regions (materials) at one or more burnups of the PATH operator *bname*. These data sets can be processed by the DBMAT operator (see there) to produce material input to HELIOS.
2) The name (identifier) of a dump, *name*, must not exceed 70 characters.
3) Different dumps may (but don't have to) be on different HERMES databases, *df*.
4) In the AREA operator *ar*, the *ss*, *st* and *cc* links of the chains that point at the regions must be sequence numbers, i.e. names or asterisks are not allowed. However, the *rg* link may be defined by number, name or asterisk. The construction ** is not allowed.
5) Regions specified by *ar* must be unique and must not be combined by <…>.
6) Dumps are only allowed from a PATH, not from a TREE.
7) If *eb* are specified, they must exist in the path *bname*. They must be in increasing order and lie within 0.001 MWd/t from the burnups of *bname*.

8) If no *eb* are specified, dumps are made at all burnups along *bname*. However, no dumps are made after an equilibrium step or a decay step.

9) If there is a state change in *bname*, then dumps are made for both states. Also, a dump is made at zero burnup.

10) Different dumps, with different *name*, may be made from the same state, on the same path, at the same burnup.

Remarks

In HELIOS, most branch-off calculations are part of the case in which the burnup calculations are done. They are based on the TREE operator and on the definition of different states, using the STAT, ISOX and OVS∗ operators (∗ = M, D or T). Examples of the flexibility these operators offer are: (i) replacing bypass-gap water by control-blade materials and *vice versa*; (ii) replacing burnt fuel in one or more fuel pins by fresh fuel; and (iii) start burnup with four fresh assemblies and replace successively the burnt assemblies by fresh ones.

This list of examples can be expanded with many more. However, some of them are somewhat artificial and not very computer-efficient. For instance, control branch-offs in a BWR require all calculations to be done with the control-blade mesh in the bypass gaps. This unnecessarily increases computational time for the other calculations.

Besides, calculations in cold and hot dimensions cannot be done in the same case. Of course, thermal expansion/contraction effects could be approximated by assigning density-correction factors to the various regions, using the OVLD and OVSD operators. These factors are the products of temperature differences and linear expansion coefficients of the regions. Otherwise, such effects must be found from different cases. This would require, amongst others, detailed number-density input for all the fuel regions at each burnup level to be examined. This is only feasible at zero burnup.

Also fuel storage calculations, where the fuel pins of an assembly are placed in a different geometry, would not be trivial. Again, the fuel number densities would somehow have to be entered by MAT operators.

To facilitate calculations in which one or more materials with a certain depletion are to be re-used, the DUMP and DBMAT operators have been introduced. Such calculations are called restart calculations because their probably most frequent application is that of "restarting" the calculations with the same fuel pins, though in a (completely) different geometry.

The DUMP operator stores the isotopics (number densities) of one or various materials at one or more burnup levels in a data-base. The initial heavy-metal density (g/cm^3) and the physical burnup of each material are also stored. The DBMAT operator retrieves these data and puts them as restart materials at the User's disposal. This is done by the input processor AURORA.

Note: Although the materials of all the regions can be dumped, it is advised to do this only with burnable materials. In a dump, and therefore in a later restart, each region represents one material. If the water of all the coolant regions were dumped and used in a later calculation, each coolant mesh (region) would get its individual material. In HELIOS, that creates as many scattering matrices as "restart" coolant materials, whereas normally all the coolant regions with the same water have a common scattering matrix.

Examples

```
1)  Pt1      = PATH(/(St1), 500/2, 1500, (St2), 2500)
    FuelDump = DUMP('dump.hrf'/FuelArea,Pt1/500,1500)
```

In this example, dumps are made from path *Pt1* at 500 and 1,500 MWd/t. The materials whose isotopics are dumped are those present in the positions pointed at by the AREA *FuelArea*. The dump identifier is *FuelDump* and the data are stored in the HERMES file *dump.hrf*. Two dumps are made at 1,500 MWd/t, one with the isotopics of state *St1*, the other with the isotopics of state *St2*.

```
2)  DumpA    = DUMP('A.hrf'/FuelArea, Pt1)
    DumpB    = DUMP('B.hrf'/FuelArea, Pt1)
```

In this example, dumps are made from path *Pt1* at all its burnups. The materials whose isotopics are dumped are those present in the positions pointed at by the AREA *FuelArea*. Two identical dumps, called *DumpA* and *DumpB*, are made in the files *A.hrf* and *B.hrf*.

```
3)  DumpNW   = DUMP('svea.hrf'/NWfuel, Pt1)!NW mini bundle!
    DumpNE   = DUMP('svea.hrf'/NEfuel, Pt1)!NE mini bundle!
    DumpSE   = DUMP('svea.hrf'/SEfuel, Pt1)!SE mini bundle!
    DumpSW   = DUMP('svea.hrf'/SWfuel, Pt1)!SW mini bundle!
```

In this example, the fuel number densities of the four mini bundles of a SVEA assembly are made from path *Pt1* at all its burnups. The AREAs are supposed to point at all the fuel regions in the four mini bundles. The dump identifiers are *DumpNW*, *DumpNE*, *DumpSE* and *DumpSW*; the dumps are all stored in the restart file *svea.hrf*.

FACE(), THE EDIT-PERIPHERY OPERATOR

Purpose

The FACE operator can be used to define a combination of peripheral segments of one or more space elements. For this segment combination, the current output specified by the CUR operator is saved in the data-base. This output may be used to evaluate discontinuity factors of homogenized fuel assemblies, which correct for homogenization errors in 3D reactor calculations.

Usage

name = **FACE (** (*fid*) **[**(*fid*)**]**… **)**

name represents an edit face, i.e. a combination of one or more not necessarily coherent peripheral segments of space elements.

fid is the identifier of one or more segments of the face *name*:

$$fid = [ss - [ss -] \ \dots] \, st, n, n$$

denotes two nodes (*n*) of one or more structures (*st*) at the end of a chain through the subsystems (*ss*)—if any.

Where:

ss is the name or sequence number of a subsystem in the preceding *ss* or in the system itself;

st is the name or sequence number of a structure in the preceding *ss* or in the system itself—a variant is

$$st = (st[-st] \ [, st[-st]] \ \dots);$$

n is the sequence number of a User-specified node in *st*.

Rules

1) The edit face *name* is a not necessarily coherent combination of peripheral segments of one or more space elements of the system. These segments may be interfaces between space elements and/or border segments of the system with any boundary condition.
2) The segments must be specified per structure as coherent peripheral segments between two nodes. When tracing the periphery from the first to the second node, the structure must lie on the right and no segment must have exact coupling, i.e., none must have $k = 0$.

   Observe that if an edit face has incoherent peripheral segments of the same structure, these can only be specified separately.
3) If the two nodes are equal, the entire periphery of the structure is specified.
4) No segment may occur more than once in an edit face, not even as the two sides of an interface (belonging once to one space element and once to its neighbour).

5) A set of coherent peripheral segments (*fid*) is given—in parentheses—by specifying one or more structures (*st*) by a chain whose links are connected by hyphens, followed by the node pair (*n*,*n*) of the begin and end points in *st*,

$$(fid) = ([ss-[ss-]\ldots]st, n, n).$$

6) The links represent the successive geometric <u>components</u> that contain the node pair(s). For each pair, the chain passes through zero, one or more subsystems (*ss*) and one structure (*st*).

7) Each link must represent one or more components. If each link has only one component, *fid* defines one node pair; if not, more pairs.

8) A component is specified by referring to its representative entry in the *s* list of the CNX operator that defines a component of the preceding link. The first link must refer to a component of the system itself.

9) A component may be specified by giving the sequence <u>number</u> (> 0) of its representative entry in the preceding operator. Alternatively, that number may be preceded by a colon and the component name.

10) One or more components may be specified by their common <u>name</u> in the preceding operator.

11) Structure specifications by number and/or name can be combined:

$$st = (st[-st]\ [,st[-st]]\ldots)$$

Each *st* inside the parentheses is the number or name of a structure; a range of one or more numbers may be given by $st_i - st_j$ with $st_i \le st_j$.

12) A node *n* must be specified by the sequence number in its structure. The node sequence is that of the *n*-list of the structure's STR operator, plus—in order of occurrence—that of possible (*x*,*y*) nodes defined in preceding CNX operators to couple the actual structure.

13) Nodes on an interface are common to the structures on both sides. If not specified in one of the structures, they have been added as a result of CNX-coupling. Such <u>anonymous nodes</u> are not User-specified and cannot be used by the FACE operator; they can still be accessed from the other structure.

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data-base (from where they can be retrieved by an output processor). The result of the FACE operator, *name*, can be considered as a homogenization of its segments. If *name* occurs in a CUR operator, the in and out-currents integrated over all its segments and angular sectors will be collapsed into a specified edit-group structure and saved in the data-base. Evidently, saving these currents involves considerably fewer data than saving all the currents of the system.

The system is discretized into space elements which are internally treated by collision probabilities and coupled by currents. Since collision probabilities are always used inside the structures defined by the STR operator, space elements smaller than a structure cannot exist. However, exact coupling, $k = 0$, can unite two or more structures into one space element (CNX operator, Rules 8 and 9 for coupling). Since no currents are used inside space elements, the FACE operator must specify only peripheral segments of space elements.

Although currents at specular border segments are not needed, they still are registered by HELIOS, so that specular segments are allowed in the FACE operator. At such segments, in- and out-currents are equal.

Examples

In the following examples, the 4×4 BWR mini-assembly of Fig. 1 is used; it is the same system as in Example 3 (and Fig. 4) of the CNX operator. For completeness, its geometric operators and the figure are repeated below. The pitch is 1 cm; the coolant film, box wall and outer gaps are 0.1, 0.2 and 0.3 cm thick.

```
Pin  = CCS(0.3,0.4 // fuel,clad)

Cell = STR((0,0)(0,1)(1,1)(1,0)/ 4,cool/ Pin,(0.5,0.5)/)

Gaps = STR((0,0)(-0.6,0.6)(4.6,0.6)(4,0) ...
            ! The rest is irrelevant; see CNX operator, Example 3 !

Row  = CNX(Cell,Cell,Cell,Cell/(1,3,4)0(2,2,1)/
                               (2,3,4)0(3,2,1)/
                               (3,3,4)0(4,2,1))      ! left-right !

Fuel = CNX(Row, Row, Row, Row /(1-1,2)(1-4,3)0(2-1,1)(2-4,4)/
                               (2-1,2)(2-4,3)0(3-1,1)(3-4,4)/
                               (3-1,2)(3-4,3)0(4-1,1)(4-4,4))

Rest = CNX(Gaps,Gaps,Gaps,Gaps/(1,3,4)1(2,2,1)              ! N+E !
                              /(2,3,4)1(3,2,1)              !  +S !
                              /(3,3,4)1(4,2,1))             !  +W !

BWR  = CNX(Fuel,Rest/(1-1-1,1)(1-4-4,3)1(2-3,4)(2-1,4))
```

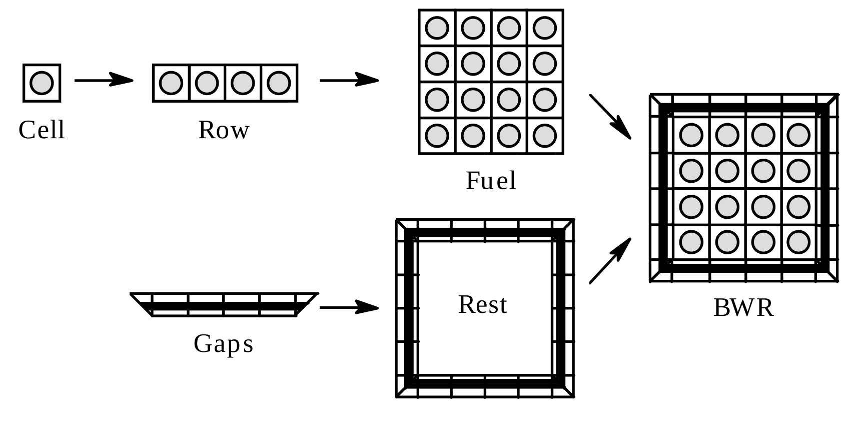


Fig. 1: The 4x4 BWR mini-assembly used in the FACE examples.

Notice the build-up of the subassemblies. *Row* is built up from left to right, *Fuel* from bottom to top, and *Rest* in clockwise direction starting in the North. Finally, the system *BWR* is constructed by coupling *Fuel* and *Rest* along their periphery from the SW to the NE corners. Further, *Fuel* is a space element, because all its structures *Cell* have exact coupling ($k = 0$). The other couplings, *Gaps* with *Gaps* and *Gaps* with *Fuel*, are by cosine currents ($k = 1$).

```
1)  N        = FACE((Rest-1,2,3))
             = FACE((   2-1,2,3))

2)  'N+S'    = FACE((Rest-(1,3),2,3))
             = FACE((   2-(1,3),2,3))
             = FACE((2-1,2,3)(2-3,2,3))

3)  'N/Fuel' = FACE((Rest-1,4,1))
             = FACE((   2-1,4,1))
             = FACE((Fuel-4-Cell  ,2,3))
             = FACE((   1-4-(1-4) ,2,3))
             = FACE((1-4-(1,4,2-3),2,3))

4)  'W/Fuel' = FACE((Rest-4,4,1))
             = FACE((   2-4,4,1))
             = FACE((Fuel-Row-1,1,2))
             = FACE((   1-Row-1,1,2))
             = FACE((1-1-1,1,2)(1-3-1,1,2)
                    (1-4-1,1,2)(1-2-1,1,2))

5)  'W/Fuel' = FACE((1-(1,3-4,2)-1,1,2))   ! error !

6)  Error    = FACE((Fuel-1-Cell,2,3))     ! error !
```

The first example defines the northern border of the system, which happens to be a single segment. The second example defines the northern and southern borders combined, although they are two incoherent segments. In this example, the two *Gaps*—the first and the third—are structures, which may be combined into (1, 3) or given individually.

The third example defines the interface between the subsystem *Fuel* and the northern (first) *Gaps*, or the interface between the subsystem *Rest* and the northern (fourth) *Row* of *Fuel*. This interface can be specified as the bottom segment of the first *Gaps*, as in the first two variants; or it can be specified as the top segments of the structures *Cell* of the fourth *Row* of *Fuel*, as in the last three variants. In the last two variants, the structures *Cell* of the fourth *Row* are given by sequence numbers.

The fourth example defines the interface between the subsystem *Fuel* and the western (fourth) *Gaps*. It can be specified as the right segment of the fourth *Gaps*, as in the first two variants; or as the left segments of the first structures *Cell* in all the four subsystems *Row*, as in the last three variants. In the last variant, the four subsystems *Row* are given by sequence numbers.

In the fifth example, representing the four subsystems *Row* by sequence numbers combined in parentheses is an error. This construction is allowed only at the level of the structures. The sixth example is wrong because the interface between the first and the second subsystem *Row* is not a peripheral segment of a space element—there is exact coupling.

ISOX(), THE ISOTOPIC-MULTIPLIER OPERATOR

Purpose

The ISOX operator is used to multiply number densities of specified material components in all the burnable and/or non-burnable materials. These multipliers can be used by the STAT operator to define a branch-off state. The system defined by such a state cannot undergo burnup, i.e., it may occur in a TREE operator, but not in a PATH operator.

Usage

name = **ISOX ([***ib*, *f* [; *ib*, *f*]...**] / [***in*, *f* [; *in*, *f*]...**])**

name	represents a combination of multipliers that can be applied to specified material components (*ib* or *in*) in all the burnable and/or non-burnable materials.
ib	is the library identifier of an isotope (B-10), an element (boron), a composite mixture (steel), a tabulated burnable absorber, or a special isotope (1/*v*-absorber) in the burnable materials.
in	is the library identifier of an isotope (B-10), an element (boron), a composite mixture (steel), a tabulated burnable absorber, or a special isotope (1/*v*-absorber) in the non-burnable materials.
f	is the isotopic multiplier of its *ib* or *in*; $0 \le f \le 1{,}000$.

Rules

1) The ISOX operator is optional; material components that have no isotopic multiplier get the default value of 1.0.
2) If a material component *ib* occurs in any burnable material, its density will be multiplied by its accompanying *f*. It does not affect the same material component in the non-burnable materials.
3) If a material component *in* occurs in any non-burnable material, its density will be multiplied by its accompanying *f*. It does not affect the same material component in the burnable materials.
4) The isotopic multipliers must lie in the range $0 \le f \le 1{,}000$.
5) The effect of an isotopic multiplier is unnatural and therefore final. A state that includes an ISOX operator cannot be used for burnup or decay calculations (see also Remarks, and the STAT, PATH and TREE operators.)

Remarks

The isotopic multipliers have been introduced to be able to selectively set certain isotopes to zero. At little extra cost, their number densities can be varied by an arbitrary factor, the isotopic multiplier *f*.

Number-density variations of material components due to the ISOX operator are unphysical. Therefore, this option cannot be used to define a state that occurs in a PATH operator. Nor can it be used in a state that undergoes decay in a TREE operator, not even if all the multipliers are ones. It can be used only in TREE operators for special purposes, in calculations that have the character of branch-off calculations. Burnup and decay of such states are still possible by defining a new material overlay set with the changed materials.

The ISOX operator does not have to be used to set isotopes to zero that disappear by radioactive decay, like Xe-135. The decay of such isotopes can always be handled by a decay step (see the PATH and TREE operators).

Examples

1) `Sm149B10 = ISOX(62149, 0 / 62149, 0; 5010, 2.0)`

2) `B10      = ISOX(/ 5010, 0)`

3) `Useless  = ISOX(62149, 1 / 5010, 1)`

In the first example, the Sm-149 concentration in all the materials is set to zero, while the B-10 contents in all the non-burnable materials is doubled. In the second example, the B-10 content in all the non-burnable materials is set to zero. The third example is useless indeed; it does not change anything. The User should be aware that although it leaves the number densities unchanged, a state that includes *Useless* cannot be subjected to decay calculations; nor may it occur in a PATH operator.

GROUP(), THE EDIT-GROUP OPERATOR

Purpose

The GROUP operator is used to define edit-group structures into which neutron and gamma output can be collapsed.

Usage

name = **GROUP (** NG **/ [***g* [, *g*]...**])**

name	represents an edit-group structure for neutrons or gammas.
NG	is either the letter N or the letter G, indicating that the group structure applies to neutrons or gammas.
g	defines the lower group limit(s) in eV, in decreasing order; if no *g* is specified, *name* represents all the groups.

Rules

1) An edit-group structure (*name*) is either for neutrons or for gammas, never for both.
2) The lower group limits (*g*) must be in eV and must be in decreasing order.
3) If *g* does not correspond to a library-group limit, HELIOS will shift *g* to the library-group limit that is nearest on a logarithmic scale; this is the true limit.
4) Regardless of Rule 3, the true limit of the last group will coincide with that of the highest library group (with the lowest energy, usually close to zero).
5) Each *g* must define a unique true group limit.
6) Apart from the group structure, the B_1 diffusion coefficients for the entire homogenized system in that structure are saved in the data base. (See the MACRO operator, Remarks.)

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data base (from where they can be retrieved by an output processor). With the GROUP operator various group structures can be defined that have fewer groups than the library. This allows, for example, certain data to be saved in one group and others in two groups only. Of course, afterwards these data cannot be expanded into more groups.

Examples

```
1)  'N-gr.1' = GROUP(N/12),   or, more consistent:  'N-gr.1'=GROUP(N/0)
2)  'N-gr.2' = GROUP(N/0.625, 0.0)
3)  'G-gr.2' = GROUP(G/10000, 0)
4)  AllG     = GROUP(G/ )
```

In the first example, '*N-gr.1*' denotes a single neutron group, which represents the entire energy range. According to Rule 4, the true lower limit is that of the library—for instance 10^{-5} eV—even though 12 eV or 0 eV have been specified. The second example defines a classical two-group structure for neutrons, and the third example defines a two-group structure for gammas. The fourth example, finally, defines all the groups of the gamma library.

MACRO(), THE PROPERTY-EDIT OPERATOR

Purpose

The MACRO operator is used to define non-isotopic properties of edit regions defined by an AREA operator, in an edit-group structure defined by a GROUP operator. These data are saved in the data-base for later output processing.

Usage

name = **MACRO (** [T/] *gr*, *ar* **/ [** *dt* [, *dt*] ...**])**

name	represents a macro, i.e. one or more sets of property data for the edit region(s) of a given area in a given edit-group structure.
T	is the letter T to save also time-averaged data.
gr	is the name of an edit-group structure.
ar	is the name of an area, i.e. one or more edit regions.
dt	denotes one of the data types of the Table below; fluxes are always saved, while volumes are saved with the other AREA-operator data.

bu	burnup (1 group)	*n*
dn	delayed-neutron data	*n*
ed	kinetic energy deposited by scattering (and absorption for γ's) (1 group)	n, γ
tr[a]	transport XS[b], $\Sigma_g = \Sigma_{ag} + \Sigma_{0,g} - \Sigma_{1,g}$	n, γ
ab	absorption XS, Σ_{ag}	n, γ
fi	fission XS, Σ_{fg}	*n*
nf	fission-neutron production XS, $\nu\Sigma_{fg}$	*n*
kf	fission-energy production XS, $\kappa\Sigma_{fg}$	*n*
ch	fission spectrum, χ_g	*n*
*p*0	P_0-scattering matrix, $\Sigma_{0,g'\leftarrow g}$[c]	n, γ
*p*1	P_1-scattering matrix, $\Sigma_{1,g'\leftarrow g}$	*n*
*p*2	P_2-scattering matrix, $\Sigma_{2,g'\leftarrow g}$	*n*
*p*3	P_3-scattering matrix, $\Sigma_{3,g'\leftarrow g}$	*n*
ng(*gr*)[d]	(n,γ) production matrix	γ

[a] These flux-weighted values occur twice

[b] XS denotes cross section

[c] Transport-corrected for γ's

[d] *n*-group structure appended in parentheses, "(*gr*)"

Rules

1) The macro *name* specifies property data of none, one, or more of the data types (*dt*) of the Table above. These data are for the edit region(s) of a given area (*ar*), in a given edit-group structure (*gr*).

2) The edit-group structure *gr* defines whether a macro is for neutrons (*n*) or gammas (γ) — see the GROUP operator.

3) The fluxes are always saved, also if no *dt* is specified, while the volumes have been saved in connection with the AREA operator.

4) Data types that, according to the Table, do not exist for gammas must not be specified in a macro for gammas.

5) The data types burnup (*bu*) and energy deposited (*ed*) can only be specified in a macro with a one-group structure.

6) Time-averaged data may be saved by specifying a T and a slash in the beginning of the operator. Averaging is over the preceding burnup step in the PATH operator. The use of the predictor/corrector mode is required.

7) Time-averaged fluxes are always saved if time-averaged data are requested. These data can be used by the output processor to evaluate time-integrated (historic) data. For gammas this requires that gamma calculations be specified at all burnups.

8) In the case of burnup, instead of time-averaged values, the burnup steps are saved. (Burnup is also energy-integrated; see Rule 5.)

9) Burnup (megawatt-days per metric ton initial heavy metal) is the physical burnup of the materials actually present in the edit regions of *ar*. For other data types, time-averaged values refer to the edit regions, not to the history of their actual materials; see Remarks.

10) The delayed-neutron data type (*dn*) creates $\nu\Sigma_f\phi$-averaged delayed-neutron yields, as well as the data types *ab*, *nf*, *ch*, *p*0 and *p*1. From these data, effective delayed-neutron yields can be produced by the output processor; see Remarks.

11) The energy deposited (*ed*) is the kinetic-energy loss of particles due to elastic and/or inelastic scattering. For gammas, their kinetic energy when being absorbed is included. The logarithmic midpoints of the library groups define the kinetic energies.

12) (n,γ) production is a neutronic reaction. It may only be specified in a macro for gammas. The emitting neutron edit-group structure must be appended in parentheses, "(*gr*)".

13) XSs and scattering matrices are flux-weighted. Leakage-weighted transport XSs (*tr*) and P_1-scattering matrices (*p*1) were also evaluated, where leakages were found from the balance for the edit regions. Since version 1.7 leakage-

averaging has been abandoned, but the flux-averaged *tr* data are still saved twice.

14) HELIOS does not save diffusion coefficients. They can be evaluated by the output processor according to one of several definitions; see Remarks.

15) For gammas, the P_0-scattering matrix (*p0*) is transport-corrected. In this correction, the self-scattering (diagonal) elements of the matrix have been reduced by the P_1-scattering XSs.

16) The data specified by a macro are saved for all the calculational (reactivity) points of the PATH and TREE operators.

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data-base (from where they can be retrieved by an output processor). Often, the MACRO operator will be used to save just a few data, for instance: Σ_g, Σ_{ag}, Σ_{fg}, $\nu\Sigma_{fg}$, $\kappa\Sigma_{fg}$ and $\Sigma_{0,g'\leftarrow g}$ in two groups for a homogenized fuel assembly; and *bu* and $\kappa\Sigma_f$ in one group for the burnup and power maps in all the fuel pins.

It should be emphasized that, apart from burnup, the data saved are those of the regions; see Rule 9. This is of importance when considering historic (time-integrated) data. Thus, if a material is moved from one region to another, its total irradiated flux does not follow it.

While HELIOS can calculate systems of more than one fuel assembly, delayed-neutron yields are often needed for just a single assembly. It is therefore essential to be able to obtain delayed-neutron yields for a part of the system. The data that the output processor needs to evaluate these yields are χ_g, Σ_{ag}, $\nu\Sigma_{fg}$, $\Sigma_{0,g'\leftarrow g}$, $\Sigma_{1,g'\leftarrow g}$ and $<\beta_{m,g}>$, the $\nu\Sigma_f\phi$-averaged delayed-neutron yields. To save these data in the data-base, the data type *dn* should be specified in a multigroup macro that represents the assembly. One of the group limits should be near 0.45 MeV, the delayed-neutron fission source. HELIOS's predecessors, PHOENIX and RECORD, used a default of ten groups with a 0.5 MeV cut-off after the third group and five groups with a 0.8 MeV cut-off after the first group, respectively.

There are many definitions of the diffusion coefficient, of which two will be discussed. One is the trivial $D = 1/3\Sigma$, where Σ is the transport-corrected total XS. Theoretically, the current-weighted Σ should be used, not the flux-weighted Σ. Since currents are not everywhere available, leakages obtained from the balance in each region were used. Since version 1.7, this option is no longer available.

Another definition, valid only for the entire homogenized system, is that of the B_1 diffusion coefficient. It is a byproduct of the B_1 calculation of the criticality spectrum, from where it is obtained as $D_{B_1} = J/B\phi$. Here, B, J and ϕ are the material buckling, the absolute net current and the flux of the homogenized system made critical by buckling leakage. The ratios of D_{B_1} to the trivial D have been saved in connection with the GROUP operator. These ratios can be used to adjust the trivial D's for parts of the system so that they—when leakage-volume weighted—reproduce the D_{B_1}'s of the system.

For gammas, the absorption XS and the transport-corrected P_0-scattering matrix are defined to include pair production, namely:

$$\Sigma_{ag} = \Sigma_g^{PE} - \Sigma_g^{PP} ; \tag{1}$$

$$\Sigma_{0,g'\leftarrow g} = \Sigma_{0,g'\leftarrow g}^{C} + 2\delta_{g'g_p}\Sigma_g^{PP} . \tag{2}$$

The superscripts *PE*, *PP* and *C* denote photoelectric effect (true absorption), pair production and Compton (true) scattering, respectively. In pair production an electron-positron pair is created from a high-energy gamma, whereupon the positron annihilates with an electron producing two gammas of about 0.511 MeV each. The subscript g denotes the gamma group, where g_p is the group around 0.511 MeV in which the gamma pair appears. In this way, HELIOS does not have to treat pair production explicitly. Notice that these definitions preserve both the total XS and the number of secondaries per collision:

$$\Sigma_g = \Sigma_g^{PE} + \Sigma_g^{PP} + \Sigma_g^{C} = \Sigma_{ag} + \Sigma_{sg} \tag{3}$$

and

$$c_g = \frac{\Sigma_g^{C} + 2\Sigma_g^{PP}}{\Sigma_g} = \frac{\Sigma_{sg}}{\Sigma_g} , \tag{4}$$

where

$$\Sigma_{sg} = \sum_{g'} \Sigma_{0,g'\leftarrow g} . \tag{5}$$

One application of gamma data is to evaluate the gamma-detector signal from the deposited energy *ed*, or the flux in a number of groups together with group-sensitivity factors of the detector (to be provided to the output processor). Another application is to evaluate the effect of gamma smearing on the power map from the deposited energy.

Examples

```
1) 'XS-ng2' = MACRO(T/ ng2,Assy/ab,fi,nf,kf,tr,p0)
2) Pinmaps  = MACRO(ng1, Pins /bu,kf)
3) Gamdet   = MACRO(gg8, Gdet /)
4) Neudet   = MACRO(ng6, Ndet /fi)
5) Totalgam = MACRO(gg1, All  /ed)
```

In these examples, *Assy*, *Pins*, *Gdet*, *Ndet* and *All* are different sets of edit regions. They represent an entire fuel assembly, all its individual pins, a gamma and a neutron detector, and all the individual regions of the assembly, respectively. Further, *ng*# and *gg*# denote neutron and gamma-group structures in # groups.

The first example causes HELIOS to save all the data needed to construct standard two-group XSs for the assembly. Time-averaged values of all the data types will also be saved. These may be used by the output processor to obtain, for instance, time-integrated two-group absorption and fission-neutron production. In the second example, the information needed to produce burnup and power maps will be saved. In the third example, eight-group fluxes will be saved, from which the output processor can evaluate a gamma-detector signal. The fourth example causes six-group fission XSs to be saved for the evaluation of the fission rate in a neutron detector. In the fifth example, finally, the gamma energy deposited in all the regions will be saved.

MAT(), THE MATERIAL OPERATOR

Purpose

The MAT operator is used to specify material compositions, regardless of whether they will occur in the system or not. However, for restart materials that have been dumped in a data-base, the DBMAT operator must be used. The OVLM operator assigns these materials to the geometric regions of the system.

Usage

name = **MAT (** [NB**/**] [*d*] **/** *i*, *c* [,*f*] **[**; *i*, *c* [,*f*] **]** ... **)**

name	represents the material specified here.
NB	is the A2-word *NB*, specifying that the material is not burnable.
d	is the density in g/cm^3, [0]: number-density input; >0: material density (weight % input); <0: heavy-metal density at zero burnup (number-density input).
i	is the library identifier of an isotope (B-10), an element (e.g. boron), a composite mixture (e.g. steel), or a tabulated burnable absorber.
c	is the number density of *i* in 10^{24}/cm^3, or weight percentage of *i*.
f	is the remaining fraction of a tabulated burnable absorber.

Rules

1) There are three categories of burnable components:
   (i) Fuel and fission-product isotopes of the burnup chains.
   (ii) Isotopes with strong absorption (sometimes also their predecessors/successors in an absorption-decay chain), such as B-10, B-11, and certain Gd, Hf, Ag, In, and Cd isotopes.
   (iii) Tabulated burnable absorbers (BAs) with library tables of shielded absorption cross-sections as a function of their number densities. Their identifiers are 991** for Gd_2O_3, and 992** for arbitrary BA chains.
2) A burnable material contains at least one burnable component.
3) The indicator NB affects only burnable materials. If NB is specified for such a material, it will not deplete with time.
4) If the density is positive, $d > 0$, it refers to the entire material and weight-percentage (wt%) input must be specified for all material components *i*. The wt%'s do not have to sum to 100, but must be non-negative with at least one positive.

5) If the density $d \leq 0$, number-density input must be specified for all material components *i*. The option $d < 0$ allows the specification of a restart material and its heavy-metal density at zero burnup, $|d|$.

6) Oxygen in fuel-oxide materials may have a zero wt%. The wt% input of the heavy isotopes must then refer to the aggregate of these isotopes. Oxygen is added according to Σ(heavy isotopes)O_2, plus the oxygen of Gd_2O_3 for BA of type 991** (see Remarks).

7) In Rule 6, the wt% of the non-heavy isotopes, BA included, must be with respect to the density of the entire material.

8) For <u>striped BA</u> discs (Fig. 1), density and number-density input must be axially averaged, i.e.

$$x = (L_{BA}x_{BA} + L_{fuel}x_{fuel}) / (L_{BA} + L_{fuel}) \qquad (1)$$

The wt% input then must be recalculated as the total BA weight divided by the averaged density.

9) The oxygen in BA of type 991** (Gd_2O_3) must not be included in the wt% of the oxygen in the rest of the material.

10) For BA of type 992**, isotopes not represented by their library tables must have their wt%'s explicitly specified. If oxygen is among them, the option of zero wt% oxygen (Rule 6) is allowed, though not recommended, as it neglects the BA's oxygen.

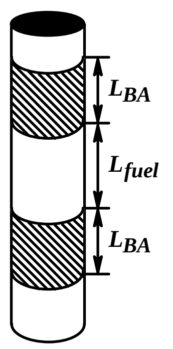


Fig. 1 Striped BA.

11) The BA fraction *f* is defined as the ratio of the BA number density in the input composition to that of fresh BA (see Remarks). It must lie in the range $0 \leq f \leq 1$. For BA in a fresh material, $f = 1$; in a restart (depleted) material $f < 1$.

12) The value of *f* does not affect the BA concentration. It merely states how much of the original absorptive power is left.

13) If $f < 1$, wt% input must not be used (see Remarks).

14) If a burnable isotope is specified, all its related isotopes—if any—are created too, i.e., all the isotopes of the chain in which it occurs and all its fission products. If these related isotopes have not been specified by the User, they will have zero number densities.

15) In the wt% input, gadolinia (Gd_2O_3), dysprosia (Dy_2O_3) and erbia (Er_2O_3) may be specified by *64000*, *66000* and *68000*—identifiers that do not exist in the library. For Gd_2O_3 this adds seven Gd isotopes (152, 154, 155, 156, 157, 158 and 160) and the oxygen to the material's isotopes. For Dy_2O_3 this adds five Dy isotopes (160 through 164) and the oxygen to the material's isotopes. For Er_2O_3 this adds the four Er isotopes (166, 167, 168 and 170) and the oxygen to the material's isotopes.

16) If gadolinia/dysprosia/erbia is specified by *64000/66000/68000* then individual Gd/Dy/Er isotopes nor BA may be specified for the same material.
17) If gadolinia/dysprosia/erbia is specified by *64000/66000/68000* and the rest of the material has no oxygen, then a negligible amount of oxygen must be given—not zero. Of course, the gadolinia/dysprosia/erbia isotopes may also be specified individually.
18) Natural zirconium as resonance absorber can be specified as 40010.

Remarks

Number densities

HELIOS needs only number densities. Hence, number-density input is used directly, without any further processing.

On the other hand, if wt% input is given, it has to be converted into number densities. Using instead of d the more common symbol ρ for the material density, the basic formula to convert the wt% c_i of component i into its number density N_i (in units of $10^{24}/\text{cm}^3$) is

$$N_i = 0.01 c_i \frac{\rho}{A_i} N_{Avo} \ , \tag{2}$$

where A_i is the atomic weight of i in atomic mass units and N_{Avo} is the Avogadro number in units of 10^{24} nuclei per mole,

$$1 \text{ amu} = 1.6603 \times 10^{-24} \text{ g} ; \quad N_{Avo} = 0.6023 \ . \tag{3}$$

If the oxygen is specified with zero wt%, there are three possible component types: oxygen (indexed by O), heavy isotopes and the other components. The formulas used for their number densities are:

$$N_i = 0.01 c_i \frac{\rho}{A_i} N_{Avo} \qquad i \notin \text{heavy isotopes} \ , \tag{4}$$

$$N_i = 0.01 c_i \frac{\rho_h}{A_i} N_{Avo} \qquad i \in \text{heavy isotopes} \ , \tag{5}$$

where the density of the aggregate of heavy isotopes, ρ_h, is given by

$$\rho_h = \frac{\left(1 - 0.01 \sum_{i \notin heavy} c_i\right)\rho}{1 + 0.02 A_O \sum_{i \in heavy} c_i / A_i} \ . \tag{6}$$

Finally, the number density of the oxygen is (see below for 991∗∗)

$$N_O = 2 \sum_{i \in heavy} N_i + 4.9180 \sum_{i \in 991**} N_i \ . \qquad (7)$$

Burnable absorbers

There are two types of tabulated burnable absorber (BA), with the library identifiers 991** and 992**. The original BA, which is of type 991**, represents only the two strongly absorbing isotopes of Gd_2O_3, Gd-155 and Gd-157. Because no other Gd-isotopes are treated, the residual Gd-absorption is not represented in the tables of shielded absorption cross-sections. However, the BA of type 992** can represent arbitrary chains containing one or more strongly absorbing isotopes.

For both types of BA, the interpolation parameter in the tables is the fraction f that is left of the BA number density. At time t it is defined as

$$f(t) = \frac{N_{BA}(t)}{N_{BA}(0)} \ , \qquad (8)$$

where $N_{BA}(t)$ is the number density at time t. It is defined as the number of absorptions still available. With this number density, the interpolated shielded cross-section gives the correct macroscopic absorption cross-section,

$$\hat{\Sigma}_{BA}(t) = N_{BA}(t)\hat{\sigma}_{BA}(f(t)) \ . \qquad (9)$$

BA of type 991** represents only the isotopes Gd-155 and Gd-157, so $N_{BA}(t) = N_{155}(t) + N_{157}(t)$. Once this BA is burnt out, $f(t) = 0$.

For BA of type 992**, $N_{BA}(t)$ represents the number of absorptions still available, i.e. all the chain locations the nuclei can assume during the life of the BA. Thus, the chain

$$N_1(t) \to N_2(t) \to N_3(t) \qquad (10)$$

is represented by the cumulative number density

$$N_{BA}(t) = 3N_1(t) + 2N_2(t) + N_3(t) \ . \qquad (11)$$

If such a BA chain ends on a non-burnable isotope, the "burnt-out" BA has a non-zero pseudo-equilibrium value $f(t) > 0$. This represents the residual absorption, which will very slowly approach zero if the last isotope has a finite absorption cross-section.

To evaluate number densities from wt% input, the atomic masses of the BAs in the library have been adjusted. BA of type 991** represents the two isotopes Gd-155 and Gd-157 in gadolinia. Because their natural abundances are 0.148 and 0.157, there are 0.610 BA nuclei per Gd_2O_3 molecule. They must represent the total atomic weight of the gadolinia molecule, 362.475 amu. Hence, the weight assigned to a BA component of type 991∗∗ is 594.221 amu.[1] (For striped BA, HELIOS cannot use the same library as PHOENIX, since PHOENIX expects also axially adjusted atomic masses.)

BA of type 992∗∗ represents an arbitrary chain of *I* isotopes *i* with the natural abundances a_i. The atomic weight that reproduces the correct initial BA number density is given by

$$A_{992**} = \frac{\sum a_i}{\sum (I - i + 1) a_i} \sum a_i A_i \ , \tag{12}$$

where the summations are over $i = 1, \ldots I$. The numerator and denominator are proportional to the true and cumulative number densities of the chain isotopes combined, while the factor on the right is their true initial atomic weight.

It should be observed that the adjusted atomic weights can only be used with fresh BA, which explains Rule 13 for detailed input.

Zirconium as resonance absorber

Since HELIOS-1.8, the library contains two natural Zr isotopes instead of one: natural Zr without and with resonance data, 40000 and 40010 respectively. This is the case only for natural Zr, not Zircaloy. To treat the natural Zr in Zircaloy as a resonance absorber (and scatterer), one must specify its composition per element. Its library (40001) composition is: 98.23 wt% Zr, 1.50 wt% Sn, 0.12 wt% Fe, 0.10 wt% Cr, and 0.05 wt% Ni.

In the case of coolant with homogenized spacers that contain Zr, it is recommended not to treat the Zr as resonance absorber. That is because HELIOS assigns to each region with a resonance absorber its own material—resonance shielding is space-dependent. In the case of the coolant this would create, unnecessarily, a considerable amount of materials with relatively large P_0 and P_1 scattering matrices. Since the homogenized Zr is almost infinitely diluted, its resonance shielding is negligible and a resonance treatment is not needed.

Examples

```
1)  Zircaloy = MAT( 6.55/40002, 100)

2)  Zircaloy = MAT(  NB//40002, 4.3248E-2)

3)  'UO2-2.8'= MAT( 9.86/92235, 2.8;   92238, 97.2;  8001, 0)

4)  'UO2-2.8'= MAT( 9.86/92235, 2.468; 92238, 85.68; 8001, 11.85)

5)  'UO2-2.8'= MAT(     /92235, 6.2333E-4; 92238, 0.021365;
                         8001, 0.043977     ! 8001: oxygen in UO2 !)

6)  'UO2+BA' = MAT( 9.65/92235, 4;  92238, 96;  8001, 0; 99179, 7, 1)

7)  'UO2+Gd' = MAT( 9.65/92235, 4;  92238, 96;  8001, 0; 64000, 7)
```

[1]The BA tables in the library are based on the abundances 0.1473 and 0.1568, so the weight assigned to a BA component in the library still is 596.021 amu.

```
8)  'UO2-B2' = MAT(-9.17/92235, 4.1966E-4; 92236, 9.2267E-6; 92238, 2.2690E-2;
                        93237, 7.6390E-8; 94238, 2.7421E-9; 94239, 3.1422E-5;
                        94240, 1.8256E-6; 94241, 2.7729E-7; 94242, 5.6938E-9;
                        95241, 9.046E-10; 95242, 9.190E-12; 95243, 1.009E-10;
                        96242, 2.796E-11; 96244, 1.842E-12; 36083, 2.3926E-7;
                        45103, 1.7386E-6; 45105, 1.7215E-8; 47109, 7.4869E-8;
                        54131, 1.3916E-6; 54135, 0;         55133, 3.2662E-6;
                        55134, 3.5373E-8; 55135, 9.9184E-7; 60143, 2.7424E-6;
                        60145, 1.8715E-6; 61147, 1.0153E-6; 61148, 4.1629E-9;
                        61148, 7.5555E-9; 62147, 3.7444E-8; 62149, 4.9785E-8;
                        62150, 5.3265E-7; 62151, 1.3152E-7; 62152, 2.4773E-7;
                        63153, 1.1140E-7; 63154, 6.1913E-9; 63155, 1.9300E-8;
                        99001, 6.6038E-5; 99002, 1.4147E-5;  8001, 4.6402E-2;
                        99123, 3.2460E-4, 0.655)
```

Examples 1 and 2 specify the canning material Zircaloy, once by wt% and once by number densities. The indicator *NB* is superfluous.

Examples 3, 4 and 5 describe the same 2.8 wt% enriched UO_2. The third example uses Rule 6 for the detailed wt% input; the fourth example does not use this rule; and the fifth example uses number densities as input.

Example 6 shows wt% input for fuel with 7 wt% BA, using Rule 6 for detailed input. The wt% of the BA would not have changed if this rule had not been used.

Example 7 is the same as example 6, except that the Gd_2O_3 will be treated explicitly, and not by interpolation in BA tables. This is generally the recommended method for representing burnable absorbers.

Example 8 shows the unwieldy input of a restart material. It is for 2 wt% enriched UO_2 with admixed BA at a burnup of 2 MWd/kg. Here, isotope identifiers of the very first HELIOS library are used, with two lumped fission products 99001 and 99002. Such input should rather be retrieved from a data-base with restart compositions, using the DBMAT operator.

MICRO(), THE ISOTOPIC-EDIT OPERATOR

Purpose

The MICRO operator is used to define isotopic data for edit regions defined by an AREA operator, in an edit-group structure defined by a GROUP operator. These data are saved in the data-base for later output processing.

Usage

name = **MICRO (** [T/] *gr*, *ar* **/** **[** *i* [, *i*] … **]** **/** **[** *dt* [, *dt*] … **])**

name	represents a micro, i.e. one or more sets of isotopic data for the edit region(s) of a given area in a given edit-group structure.
T	is the letter T to save also time-averaged data.
gr	is the name of an edit-group structure.
ar	is the name of an area, i.e. one or more edit regions.
i	is the library identifier of an isotope (B-10), an element (boron), a composite mixture (steel), a tabulated burnable absorber, or a special "isotope" (1/*v*-absorber): no *i* - data for all the isotopes present in the edit regions.
dt	denotes a data type of the Table below; initial heavy-isotope densities, fluxes and number densities (at least 10^{-20}/barn×cm) are always saved, while volumes are saved with the other AREA-operator data.

ed	kinetic energy deposited by scattering (and absorption for γ's) (1 group)	n,γ
tr[a]	transport XS[b], $\sigma_g = \sigma_{ag} + \sigma_{0,g} - \sigma_{1,g}$	n,γ
ab	absorption XS, σ_{ag}	n,γ
fi	fission XS, σ_{fg}	n
nf	fission-neutron production XS, $\nu\sigma_{fg}$	n
kf	fission-energy production XS, $\kappa\sigma_{fg}$	n
*p*0	P_0-scattering matrix, $\sigma_{0,g'\leftarrow g}$[c]	n,γ
*p*1	P_1-scattering matrix, $\sigma_{1,g'\leftarrow g}$	n
*p*2	P_2-scattering matrix, $\sigma_{2,g'\leftarrow g}$	n
*p*3	P_3-scattering matrix, $\sigma_{3,g'\leftarrow g}$	n
ng(*gr*)[d]	(n,γ) production matrix	γ

[a] These flux-weighted values occur twice
[b] XS denotes cross section
[c] Transport-corrected for γ's
[d] n-group structure appended in parentheses, "(*gr*)"

Rules

1) The micro *name* specifies isotopic data of none, one, or more of the data types (*dt*) of the Table above. These data are for the specified "isotopes" (*i*)—see also Rule 3—in the edit region(s) of a given area (*ar*), in a given edit-group structure (*gr*).

2) Each "isotope" has a unique library identifier (*i*), which denotes: a true isotope like B-10, an element like natural boron, a composite mixture like steel, a tabulated burnable absorber, or a special "isotope" like the 1/*v*-absorber.

3) All the "isotopes" present in the edit regions can be specified by not giving any "isotope" at all—see also Remarks.

4) If an "isotope" is given by *i*, its specified data types will be saved for all the edit regions.

5) "Isotopes" that do not exist in one or more of the regions that make up an edit region, get a "zero" number density of 10^{-20}/barn×cm in these regions. This can be used to obtain reaction rates of <u>detector isotopes</u> in very thin foils or pins. Such isotopes do not exist in the edit region that represents the detector, i.e., they do not influence the flux calculation.

6) The edit-group structure *gr* defines whether a micro is for neutrons (*n*) or gammas (γ)—see the GROUP operator.

7) Initial heavy-isotope densities, fluxes and number densities of the isotopes, are always saved for the edit regions of the area. Number densities of detector isotopes will be saved as 10^{-20}/barn×cm. (Volumes have been saved in connection with the AREA operator.)

8) Data types that, according to the Table above, do not exist for gammas must not be specified in a micro for gammas.

9) Energy deposited (*ed*) may only be requested in a micro based on a one-group structure. For gammas, when *ed* is requested, the data types *ab* and *p0* are also created (only for MICROs, not MACROs).

10) Time-averaged data may be saved by specifying a T and a slash in the beginning of the operator. They are averaged over the preceding burnup step in the PATH operator. The use of the predictor/corrector mode is required.

11) Time-averaged fluxes and isotopic densities are always saved if time-averaged data are requested. These data can be used by the output processor to evaluate time-integrated (historic) data. For gammas this requires that gamma calculations be specified at all burnups.

12) The energy deposited (*ed*) is the kinetic-energy loss of particles from elastic and/or inelastic scattering. For gammas, their kinetic energy when being absorbed is included. The logarithmic midpoints of the library groups define the kinetic energies.

13) (n,γ) production is a neutronic reaction. It may only be specified in a micro for gammas. The emitting neutron edit-group structure must be appended in parentheses, "(*gr*)".

14) XSs and scattering matrices are flux-weighted. Leakage-weighted transport XSs (*tr*) and P_1-scattering matrices (*p*1) were also evaluated, where leakages were found from the balance for the edit regions. Since version 1.7 leakage-averaging has been abandoned, but the flux-averaged *tr* data are still saved twice.

15) For gammas, the P_0-scattering matrix (*p0*) is transport-corrected. In this correction, the self-scattering (diagonal) elements of the matrix have been reduced by the P_1-scattering XSs.

16) The data specified by a micro are saved for all the calculational (reactivity) points of the PATH and TREE operators.

Remarks

Output operators do not influence the HELIOS calculations. Their purpose is to limit the amount of output data saved by HELIOS in the data-base (from where they can be retrieved by an output processor). Often, the MICRO operator will be used to save just a few data - for instance, the assembly-averaged number density and thermal absorption XS of Xe-135, selected isotopic number densities for the pins, and one-group microscopic XSs of burnable absorbers to obtain their reaction rates.

The fluxes in the edit regions and edit groups, and the number densities of the specified non-detector isotopes in the edit regions, are always saved. So by giving no "isotope" at all between the slashes of the operator, the number densities of all the existing "isotopes" are saved (Rule 3).

Existing "isotopes" are "isotopes" that occur in any edit region of the area *ar*. So if *ar* consists of one gap and one fuel region, then data of all fuel and water "isotopes" are saved for both regions. Moreover, if in any of the states any of these regions contains another material, say a control rod, then data of its "isotopes" will be saved, too.

For gammas, the absorption XS and the transport-corrected P_0-scattering matrix are defined to include pair production, namely:

$$\sigma_{ag} = \sigma_g^{PE} - \sigma_g^{PP}, \tag{1}$$

$$\sigma_{0,g'\leftarrow g} = \sigma_{0,g'\leftarrow g}^{C} + 2\delta_{g'g_p}\sigma_g^{PP}. \tag{2}$$

The superscripts *PE*, *PP* and *C* denote photoelectric effect (true absorption), pair production and Compton (true) scattering, respectively. In pair production an electron-positron pair is created from a high-energy gamma, whereupon the positron annihilates with an electron producing two gammas of about 0.511 MeV each. The subscript g denotes the gamma group, where g_p is the group around 0.511 MeV in which the gamma pair appears. In this way, HELIOS does not have to treat pair production explicitly. Observe that these definitions preserve both the total XS and the number of secondaries per collision:

$$\sigma_g = \sigma_g^{PE} + \sigma_g^{PP} + \sigma_g^{C} = \sigma_{ag} + \sigma_{sg} \tag{3}$$

and

$$c_g = \frac{\sigma_g^{C} + 2\sigma_g^{PP}}{\sigma_g} = \frac{\sigma_{sg}}{\sigma_g}, \tag{4}$$

where

$$\sigma_{sg} = \sum_{g'} \sigma_{0,g'\leftarrow g} \,. \tag{5}$$

Examples

```
1) Xedat   = MICRO( T/ ng2, Assy / 54135 / ab )
2) Ndmaps  = MICRO( ng1, Pins / 92235, 94239, 94241 / ab, fi )
3) Allregs = MICRO( ng1, All / / )
```

In these examples, *Assy*, *Pins* and *All* are sets of edit regions. They represent the fuel assembly, all its individual pins, and all its individual regions, respectively. Further, *ng*# denotes an edit-group structure for neutrons in # groups.

In the first example, two-group absorption XSs of Xe-135 for the entire assembly will be saved. The assembly-averaged number density of Xe-135 will also be saved, together with the two-group fluxes and the initial heavy-isotope density. Time-averaged values of these quantities, except the heavy-isotope density, will be saved, too.

The second example causes one-group absorption and fission XSs of U-235, Pu-239 and Pu-241 to be saved for all the pins. Also initial heavy-isotope densities, one-group fluxes, and number densities of the specified "isotopes" will be saved. The densities can be used to produce weight-percentage maps of the specified isotopes.

The third example causes initial heavy-isotope densities, one-group fluxes and number densities (of all the existing "isotopes") to be saved for all the regions of the system. Had *Allregs* been defined to obtain only number densities, any group structure would have done. Specifying the entire energy range (*ng*1) is in that case the most economical, because it reduces the amount of flux data that are automatically saved to one group only.

NEWK(), THE COUPLING-MODIFICATION OPERATOR

Purpose

The NEWK operator can be used to modify the angular representation of interface currents that was earlier defined by the CNX (and NEWK) operators. For example, the User may improve the interface coupling of a selected pin cell with its neighbours without having to change the standard build-up the fuel assembly from structures and subsystems.

Usage

name = **NEWK (** *k***/**(*c*, *n*, *n*) [(*c*, *n*, *n*)]… **[/***k***/**(*c*, *n*, *n*) [(*c*, *n*, *n*)]…**]**… **)**

name	represents a (sub)system, earlier defined by a CNX operator and perhaps already modified by earlier NEWK operators.	
k	specifies an angular representation of interface currents.	
c	specifies a structure in *name* by its chain,	$c = [l–]\ldots l$
n	specifies a node in that structure,	$n = n_i$

Where:

l	is a link along the chain that defines the structure—the last link—whose nodes *n* are used to define the interface section whose angular coupling gets modified to *k*;
n_i	is the sequence number of a User-specified node in the STR operator of that structure.

Rules

1) While the CNX operator defines a (sub)system by geometric and interface coupling, the NEWK operator redefines part or all of the interface coupling. The result is a (sub)system of the same name, which replaces the old one in all later—not earlier—references.

2) The new coupling *k* must be specified for one peripheral section of a structure at a time. Since both sides of an interface have the same coupling, several structures may be involved on the other side.

3) A peripheral section is made up of one or more coherent line segments. When tracing a peripheral section from begin to end point, the structure in which it is defined must lie on the right.

4) A peripheral section is specified by the location *c* of the structure to which it belongs, and by the node locations *n* of its begin and end point in that structure, enclosed in parentheses, i.e. (*c*, *n*, *n*).

5) If begin and end point are equal, the entire periphery of the structure is specified.
6) The location of the structure is specified by a chain *c* whose links *l* are connected by hyphens. The links represent all the successive subsystems that contain the structure, and the structure itself.
7) Each link must refer to an entry in the *s*-list of the CNX operator that defines the preceding link. The first link must refer to an entry in the *s*-list of the actual (sub)system. Only the last link can and must be a structure.
8) Each link must be specified as the sequence number of the subsystem or the structure in its defining *s*-list. The User may also use the name of the subsystem or the structure, immediately followed by a colon and its sequence number.
9) A node location *n* is given by the sequence number n_i in its structure. The node sequence is that of the *n*-list of the structure's STR operator, plus—in order of occurrence—that of possible (*x*,*y*) nodes defined in preceding CNX operators to couple the actual structure.
10) Nodes on an interface are common to the structures on both sides. If not specified in one of the structures, HELIOS has added them as a result of the CNX-coupling. Such "anonymous" nodes are not User-specified and cannot be used by the NEWK operator; they can still be accessed from the other structure.
11) Modified *k*-values assigned to a part of the periphery that is not an interface have no effect whatever.
12) A NEWK operator must at least be preceded by the CNX operator to which it refers, and may be followed by more CNX operators.

Remarks

The NEWK operator is used to selectively modify the angular interface coupling in an earlier defined (sub)system without changing its name. It acts on the peripheral segments of the structures in that (sub)system. This operator provides the User with the flexibility to vary the coupling across the individual segments of each structure of a (sub)system.

Rule 1 describes the cumulative character of the NEWK operator. While all the modifications in a (sub)system can be collected in a single NEWK operator, they also may be specified successively, one or a few at a time, as shown in the Examples.

It should be remarked that even an earlier fusion (exact coupling) can be changed into current coupling, i.e. $k=0$ may be changed into any $k\neq0$. The opposite is also possible, provided that Rule 9 for coupling in the CNX operator is observed.

Examples

```
    sub = CNX(A, B, C / (1,5,3)1(2,4,(1.6,0)) / (1,3,4)1(3,3,2) )
1)  sub = NEWK(-3 / (2, 3,4) )
2)  sub = NEWK(-3 / (2, 3,7) (2, 7,4) )
3)  sub = NEWK(-3 / (2, 3,7) / -3 / (2, 7,4) )
4)  sub = NEWK(-3 / (2, 3,7) )
    sub = NEWK(-3 / (2, 7,4) )
5)  sub = NEWK(-3 / (2, 3,7) (1, 5,3) )
6)  sub = NEWK( 3 / (B:2, 2,2) )
7)  sub = NEWK(-3 / (1, 3,3) (3, 3,3) )
    sub = NEWK( 1 / (1, 3,4) )
```

In this example, a subsystem *sub* is considered, which consists of the structures *A*, *B* and *C*, as shown in Fig. 1. The CNX operator that defines *sub* is given first. It follows from Fig. 1 that node 5 of *A* must have been specified in the *s*-list of *A* = STR(…) and that the *s*-list of *B* must contain six nodes. However, node 7 is specified in the CNX operator and does not belong to *B* but rather to the structure *B*:2 as part of *sub*. In this structure, it is the seventh node.

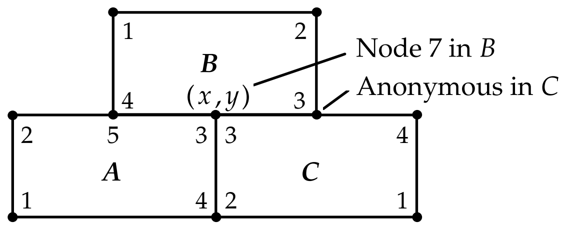


Fig. 1: Example to illustrate the use of NEWK( ).

Originally, the interface coupling between the three structures was everywhere $k = 1$. In the examples, this has been changed to $k = -3$ for the common interface of *B* with *A* and *C*. Observe the variety of ways in which the *k* modification can be specified. For instance, Examples 4 and 7 illustrate the cumulative character of the NEWK operator. In Examples 6 and 7 entire peripheries are used, which, according to Rule 11, affects only the interfaces. In Example 7, after changing the entire peripheries of *A* and *C* to $k = -3$, the interface coupling between *A* and *C* was restored to $k = 1$. Finally, apart from Example 6, sequential numbers have been used for the one-link chains. In Example 6, the combination "name:sequence number" has been used.

OVLD(), THE DENSITY-OVERLAY OPERATOR
OVLT(), THE TEMPERATURE-OVERLAY OPERATOR

Purpose

The OVLD and OVLT operators are used to assign correction factors for material densities (*D*) and temperatures (*T*) to one or more regions of the system. The OVSD and OVST operators can combine one or more of these overlays into overlay sets to assign these properties to the entire system.

Usage

name = **OVL∗(** $dt/pid[,pid]\ldots$ **[** $/dt/pid[,pid]...$ **]** $\ldots$ **)** ∗ = **D** or **T**

name	represents an overlay in which one or more *D* or *T* values are each assigned to one or more regions of the system.
dt	is a real number that represents the value of the property *D*, or *T* (Kelvin) — $0 \le D \le 1{,}000$ and $276 \le T \le 5{,}000$.
pid	is a position identifier that refers directly or indirectly to one or more regions.

Direct *pid* references

$pid = [ss-[ss-]\ldots]\,st-cc-rg$

denotes one or more regions (*rg*) as the last link of a chain that passes through the subsystems (*ss*) — if any — the structures (*st*) and the CCS-es (*cc*; for non-CCS regions $cc = 0$).

Variants:	$x = (x[-x]\ [,x[-x]]\ldots)$	$x = st$, *cc* or *rg*;
	$ss-[ss-]\ldots = *$	all subsystems;
	$cc-rg = **$	all regions of *st*;
	$x = *$	all *x* of preceding link.

Indirect *pid* references

$pid = ov-dt$ **[** $(p[-p]\ [,p[-p]]\ldots)$ **]**

refers to one or more locations in the *pid* list of a property value (*dt*) of an earlier-defined overlay (*ov*) of the same property type.

Rules in general

1) An overlay is a property configuration that fills a part (or all) of the system; for density factors and temperatures they are defined by the OVLD and OVLT operator, respectively.
2) An overlay set is a combination of overlays that fills all of the system; for the density factors and temperatures they are defined by the OVSD and OVST operators, respectively (see there).

3) The density or temperature identifier (*dt*) must be the value of the density factor ($0 \le D \le 1{,}000$) or the temperature ($276\,\mathrm{K} \le \mathrm{T} \le 5{,}000\,\mathrm{K}$).
4) The same property value (*dt*) may appear more than once inside an OVL∗ operator (as opposed to the OVLM operator, where each material identifier must be unique). Still, each *dt* may have more than one *pid* in its *pid* list.
5) The *pid*'s in the *pid* list of any *dt* must not specify any region more than once.
6) Density factors and temperatures must always be assigned. So if the entire system has the density factor of 1.0, then this density factor must be assigned to all its regions.

Rules for direct pid referencing (Rules 2-13 of the OVLM operator)

1) The position identifier (*pid*) used for direct referencing specifies one or more regions by a chain, whose links are connected by hyphens,

$$pid = [\,ss - [\,ss -]\,\ldots\,]\,st - cc - rg.$$

2) The links represent the successive geometric <u>components</u> that contain the region(s). For each region, the chain passes through zero, one or more subsystems (*ss*), one structure (*st*), and one CCS (*cc*), to the region (*rg*). For non-CCS regions, $cc = 0$.
3) Each link may represent one or more components. If each link has only one component, *pid* defines one region; if not, more regions.
4) A component is specified by referring to its corresponding entry in an operator that defines a component of the preceding link. The first link must refer to a component of the system itself.
5) The *ss* and *st* links must refer to entries in the *s* lists of the CNX operators that define the subsystems of the preceding *ss* links.
6) The *cc* link must refer to one or more CCS-es (*cname*) in the STR operator(s) that define the structure(s) of the preceding link.
7) The *rg* link must refer to one or more CCS regions (*cid*) in the CCS operator(s) that define the CCS(-es) of the *cc* link, or to one or more non-CCS regions in the STR operator(s) that define the structure(s) of the *st* link (their order is: first *bid*, then the *rid*'s, if any).
8) A component may be specified by giving the sequence <u>number</u> of its representative entry in the preceding operator. Alternatively, that number can be preceded by a colon and the component name. For partial CCS-es, observe CCS Rule 5 of the STR operator.
9) One or more components may be specified by their common <u>name</u> in the preceding operator.
10) All the components of a certain type may be specified by an <u>asterisk</u>.
    (i) A double asterisk (∗∗) denotes all the regions of a st.

(ii) An asterisk denoting all the ss's (there must be at least one ss) is the first link of all chains, even those without a ss.

(iii) Thus, all the regions of a system with at least one ss in its CNX operator are given by $*-*-**$, otherwise by $*-**$.

(iv) The construction $*$–st number– ... must not be used.

11) Specifications by number and/or name can be combined as follows:

$$x = (x[-x]\ [,x[-x]]\dots) \qquad \text{with } x = st,\ cc \text{ or } rg.$$

Each x inside the parentheses is a component number or name; a range of one or more numbers may be given by $x_i - x_j$ with $x_i \le x_j$.

12) Rule 11 defines multiple *pid*'s, each being one specification by number or name. If this construction occurs in more than one link, the *pid*'s are ordered such that *rg* varies fastest and *st* slowest.

Rules for indirect *pid* referencing

1) The position identifier (*pid*) used for indirect referencing must refer to the *pid* of a property (*dt*) of the same type—*D* or *T*—in an earlier-defined overlay (*ov*). This leads to the construction

$$pid = ov - dt\ \mathbf{[}\,(p[-p]\ [,p[-p]]\ \dots)\,\mathbf{]}.$$

2) If *pid* is indirect, *ov* must be specified by name and *dt* must be specified as the sequence number of one of the *dt*'s in *ov*.

3) Each *p* must be the sequence number of a *pid* in the *pid* list of *dt*, where multiple *pid*'s must be counted according to Rule 12 above.

4) A range of one or more *p*'s may be specified by $p_i - p_j$ with $p_i \le p_j$. If no *p*'s are specified, the entire *pid* list of *dt* is used.

5) The *p*'s must only refer to direct *pid*'s.

6) By representing a *pid*, each *p* specifies ultimately either one region by number, or one or more regions by name or asterisk (see Rules 8-10 for direct referencing).

Remarks

Because indirect references to *D* and *T* are meaningless, the density and temperature overlays are much simpler than the material overlays, as can be seen in the examples below.

The main application areas of the density overlay are:

(i) Void branch-off calculations—see Example 1.

(ii) Coolant-density variation across an assembly.

(iii) Thermal-expansion/contraction effects (approximately).

Thermal-expansion/contraction effects can be dealt with exactly by specifying the geometric system with its new dimensions and the materials with their new densities. However, the next best—and certainly the easiest—way would be to keep the geometric dimensions, and to account only for the decreased/increased number of the nuclei in each region. The density-correction factors are then the products of the linear expansion coefficients of the materials in the regions and the temperature differences.

Generally, standard fuel-assembly calculations require very few different density factors. For PWRs the trivial value of 1.0 might be the only one, with BWRs requiring an extra value to describe the void in the coolant. Therefore, the density overlays are straightforward.

Further, three temperatures are usually enough—for fuel, clad and the rest. Startup, shutdown and cold shutdown-margin calculations often require only one temperature for the entire system. Thus, also the temperature overlays are simple, unless temperature profiles inside fuel pins, or pin-to-pin fuel-temperature variations are studied.

Examples

1)
```
ov30 = OVLD(  1/*-*-**/  0.7/*-*-0-cool/  0.8/*-*-0-film)
ov50 = OVLD(0.5/ov30-2/ 0.65/ov30-3)
```

These could be the overlays for void branch-off calculations in a BWR assembly at 30% and 50% absolute void (coolant densities of 70% and 50%). Observe that the coolant film between regular lattice and flow-box wall has a higher density of 80% and 65%, respectively. Further, it is assumed that the coolant regions of all the pin cells have been called *cool*—although there may be four per cell—and all the coolant-film regions *film*. Finally, a single overlay is enough to assign density factors to the entire system. Of course, this overlay may be split up into more overlays.

2) This is the example of a small 3×3 PWR fuel assembly, shown in Fig. 1. All pins have the same geometry and the same fuel material in the *fuel* region(s)—except the NE-pin which has water instead of fuel. The system has been built up, from bottom to top, of three rows, which have been built up, from left to right, of three pin cells.

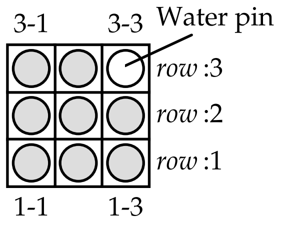


Fig.1: 3x3 fuel assembly with a water pin.

At operating conditions, let the fuel have the (Doppler) temperature of 950 K, the fuel cladding 560 K and the rest 530 K. Overlay *temp1* assigns 950 K to all the regions, then overwrites all the *clad* regions by 560 K and, finally, the non-CCS regions by 530 K. Because all the pins have the same generic CCS—same geometry and region names—the water in the pin and its cladding have now also 950 K and 560 K, instead of 530 K. This is remedied by the second overlay, *temp2*, of which three variants are shown.

```
temp1 = OVLT(950/*-*-**/      ! 950K to all regions            !
             560/*-*-*-clad/  ! 560K to clad regions           !
             530/*-*-0-*)     ! 530K to non-CCS regions        !
```

```
temp2 = OVLT(530/3-3-**)       ! Direct pid, entire cell        !
temp2 = OVLT(530/3-3-1-*)      ! Direct pid, entire pin         !
temp2 = OVLT(530/3-3-*-*)      ! Direct pid, all pins of cell   !
                               ! (3-3, which has only one pin)  !
```

The same overlays that have been shown above can also be used for a BWR assembly. The last member of *temp1* assigns 530 K to all the non-CCS regions, which would include coolant film, flow-box wall and outer gaps. A possible cruciform control element, geometrically described by the structure *CR*, would require a third overlay, *temp3* (at least for the absorbing pins). Three variants are given. In the first, *CR* appears in the *s* list of the CNX operator defining the system; in the other two, *CR* itself is part of a subsystem (*ss*).

```
temp3 = OVLT(530/CR-**)        ! Direct, CR starts the chain    !
temp3 = OVLT(530/*-CR-**)      ! Direct, CR part of a subsystem !
temp3 = OVLT(530/ss-CR-**)     ! Direct, CR part of subsystem ss !
```

To assign room temperature to the entire system, a fourth overlay, *temp4*, is defined. Below, two variants are shown: the first uses direct referencing, the obvious choice; the second is just an illustration of indirect referencing, not the obvious choice.

```
temp4 = OVLT(293/*-*-**)       ! Direct pid, the entire system   !
temp4 = OVLT(293/temp1-1)      ! Indirect pid, the entire system !
```

OVLM(), THE MATERIAL-OVERLAY OPERATOR

Purpose

The OVLM operator is used to assign materials to one or more regions of the system. The OVSM operator can combine one or more material overlays into an overlay set to assign materials to the entire system.

Usage

$name$ = **OVLM (** $mid\,/\,pid\,[,pid]\ldots$ **[** $/\,mid\,/\,pid\,[,pid]\ldots$ **]** $\ldots$ **)**

name	represents an assignment of materials to regions of the system.
mid	is a material identifier that refers directly to one or more materials, or indirectly through their regions.
pid	is a position identifier that refers directly or indirectly to one or more regions.

Direct references

$mid = m$, dbm, or 'dbm-i' [−'dbm-j'] with $i < j$

is a material defined by the name of a MAT operator, or one or more materials defined by the name of a DBMAT operator with specified burnup, or a range of one or more such materials.

$pid = [ss - [ss -]\ldots]\,st - cc - rg$

denotes one or more regions (rg) as the last link of a chain that passes through the subsystems (ss)—if any—the structures (st) and the CCS-es (cc; for non-CCS regions $cc = 0$).

Variants:	$x = (x[*x]\ [,x[-x]]\ldots)$	$x = st$, cc or rg;
	$ss*[ss*]\ldots = *$	all subsystems;
	$cc * rg = **$	all regions of st;
	$x = *$	all x of preceding link.

Indirect references

$mid = os - ov - M$ **[**$(p[-p]\ [,\,p[-p]]\ \ldots)$**]** with $M = m$, dbm, or 'dbm-i'

refers to one or more materials by specifying the regions to which one or more materials (M) have been assigned in the overlay (ov) of the overlay set (os). These regions are specified by certain *pid*'s in M's *pid*-list, where each *pid* is denoted by its location (p) in that *pid*-list.

$pid = ov - M$ **[**$(p[-p]\ [,p[-p]]\ldots)$**]**

refers to one or more locations in the *pid*-list of a directly defined material (M) in an earlier-defined overlay (ov).

$pid = 0$ is the null movement, the materials keep their positions.

Rules in general

1) A material overlay is a material configuration that fills part (or all) of the system; it is defined by the OVLM operator.
2) A material-overlay set is a combination of overlays that fills all of the system; it is defined by the OVSM operator (see there).

   Although the geometry of the system is fixed, the same regions can be filled by several overlay sets: Control-rod regions with control-rod or gap materials, fuel assemblies with identical or different enrichment distributions, etc.
3) Each material identifier inside an OVLM operator must be unique, but may have more than one *pid* in its *pid*-list.
4) In an overlay, no material may overwrite itself in any region. Different materials, however, may overwrite each other.
5) Direct and indirect referencing may be mixed, i.e. any type of *mid* (direct or indirect) may be combined with any or both types of *pid*.

Rules for materials

1) A material can be defined in one of three ways:

   (i) The name of a MAT operator (*m*) defines a <u>generic material</u>, which can be assigned to more than one region.

   (ii) The name of a DBMAT operator with specified burnup (*dbm*) represents $NRA \geq 1$ <u>restart materials</u>, which can be used in three ways:
   - The name refers to all the restart materials, *dbm*.
   - The name followed by a hyphen and a sequence number $i \leq NRA$ refers to the *i*'th restart material, '*dbm-i*'.
   - The names of the *i*'th and *j*'th restart material connected by a hyphen refers to the range of restart materials *i*, *i*+1,...*j*, with $i < j \leq NRA$, '*dbm-i*'–'*dbm-j*',.

   (iii) An <u>indirectly-referenced material</u> is defined by specifying the region to which a generic or restart material has been assigned (see Rule 2 and Rules for indirect referencing).

   Restart and indirectly-referenced materials are unique—inside an OVLM operator, each can be assigned to one region only.[1]
2) A true material is a generic or a single restart material with specified burnup, assigned to a region of the system. It is defined by its <u>first occurrence</u>, i.e., the moment a material-overlay set is specified (see OVSM) that contains a material overlay in which it has been <u>directly referenced</u> by its name *M*.

   Observe that if the above overlay occurs in more overlay sets, then not one but more true materials are defined. See example 3 of the TREE operator.

[1] Such a material can be assigned to more than one region by combining different OVLMs into one material-overlay set (see OVSM operator).

3) True materials based on the same burnable generic or restart material, will change their compositions with burnup and become different from the original material and from each other. ("Burnable" is defined in Rules 1 and 2 for detailed input of the MAT operator.)
4) Because of Rule 3, each next occurrence of a burnable true material must be specified by indirect referencing to its first occurrence.
5) Next occurrences of a non-burnable true material may be specified by direct referencing to a generic or restart material, or by indirect referencing to a true material.
6) Indirect referencing allows the User to keep track of burnable materials and move them between overlay sets, e.g. to interchange used fuel pins, or to insert a used control rod in another fuel assembly.
7) Because a true material is unique, it can be assigned to only one region in an overlay set. Though unphysical, this region may differ in shape and/or volume from the region where it was defined.

Rules for direct referencing

1) The material identifier (*mid*) for direct referencing represents one or more materials. It is one of the following:
   (i) The name of a generic material, *m*.
   (ii) The name of a set of restart materials with specified burnup, *dbm*.
   (iii) The name of a single restart material with specified burnup, '*dbm-i*'.
   (iv) A range of restart materials with specified burnup, '*dbm-i*' –'*dbm-j*'.
2) The position identifier (*pid*) used for direct referencing must specify at least one region by a chain whose links are connected by hyphens,

$$pid = [\,ss - [\,ss -]\ \ldots\ ]\,st - cc - rg.$$

3) The links represent the successive geometric components that contain the region(s). For each region, the chain passes through zero, one or more subsystems (*ss*), one structure (*st*), and zero or one CCS (*cc*), to the region (*rg*). For non-CCS regions, $cc = 0$.
4) Each link must represent one or more components. If each link has only one component, *pid* defines one region; if not, more regions.
5) A component is specified by referring to its representative entry in an operator that defines a component of the preceding link. The first link must refer to a component of the system itself.
6) The *ss* and *st* links must refer to entries in the *s* lists of the CNX operators that define the subsystems of the preceding *ss* links.
7) The *cc* link must refer to one or more CCS-es (*cname*) in the STR operator(s) that define the structure(s) of the preceding link.

8) The *rg* link must refer to one or more CCS regions (*cid*) in the CCS operator(s) that define the CCS(-es) of the *cc* link, or to one or more non-CCS regions in the STR operator(s) that define the structure(s) of the *st* link (their sequence is: first *bid*, then the *rid*'s, if any).

9) A component may be specified by giving the sequence <u>number</u> (>0) of its representative entry in the preceding operator. Alternatively, that number may be preceded by a colon and the component name. For partial CCS-es, observe CCS Rule 5 of the STR operator.

10) One or more components may be specified by their common <u>name</u> in the preceding operator.

11) All the components of a certain type may be specified by an <u>asterisk</u>.
    - (i) A double asterisk ($**$) denotes all the regions of a *st*.
    - (ii) An asterisk denoting all the *ss*'s (there must be at least one *ss*) is the first link of all chains, even those without a *ss*.
    - (iii) Thus, all the regions of a system with at least one *ss* in its CNX operator are given by $*{-}*{-}**$, otherwise by $*-**$.
    - (iv) The construction $*-st$ number– … must not be used.

12) Specifications by number and/or name can be combined as follows:

    $x = (x[-x]\ [,x[-x]] \ldots)$ with $x = st$, cc or rg.

    Each x inside the parentheses is a component number or name; a range of one or more numbers may be given by $x_i - x_j$ with $x_i \le x_j$.

13) Rule 12 defines multiple *pid*'s, each being a specification by number or name. If this construction occurs in more than one link, the *pid*'s are ordered such that *rg* varies fastest and *st* slowest.

14) For restart materials, the *ss*, *st* and *cc* components must be specified as sequence numbers, which may be combined according to Rule 12.

Rules for indirect referencing

1) An indirect material identifier (*mid*) refers to one or more true materials. This involves an overlay set (*os*), an overlay (*ov*), a generic material, a single restart material, or all the restart materials of a DBMAT operator with specified burnup (*M*) and one or more regions, i.e.,

    $$mid = os - ov - M\,\mathbf{[}\,(p[-p]\ [,p[-p]] \ldots)\,\mathbf{]}.$$

2) An indirect position identifier (*pid*) must refer to a *pid* in an earlier-defined overlay (*ov*), i.e.,

    $$pid = ov - M\,\mathbf{[}\,(p[-p]\ [,p[-p]] \ldots)\,\mathbf{]}.$$

3) In *mid* and *pid*: *M*, *ov* and *os* must be specified by name (*p* must be a number); *M* must occur in *ov*, *ov* in *os*, while *os* must occur in a state (see STAT operator).

4) In *mid* and *pid*: each *p* must be the sequence number of a *pid* in the *pid*-list of *M*, where multiple *pid*'s must be counted according to Rule 13 above.
5) In *mid* and *pid*: multiple *p*'s may be specified as follows:

   $p = (p[-p]\ [,p[-p]] \dots)$,

   where a range of one or more *p*'s may be given by $p_i - p_j$ with $p_i \leq p_j$. If no *p*'s are specified, the entire *pid*-list of material *M* is used.
6) In *mid*: the *p*'s may refer to direct and indirect *pid*'s.
7) In *pid*: the *p*'s must only refer to direct *pid*'s.
8) By representing a *pid*, each *p* specifies ultimately either one region by number, or one or more regions by name or asterisk (see Rules 9-11 for direct referencing).
9) If *p* specifies more than one region, their specification sequence in HELIOS is not known. Because each true material corresponds to exactly one region (Rule 4 for materials), indirect material referencing must be avoided if more than one region is involved.
10) Rule 9 has two exceptions:
    (i) Indirect referencing in *mid* may always be combined with the null movement of its materials by specifying *pid* = 0.
    (ii) Indirect referencing in *mid* may always be combined with a parallel movement of its materials, i.e. ,the *pid*'s of the original and final positions must have link-by-link duplicate (same name) components.

Remarks

Overlays and overlay sets

The OVLM operator assigns materials to regions, not necessarily filling the entire system. To assign materials to the entire system, one or more material overlays must be combined into an overlay set by the OVSM operator (see there). This concept allows the major parts of a system to be defined as separate overlays, whose regions do not have to be coherent. Fig. 1 shows examples for a very small BWR assembly.

The material configuration of the uncontrolled assembly (*Overlay Set 1*) consists of *Overlay 1* and *Overlay 2*. In the top row, these overlays are the frame of all box and gap regions, and the interior of all pin-cell regions, respectively. In the bottom row, *Overlay 1* fills all the gap, box, clad and coolant regions, while *Overlay 2* fills the fuel regions. The control-rod regions, shown as dashed contours, are filled also with gap material. In the controlled assembly (*Overlay Set 2*) it is *Overlay 3* that assigns control-rod materials to the control-rod regions.

Overlays may overlap. In the controlled assembly, *Overlay 3* overlaps part of *Overlay 1*. In fact, the fuel regions could have been included in *Overlay 1*, i.e. filled with coolant, to be replaced by the correct materials by *Overlay 2*. This would have made the definitions of *Overlay 1* as simple as:

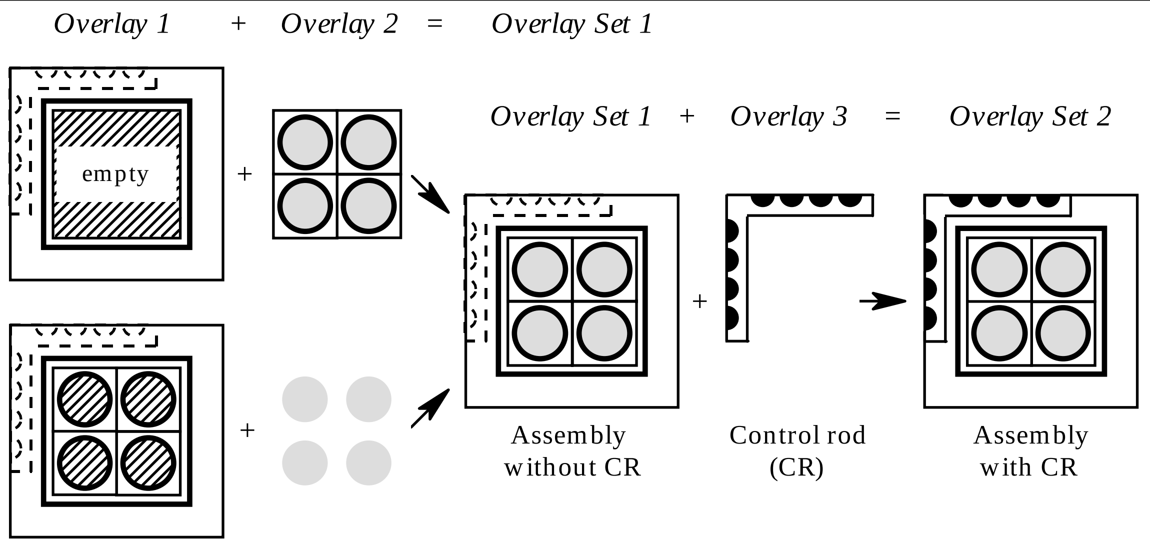

Overlay 1 + Overlay 2 = Overlay Set 1
Overlay Set 1 + Overlay 3 = Overlay Set 2
empty
+
+
+
Assembly without CR
Control rod (CR)
Assembly with CR

Fig. 1: Making up a mini BWR assembly from its major parts.

'*Overlay 1*' = OVLM(*H2O* / ∗−∗−∗∗/ *Zr* / ∗−∗−0−*box*) and

'*Overlay 1*' = OVLM(*H2O* / ∗−∗−∗∗/ *Zr* / ∗−∗−∗−*clad*, ∗−∗−0−*box*).

Here it is assumed that all the coolant and gap regions contain *H2O*. Further, all the box regions should have the name *box*, and all the clad regions *clad*; and both should contain the same material, *Zr*.

Materials

A material participates in the calculations if:

(i) it is specified in an overlay where it is not overwritten;

(ii) the overlay occurs in an overlay set, where the other overlays do not overwrite it—this makes it a true material;

(iii) the overlay set is part of a state, i.e., a system that can be calculated by HELIOS (see STAT operator);

(iv) the state is used by a PATH or a TREE operator (see there).

Such materials can burn if they are burnable and used by a PATH operator. (The TREE operator for the branch-off calculations does not involve burnup, e.g., a control rod that is momentarily inserted at different times to determine its worth does not burn.)

An overlay set has as many true materials as regions, but it requires usually far fewer generic materials to fill them. If these materials burn, their compositions change with time, so each generic material becomes as many different materials as regions to which it has been assigned. If such a material occurs in other overlay sets, it no longer can be described by its generic (or restart) material alone.

To use such a material in more than one overlay set, indirect referencing has been introduced. This provides the additional information on what makes that material unique (true), by specifying the region in the overlay set to which it was assigned for the first time. The indirect referencing of materials that do not burn—burnable or not—is allowed, though not necessary.

For reasons of computer efficiency, HELIOS does not use all the true materials, but rather those materials—the H materials—that are needed in the calculations. A generic material only then becomes more than one H material if (a) it occurs in more than one region of an overlay set, and (b) at least one of the following conditions is fulfilled:

(i) it is burnable—its composition change depends on the region (if time is involved);

(ii) it contains resonance isotopes—its Dancoff factor depends on the region;

(iii) the region temperatures differ more than DTMIN (20K at present)—its scattering matrix depends on the region.

Thus, although coolant and cladding occur in many regions, they are often represented by one H material each. Notice also that condition (i) causes permanent material changes with time, while the other two conditions merely cause changes during the calculations. That is why indirect referencing is required only for materials that have burnt.

Indirect referencing

Indirect referencing offers the possibility to use burnt materials in more than one overlay set. An example is branch-off calculations with a control rod. First, the burnup calculation is done in an overlay set without control-rod materials; next, branch-off calculations are done in an overlay set with control-rod materials. Other examples are the interchanging of used fuel pins, or inserting a used control rod into another fuel assembly; both also lead to other overlay sets.

Indirect referencing can be complicated. However, it is not needed for the common burnup and branch-off calculations of reload-fuel design—not even to replace used fuel pins by fresh ones. This is due to the concept of a background overlay set (see the OVSM operator, and then example 3 of the TREE operator).

Multiple material assignments to any region in the system are possible. The general Rule 4 allows it for overlays, while the overlays in an overlay set may also overlap. In fact, the background overlay set in the OVSM operator is clearly based on overwriting. The last material assigned to a region is the one that counts, i.e., the true material. This is the material that is specified by indirect referencing.

Indirect referencing specifies regions in an overlay set. The referenced materials are the true materials in those regions. The generic or restart materials *M* used in the indirect reference merely indicate a *pid* list in an overlay—the *p*'s pointing at certain entries of that list. The User should be aware that *M* may have been overwritten and does not correspond at all to the true materials. Although it violates Rule 1, this *M* may still be used provided that the true materials in these regions have been assigned by direct referencing; if not, it is an error. To avoid misinterpretations, it is recommended to use *M* of the true materials.

Inactive overlays in indirect referencing

An overlay that does not appear in any overlay set is inactive; it is not part of a system to be calculated by HELIOS. Still, such an overlay may be defined. Its directly-defined *pid*'s (Rule 7), so long as they belong to a generic material name *m* (Rule 3), may be used to define other indirect *pid*'s (Rule 2). However, the materials in such an overlay are not true materials and cannot be used to define an indirect *mid* (Rule 1).

Default restart material overlay

When defining *NRA* restart materials by *dbm* = DBMAT(…), a default OVLM operator may also be created, called '*dbm–db*' (see DBMAT operator). This operator assigns the *NRA* restart materials to the same *pid*'s as those from where they were dumped in the original case. Such an overlay can be useful in restart calculations in which all or most of the fuel pins of an assembly are placed in another geometric system. It must exist for a DBMAT with unspecified burnup; it may exist for a DBMAT with specified burnup—(see the DBMAT operator).

Example

Here, the use of the OVLM operator is illustrated in the overlay sets *A1*, *A*2 and *A3* of Fig. 2. Each of these overlay sets fills the same geometric system of four pin cells. The cells are duplicates of the generic structure *cell*, with *cool* as only non-CCS region; the generic name of the CCS is *pin*, and its two regions are *fuel* and *clad*. The top row of Fig. 2 shows how the system has been built up, knowledge that is necessary when specifying the chains for direct referencing of the position identifiers, *pid*.

The generic materials that are involved are *H2O*, *Zr* and four different fuels: *a*, *b*, *c* and *d*. Of the fuels, *a* and *b* appear in two pins, which is indicated by the affixes *1* and *2*.

Table 1 lists some examples of direct referencing of some of the regions—not all of the possibilities are given. Obviously, the referencing is the same for each of the overlay sets, because they refer to the same geometric system.

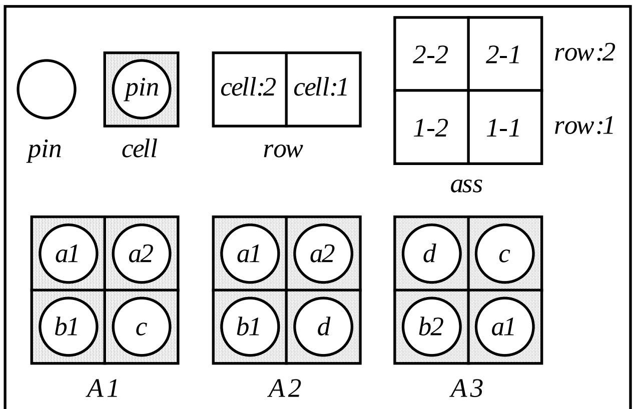


Fig. 2: Overlay sets to illustrate the use of the OVLM operator.

In this example, there is only one generic structure (*cell*), which has one CCS (*pin*) and one non-CCS region (*cool*). Therefore, the following references to a CCS in the *cc* link are equivalent: 1, *pin* and ∗. For the same reason, when referring to a non-CCS region, the following chain endings (*cc*−*rg*) are equivalent: 0−∗, 0−*cool* and 0−1.

Table 1. Direct referencing of regions in the geometric system of Fig. 2.

Region	Some possible position identifiers, *pid*		
NW-coolant	*row*:2−*cell*:2−0−*cool*	2−2−0−∗	2−2−0−1
All N-coolant	*row*:2−*cell*−0−*cool*	2−*cell*−0−∗ 2−∗−0−∗	2−(1,2)−0−1
All clad	*row*−*cell*−*pin*−*clad*	∗−*cell*−∗−2 ∗−∗−*pin*−2	∗−∗−∗−*clad* ∗−∗−1−2
SE-fuel	*row*:1−*cell*:1−*pin*−*fuel*	1−1−∗−*fuel*	1−1−1−1

Table 2 shows examples of overlays that make up the three overlay sets. Because *A1* is defined first, it cannot have indirect material referencing. Assuming *Zr* non-burnable, one might say that *A2* is the same assembly as *A1*, but with another SE pin. In *A3*: the NW and SE-pins of *A2* are interchanged, the SE pin of *A1* appears in the NE cell, and the SW pin is another than the one in *A2*.

The example does not show how the overlay sets are made up from the overlays. This is postponed until Example 1 of the OVSM operator, where the background concept also is demonstrated.

It should be stressed that the order of the overlay sets has nothing to do with their burnup, which is determined by the PATH operators; it merely specifies which materials occur where. So *a1* in *A1* can be fuel *a* that has burnt first in *A2* and then in *A3*.

Table 2. Some overlays that fill the overlay sets *A1*, *A2* and *A3* of Fig. 2.

A1 = *ov1* + *ov2*		
ov1 = OVLM(*H2O*/*−*−**/	Puts *H2O* everywhere	
Zr/*−*−*−*clad*)	*Zr* instead of *H2O* in *clad*	(1)
ov2 = OVLM(*a*/2−(2,1)−1−1/	Both top pins get *a* in *fuel*	(2)
b/1−2−1−1/	SW-pin gets *b* in *fuel*	
c/1−1−1−1)	SE-pin gets *c* in *fuel*	
A2 = *ov1* (*ov3*) + *ov4* (*ovx*)		
ov1 or *ov3*	Non-burnable materials	
ov3 = OVLM(*A1*−*ov1*−*H2O*/0)	Indirect—null movement	(3)
ov4 = OVLM(*A1*−*ov2*−*a*/*ov2*−*a*/	Indirect *mid*; indirect *pid*	(4,5)
A1−*ov2*−*b*/1−2−1−1/	Indirect *mid*; direct *pid*	(4)
d/*ov2*−*c*)	Direct *mid*; indirect *pid*	(5)
ovx = OVLM(*d*/*ov2*−*c*)	Only together with *ov3*	(5,6)
A3 = *ov1* (*ov3*) + *ov6*		
ov1 or *ov3* (no *ov5* similar to *ov3*)	Non-burnable materials	(7)
ov6 = OVLM(*A2*−*ov4*−*d*/*ov2*−*a*(1)/	*d* of *A2* replaces *a1* of *A1*	(8)
A1−*ov2*−*c*/*ov2*−*a*(2)/	*c* of *A1* replaces *a2* of *A1*	(8)
b/*ov2*−*b*/	*b2* replaces *b1* of *A2*	(9)
A1−*ov2*−*a*(1)/1−1−1−1)	*a1* of *A1* goes to SE-pin	(8,10)

Notes to Table 2

(1) General Rule 4 allows overwriting *H2O* in the *clad* regions by *Zr*.

(2) Rules 9 and 12 for direct referencing are used, since the locations of *a1* and *a2* will be referenced separately in *ov6* of *A3*. Specifying the structures by name (*cell*) would make that impossible.

(3) Recalling the Remark on indirect referencing, *H2O* specifies all regions, i.e., the entire *A1*. On account of Rule 10(i) for indirect referencing, the null movement does not violate Rule 9.

(4) The null movement may also be used, because *a1*, *a2* and *b1* keep the positions they had in *A1*.

(5) The direct *pid* of *ov2* may also be used.

(6) It is enough to change *d* in *ov3*—Note (3).

(7) Rule 3 for indirect referencing forbids *ov5* = OVLM(*A2*−*ov3*−1/0); only generic materials (names, *m*) must be used.

(8) Explicit use of *p* refers to one of the two *pid*'s of *a* in *ov2*.

(9) Rule 3 for indirect referencing forbids *ov4*−2.

(10) Rule 7 for indirect referencing allows *pid* = *ov2*−*c*, not *pid* = *ov4*−*d*.

OVSD(), THE OVERLAY-SET OPERATOR FOR THE DENSITY OVST(), THE OVERLAY-SET OPERATOR FOR THE TEMPERATURE

Purpose

The OVSD and OVST operators are used to combine density and temperature overlays into a density and a temperature-overlay set, which assign these properties to all regions of the system. These overlay sets are part of the property configurations needed by the STAT operator to define a state, i.e., a system that can be calculated by HELIOS.

Usage

name = **OVS∗ (** [*bos*/] *oid*[, *oid*] ... **)** ∗= **D** or **T**

name	represents an overlay set, which assigns density factors or temperatures to all the regions of the system by combining a possible background overlay set and one or more overlays of type D or T.
bos	= *os*, the background set, i.e. the name of an earlier-defined overlay set that will be partially or entirely overwritten by overlays of type D or T.
oid	= *ov*, the name of an earlier-defined overlay of type D or T.

Rules

1) An overlay is a property configuration that fills a part (or all) of the system. For density factors and temperatures they are defined by the OVLD and OVLT operator, respectively.
2) An overlay set is a combination of overlays that fills all of the system. For density factors and temperatures they are defined by the OVSD and OVST operators, respectively.
3) The background set (*bos*) must be an earlier-defined overlay set (*os*) of the same type, specified by name.
4) The overlay identifier (*oid*) must be an earlier-defined overlay (*ov*) of the same type, specified by name.
5) Each *oid* inside an OVS∗ operator must be unique.
6) The overlays in an overlay set may assign properties to overlapping areas of the system. Because the properties overwrite each other, the last-specified property in the last-specified overlay counts.
7) At least one density-factor and one temperature overlay set must be given.

Remarks

An overlay set assigns a property (here, density factors or temperatures) to the entire system, by combining one or more overlays for its separate parts. The concept of combining smaller and/or simpler overlays into an overall configuration enhances the flexibility and ease of the overlay input. This has been discussed under the Remarks of the OVLM operator. In general, density-factor and temperature configurations are simpler than material configurations.

Although the geometry of the system is fixed, different overlay sets may assign different properties to the same regions. For example, void branch-off calculations require one density-overlay set per void level. Another example is the Doppler branch-off calculations, which require at least two temperature-overlay sets, one at operating conditions and one at, say, a 50 K higher fuel temperature.

Examples

1)
```
ov00  = OVLD(1.0/*-*-**/ 1.0/*-*-0-cool, *-*-0-film)
ov30  = OVLD(1.0/*-*-**/ 0.7/*-*-0-cool, *-*-0-film)
ov70  = OVLD(1.0/*-*-**/ 0.3/*-*-0-cool, *-*-0-film)
film  = OVLD(1.0/*-*-0-film)

os00  = OVSD(ov00)
os30  = OVSD(ov30)
os30x = OVSD(os30/film)       or    os30x = OVSD(ov30,film)
os70  = OVSD(ov70)
os70x = OVSD(os70/film)       or    os70x = OVSD(ov70,film)
```

The above density-overlay sets could be used for burnup and/or branch-off calculations in a BWR-fuel assembly at 0%, 30% and 70% absolute void. In the sets *os00*, *os30* and *os70*, the thin coolant film between regular lattice and flow-box wall has the same void contents as the main coolant. In the sets *os30x* and *os70x*, the coolant film has zero void. Observe that the background does not introduce any simplification in this example.

2) This is the same system as that of Example 2 of the OVLT operator, the small 3×3 PWR fuel assembly shown in Fig. 1. All the pins have the same geometry and the same fuel material in the *fuel* region(s)—except the NE-pin which has water instead of fuel. The system has been built up, from bottom to top, of three rows, which have been built up, from left to right, of three pin cells.

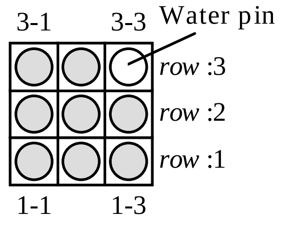


Fig.1: 3x3 fuel assembly with a water pin.

As before, the operating temperatures of fuel, clad and the rest of the assembly are 950 K, 560 K and 530 K. The overlay *temp1* is divided into the overlays *temp1a* and *temp1b*, while *temp2* for the water pin and its clad is yet another variant of this overlay. The extra overlay *tempd* has been introduced for Doppler branch-off calculations. The overlay sets for the regular and the Doppler calculations, *TEMP* and *TDOP*, are shown in boxes. Observe again that the background does not simplify matters.

```
temp1a = OVLT( 950/row-cell-pin-fuel) !  950K to all fuel regions !
temp1b = OVLT( 560/*-*-pin-clad/      !  560K to all clad regions !
               530/*-*-0-*)           !  530K to non-pin regions  !
temp2  = OVLT( 530/3-3-pin-*)         !  530K to entire water pin !
tempd  = OVLT(1000/row-cell-pin-fuel) ! 1000K to all fuel regions !
```

```
TEMP   = OVST(temp1a, temp1b, temp2)
TDOP   = OVST(TEMP/   tempd,  temp2)
                                        or
TDOP   = OVST(tempd,  temp1b, temp2)
```

In the second variant of *TDOP*, *temp1b* cannot be left out, assuming that the non-*fuel* regions still have the temperatures assigned in the *TEMP* overlay set. That is because each overlay set must fill the entire system with its property.

OVSM(), THE OVERLAY-SET OPERATOR FOR MATERIALS

Purpose

The OVSM operator is used to combine material overlays into a material-overlay set, which assigns materials to all the regions of the system. A material-overlay set is one of the property configurations needed by the STAT operator to define a state, i.e., a system that can be calculated by HELIOS.

Usage

name = **OVSM (** [*bos*/] *oid*[, *oid*] ... **)**

name	represents a material-overlay set, which assigns materials to all the regions of the system by combining a possible background overlay set and one or more material overlays.
bos	is the background, i.e., the name of an earlier-defined material-overlay set that will be partially (or entirely) overwritten by material overlays.
oid	is an overlay identifier that refers directly (d) or indirectly (i) to all or part of an earlier-defined material overlay:

(d)	$oid = ov$	is the name of a material overlay;
(i)	$oid = os - ov[-m]$ denotes the generic material (*m*), or all the materials in an overlay (*ov*) of an earlier-defined material-overlay set (*os*).	

Rules

1) A material overlay is a material configuration that fills part (or all) of the system; it is defined by the OVLM operator or by a DBMAT operator (see there).
2) A material-overlay set is a combination of overlays that fills the entire system; it is defined by the OVSM operator.
3) The background must be a material-overlay set specified by name.
4) The overlay identifier (*oid*) used for direct referencing must be the name of an overlay (*ov*), earlier defined by the OVLM operator.
5) If *oid* is an indirect reference, it must refer to either one generic material (*m*), or to all the generic materials in a material overlay (*ov*) in a material-overlay set (*os*). This leads to the construction

$$oid = os - ov[-m].$$

6) If *oid* is an indirect reference, *os* must be the name of an earlier-defined material-overlay set.

7) If *oid* is an indirect reference, *ov* must be the name of a material overlay that occurs in *os*.
8) If *oid* is an indirect reference, and *m* is given, *m* must be the name of a generic material in *ov*.
9) If *oid* is an indirect reference, and no *m* is given, *oid* refers to all the material identifiers in *ov*, which must contain only generic (direct) materials.
10) Each *oid* inside an OVSM operator must be unique.
11) The overlays in an overlay set may assign materials to overlapping areas of the system. Because the materials overwrite each other, the last-specified material in the last-specified overlay counts.
12) There may be not more than one OVSM operator that contains the name of an OVLM operator generated by a DBMAT with unspecified burnup; see DBMAT operator, Rule 5 for restart materials with unspecified burnup.

Remarks

An overlay set assigns a property (here, materials) to the entire system, by combining one or more overlays for its separate parts. The concept of combining various overlays into an overall configuration enhances the flexibility and ease of the overlay input. This has been discussed under the Remarks of the OVLM operator.

While the geometry of the system is unique, the same regions may be filled by several overlay sets. For example, the difference between an uncontrolled and a controlled fuel assembly is that the control-rod regions are filled with either gap water or control-rod materials. Another example is that of two overlay sets which differ by the material in one fuel pin only. This could be used to describe the burnup of an assembly in which a failed pin is replaced by a fresh one. In a third example, two identical overlay sets, with different names, represent two identical but physically distinct fuel assemblies. In that case, geometrically identical pins of the two assemblies (at the same or different burnups) can be used in a third, reconstituted fuel assembly.

Being the null movement of an entire overlay set, the background may be considered an indirect material reference. Owing to the background concept, no other form of indirect referencing is needed for common burnup and branch-off calculations of reload-fuel design—not even for the replacement of used fuel pins by fresh ones. This is illustrated in the Examples, and also in example 3 of the TREE operator.

The order of defining the overlay sets has no bearing on the order in which they are being used in the calculations. That is determined by the PATH and TREE operators (see there). Actually, an overlay set that does not appear in any state will not figure at all in the calculations (see STAT operator). Example 2 will show that sometimes it may be more convenient to define the overlay sets in a sequence different from the one in which they will be used.

Examples

1)
```
ov1 = OVLM(H2O/*-*-**/ Zr/*-*-*-clad)
ov2 = OVLM(a/2-(2,1)-1-1/ b/1-2-1-1/ c/1-1-1-1)
ovx = OVLM(d/ov2-c)
A1  = OVSM(ov1,ov2)
A2  = OVSM(A1/ ovx)
ov6 = OVLM(A2-ovx-d/ov2-a(1)/ A1-ov2-c/ov2-a(2)/
                    b/ov2-b   / A1-ov2-a(1)/ov2-c)
A3  = OVSM(A1/ ov6)
```

This is the same example as for the OVLM operator. For easy reference, the figure is reproduced here as Fig.1. Also, the definitions of some of the overlays are repeated. However, the indirect reference to *ov4* has been replaced by a reference to *ovx*.

Because of the background, no indirect referencing is needed for *A2*. This represents the case in which a broken-down fuel pin in an "assembly" is replaced by a fresh one. Because the sequence in which they are defined does not specify their burnup order, *A1* could represent the assembly that burns until the pin breaks down and *A2* the assembly with the fresh pin; or just the other way around.

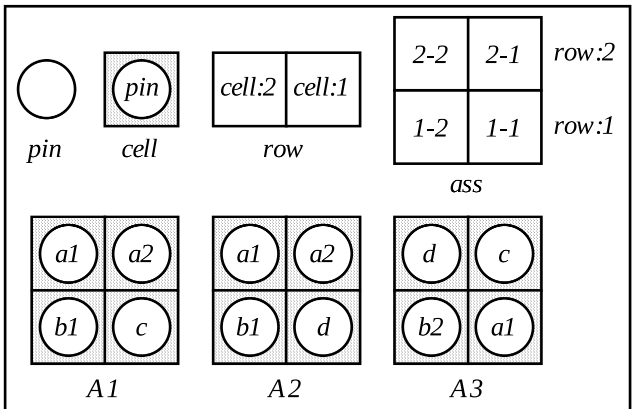


Fig. 1: Overlay sets to illustrate the use of the OVLM and OVSM operators.

Overlay set *A3* differs significantly from both *A1* and *A2*. The pins it has in common with the other two overlay sets don't share the same regions. This case, where true materials change location, can no longer be described by the implied null movement of a background. Indirect referencing becomes a necessity.

2) The next example is illustrated in Fig.2. The overlay sets *S1* and *S2* represent a fuel assembly with and without the control rod *cr*. *S*3 represents another assembly, controlled by the same *cr*.

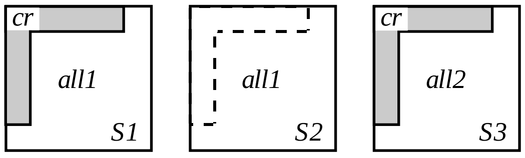


Fig. 2: The overlay sets of Example 2.

First, the controlled assembly (*S1*) is burnt for a while. Next, the control rod is removed and the uncontrolled assembly (*S2*) is burnt. Finally, the other controlled assembly (*S3*) is burnt, with the control rod that was already in *S1*, where it acquired a certain burnup.

A variety of overlay and overlay-set operators that can be used in this example are collected in Table 1. For brevity's sake, not all the OVLM arguments are stated explicitly. Overlay sets that are wrong are in brackets. Observe that a mere change in the sequence of defining the overlay sets, simplifies their operators.

If the control rod is always used with the same assembly, the only indirection necessary is the background. Reinserting the control rod into the original assembly, and not into the other which resulted in *S3*, would be a return from overlay set *S2* to *S1*. The fuel pins, which have burnt all the time, will burn on, while the *cr* materials will resume their depletion. Also, both regular and inverted control branch-off calculations can be done with these two sets only.

Table 1. Various overlays and overlay sets of Example 2.

all1 = OVLM(assembly #1) *all2* = OVLM(assembly #2) *cr* = OVLM(control-rod) *nocr* = OVLM(control-rod)	Fills entire assembly; water in *cr*-regions As *all1*, but now the other assembly Fills *cr*-regions with *cr*-materials Fills *cr*-regions with water
[*S1*= OVSM(*cr*, *all1*)] *S1* = OVSM(*all1*, *cr*)	Error: *all1* fills *cr* with water again Good: *cr*-materials overwrite *cr*-regions
allx = OVLM(*S1*−*all1*/0) *crx* = OVLM(*S1*−*cr*/0)	Careful: in *S1*, in the meantime, *cr*-regions have been filled with *cr*-materials Good: *crx* has the materials of *cr*
[*S2*= OVSM(*allx*)] [*S2*= OVSM(*all1*)] *S2* = OVSM(*allx*, *nocr*) *S2* = OVSM(*S1*−*all1*, *nocr*) *S2* = OVSM(*S1*/ *nocr*)	Error: *allx* has *cr*-materials in *cr*-regions Error: *all1* in *S2* specifies fresh materials (OVLM operator, Rule 2 for materials) Good: *cr*-regions got their water back Good: as above, but avoids *allx* Better: background, no other indirections
[*S3*= OVSM(*all2*, *cr*)] *S3* = OVSM(*all2*, *crx*) *S3* = OVSM(*all2*, *S1*−*cr*)	Error: *cr* in *S3* specifies fresh materials (OVLM operator, Rule 2 for materials) Good: *crx* has the used materials of *cr* Better: avoids *crx*, not its indirection
S2 = OVSM(*all1*) *S1* = OVSM(*S2*/ *cr*) *S3* = OVSM(*all2*, *S1*−*cr*)	Best: this modified definition order avoids the need for *nocr* and has the fewest indirections

PAR(), THE PARAMETER OPERATOR and ARITHMETIC EXPRESSIONS

Purpose

The PAR operator is used to define parameters. A parameter is a name (including the leading "$" character) that represents some or all of the arguments and delimiters of any input operator but the SET operator. Utilizing parameters, a generalized input can be set up from which the input of a specific case is obtained by defining the parameter values. The usefulness of parameters is enhanced if they are used in arithmetic expressions.

Usage

$*name* **= PAR(** *par* **)**

$*name*	represents a combination of arguments and delimiters that is allowed inside at least one input operator, the SET operator excluded.
par	specifies the above combination of arguments and delimiters.

Rules for parameters

1) The PAR operator defines a parameter by its name ($*name*), which must begin with a dollar sign ($) that must be outside possible apostrophes enclosing *name*. $*name* must not exceed 80 characters, including the $, but excluding the possible apostrophes.
2) A parameter represents the argument(s and delimiters) of its defining PAR operator (*par*).
3) A parameter may occur inside any operator, except the SET operator, and must be separated from possible other arguments by delimiters.
4) As a consequence of Rules 2 and 3, *par* must have a syntax that is allowed inside the operator(s) where $*name* will occur.
5) Parameters that occur directly or indirectly in other operators than the PAR operator must be defined in the input.
6) If a parameter represents an arithmetic expression or part thereof, it must be in quotes: $*name* = PAR("*par*"). See Rules for arithmetic expressions.

Rules for arithmetic expressions

1) In any input operator, an argument that must be a numerical constant may be represented by an arithmetic expression whose result is equal to the value of that constant.
2) An arithmetic expression must be enclosed in quotes ("…").

3) An arithmetic expression consists of one or more *terms*, separated by one of the following *operators*:

+	for addition (must not be used as the sign of a term);
–	for subtraction, or before a term as its sign;
∗	for multiplication;
/	for division;
∗∗	for exponentiation.

4) A term is a numerical constant, an expression enclosed in a pair of balancing parentheses, or a parameter. Actually, the quotes that enclose an arithmetic expression are equivalent to a pair of parentheses; see also Rule 6 for parameters.

5) Parentheses can be used to avoid the occurrence of two operators in sequence, such as 2∗–5 — which must be written as 2∗(–5) — or to override the execution sequence of operators described in Rule 6.

6) The evaluation sequence of an expression in parentheses is: first exponentiations, from right to left; next multiplications and divisions, from left to right; and, finally, additions and subtractions, from left to right.

7) All arithmetic is done in double precision, and the final result is a real number or a true integer. A true integer is the result of operations on integer constants, in which both the intermediate and the final result are integers.

8) The final result that replaces an expression is presented in one of three ways:
   - as a fixed-point number with at most seven significant digits if it is real and lies inside the range (9,999 – 0.01);
   - as a floating-point number (E-format) with at most seven significant digits if it is real and lies outside the range (9,999 – 0.01);
   - as an integer with at most twelve digits if it is a true integer.

   In all presentations, leading blanks are omitted, as are trailing zeros behind the decimal point, and, if possible, also the decimal point.

9) The rules governing the representation of numerical constants are described in the Introduction; see also Remarks.

Remarks

There are four tools that facilitate the input preparation: sets (see SET operator), parameters, names and arithmetic expressions. The sets predefine one or more entire input operators. For a given type of fuel assembly, its geometric description, as well as the compositions of the most commonly occurring materials, could be stored as one or more sets in a data-base. It then suffices to specify these sets, instead of having to set up the relevant input operators for each case to be run.

Parameters, on the other hand, represent one or more undefined arguments of input operators, some of which have to be defined for each case input. For example, in a generalized input based on sets, pin radii, temperatures, and material assignments could be given as parameters. This introduces flexibility in the set input; different lattices of a fuel-assembly type could be run by merely defining the parameters differently.

Further, a judicious use of region names significantly facilitates the overlay input. It is easy to assign a material *H2O* to all the coolant regions if they all have the name *cool*. The only input that might appear less trivial to simplify, the pin map with the enrichment distribution, can also be made easy, as is shown in the Examples.

Finally, the introduction of arithmetic expressions further increases input flexibility. For instance, in a pin cell with pitch p and clad radius r, the node coordinates that specify a subdivision of the coolant into four quadrants require the values of p, $p/2$ and $p/2 \pm r$. Using arithmetic, they can be specified as $\$p$, "$\$p/2$" and "$\$p/2 \pm \$r$". Then, for each case to be run, only two parameters have to be defined, $\$p$ and $\$r$, instead of one for each of the four values.

The arithmetic capability was primarily developed for use with parameters, which is why it is discussed here in connection with the PAR operator. However, arithmetic without parameters is also possible, like "1.5/2" instead of "$\$p/2$" with $\$p =$ PAR("1.5"). (Observe that the use of quotes in the PAR operator is in accordance with Rule 6 for parameters.)

The representation of numerical constants and the arithmetic differ from those in FORTRAN as follows:

- embedded blanks in numerical constants are ignored;
- if scientific notation with E-format is used, there must be at least one digit before the E;
- integer constants without decimal point or E may represent real constants, e.g., 919 may represent 919.0;
- integer division is executed as a normal division, e.g., $2/3 = 0.666\ldots$ instead of 0.

Example

<u>**Note**</u>: The example given below serves only to demonstrate what can be done with include sets, arithmetic and parameters. It is in no way a complete or a recommended example of a BWR-lattice input. Complete inputs for various lattice types can be obtained from Studsvik Scandpower.

This example demonstrates a possible <u>short input</u> for the 4×4 mini-BWR assembly of Fig. 1 (not to scale!). The system is made up of ten structure types: one pin cell, three side, four corner and two control-element structures. The side and corner structures consist of parts of the wide and narrow gaps, of the flow box and of the coolant film at the inside of the box. The wide-narrow and narrow-wide corners are different structures, as one can not be transformed into the other by rotations and/or translations. Observe also the detector in the narrow-narrow corner.

The mesh division and node numbering are shown, except inside the pins. The cell coolant is divided into four quadrants, a division that extends into the side structures; while the widths of the wide and narrow gaps are divided into three and two layers, respectively.

To keep the example simple, the centre piece of the control element has been chosen to coincide with the gap mesh. The control blades contain only four absorber pins whose radii coincide with one of the gap mesh lines.

Two sets are used: &*MAT* with preset materials and the diagonal albedo matrix, and &'*BWR*' with parametrized geometry and overlay input. The set input is often called the expert input. In the short input, only the second set has to be specified, because it includes the first one. The set &*MAT* and the &ADD instruction to store it into the data-base are shown below.

Observe that both sets reside in the data-base 'A.SETS'. They are stored below the catalog AURORA, which in turn is stored below the catalog SAMPLE in the root catalog.

```
+SET                                ! See Introduction, "How to run AURORA" !

&ADD      = SET('A.SETS'/SAMPLE;AURORA;&MAT)

'H2O-300' = MAT(0.9961/ 1001, 11.19; 8016, 88.81)            ! H2O at 300K !
'H2O-400' = MAT(0.9378/ 1001, 11.19; 8016, 88.81)            ! H2O at 400K !
'H2O-500' = MAT(0.8309/ 1001, 11.19; 8016, 88.81)            ! H2O at 500K !
'H2O-550' = MAT(0.7595/ 1001, 11.19; 8016, 88.81)            ! H2O at 550K !

 Zr       = MAT(      /40002, 4.292E-2)                      ! Zircaloy    !
 Steel    = MAT( 7.9  /26001, 100)                           ! Stainless   !
 B4C      = MAT( 1.8  / 5000, 78.28; 6000, 21.72)            ! B4C!        !
'UO2-det' = MAT( NB / /92235, 1E-10)                         ! Detector U5 !

'UO2-1.6' = MAT(  10  /92235, 1.6;  92238, 98.4;  8001, 0)    ! 1.6wt% UO2  !
'UO2-1.8' = MAT(  10  /92235, 1.8;  92238, 98.2;  8001, 0)    ! 1.8wt% UO2  !
'UO2-2.0' = MAT(  10  /92235, 2.0;  92238, 98.0;  8001, 0)    ! 2.0wt% UO2  !
'UO2-2.2' = MAT(  10  /92235, 2.2;  92238, 97.8;  8001, 0)    ! 2.2wt% UO2  !
'UO2-2.4' = MAT(  10  /92235, 2.4;  92238, 97.6;  8001, 0)    ! 2.4wt% UO2  !
'UO2-2.6' = MAT(  10  /92235, 2.6;  92238, 97.4;  8001, 0)    ! 2.6wt% UO2  !
'UO2-2.8' = MAT(  10  /92235, 2.8;  92238, 97.2;  8001, 0)    ! 2.8wt% UO2  !
'UO2-3.0' = MAT(  10  /92235, 3.0;  92238, 97.0;  8001, 0)    ! 3.0wt% UO2  !
'UO2-3.2' = MAT(  10  /92235, 3.2;  92238, 96.8;  8001, 0)    ! 3.2wt% UO2  !
'UO2-BA'  = MAT(                                   ! 3wt% UO2 + 3wt% Gd2O3 !
                /92235, 6.563E-4;  92238, 2.095E-2;  8001, 0.04471;
                 64154, 2.090E-5;  64155, 1.473E-4; 64156, 2.050E-4;
                 64157, 1.563E-4;  64158, 2.468E-4)

 White    = ALB(1/1/1)  ! Diagonal albedo: a(i,i)=1 for neutrons and gammas !
```

It is in the set &*BWR*, shown below and on the following pages, that full use is made of both parameters and arithmetic. First, the set &*MAT* is read. Next, using arithmetic, parameters are set up for the mesh coordinates in the side and corner structures, whereupon these structures are defined. They are based on the as yet undefined parameters $*p*, $*box*, $*bwt*, $*wide* and $*narrow*, i.e., pitch, box and gap dimensions must be defined in the short input. Often, these data are fixed for a given fuel design, so they could have been defined in the present set, too.

Note that the narrow-narrow corner contains an annular detector, described by the parameters $*rdi* and $*rdo,* the inner and outer radii of the detector tube. The two regions are called *deti* and *deto*, the inner and outer regions. Again, to keep this example simple their radii coincide with gap-mesh lines.

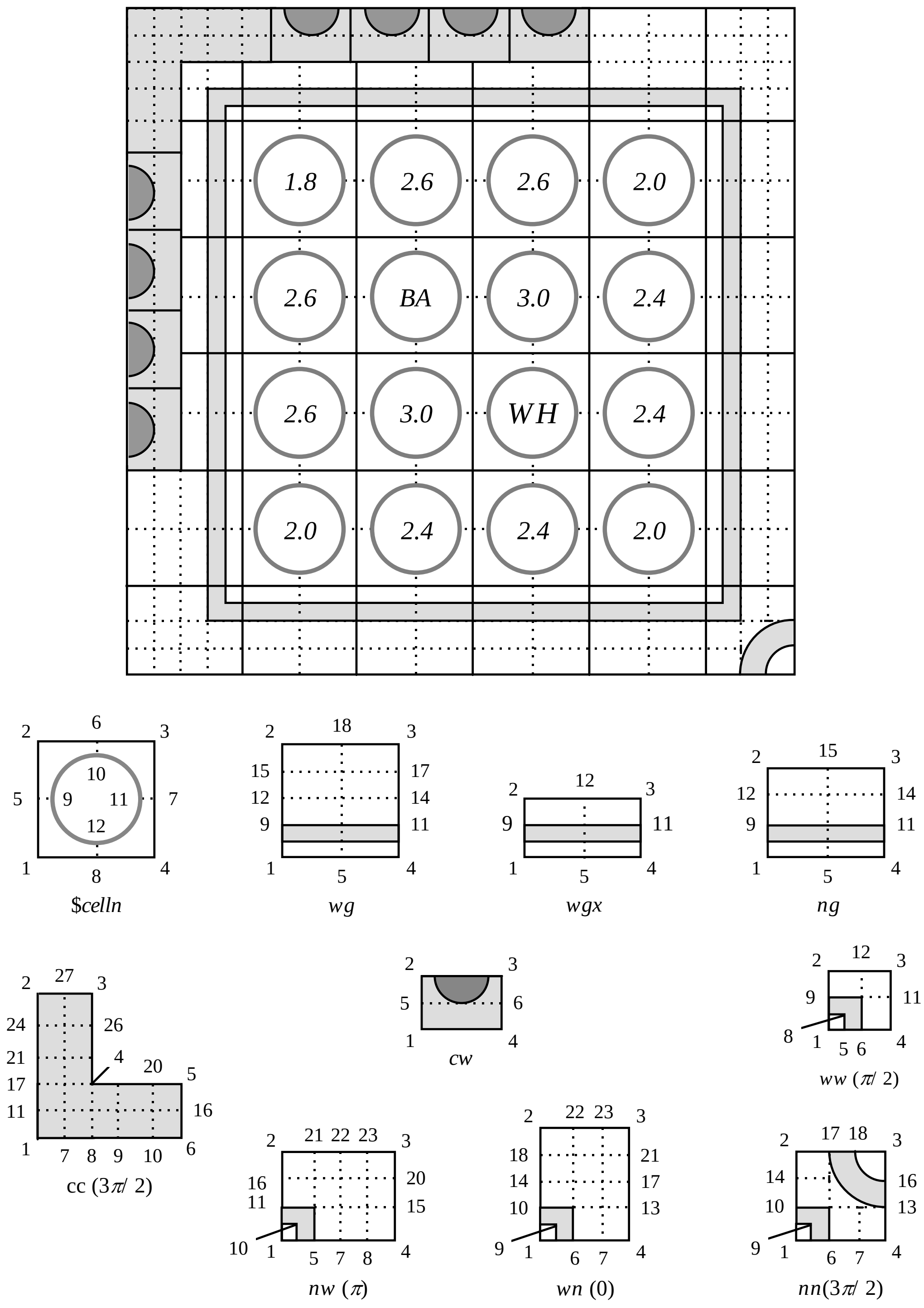


Fig. 1: Example of a BWR assembly and its ten basic structures.

Next, one of the four absorber-pin cells in the blade of the control element and the centre piece are set up. They are based on the parameters $*clong*, $*wlong* and $*rab*. Again, to keep the example simple, the wing length $*wlong* was chosen such that the wings terminate at the extension of a mesh line of the pin cells. Also the radii of the absorber pins, $*rab*, are equal to the mesh spacing of the wide gaps. Before proceeding with the cell structures, an entire control wing is set up. By specifying the coupling order $k = 0$, each wing is treated as one space element in the calculations. [For true control blades this is not recommended. In a long blade the probability of a particle being born or entering at one end of the blade to escape or collide at the other end is insignificant.]

Then, four cell structures are specified as parameters; the auxiliary $*peri* and $*cregs* merely shorten their specifications. While the CCS names are fixed, their outer radii and internal structures are left free. This allows the User to define (name) in the input any number of structures with four different geometric pin types.

Thereafter, the system is built up from six rows of six structures, the centre piece of the control element and its two wings. [Using the intermediate subsystem $*row* is not recommended. The geometric input is easier to understand and debug if the system is flat, i.e., has few subsystem levels, even though this makes the final CNX operator more complex.] The three reduced wide-gap elements, and the four narrow-gap elements are combined into four space elements ($k = 0$). Otherwise, coupling between gap and corner elements is by $k = 2$. The remaining coupling order, k, is a free parameter. White (isotropic) boundary conditions are applied. By leaving $*prow** free, it is possible to produce later in the short input a visual pin map.

Finally, leaving the enrichment distribution, and the fuel and coolant temperatures as free parameters ($*fuels*, $*tfuel* and $*tcool*) the overlays and four states with fixed names are specified.

```
&ADD    = SET('A.SETS'/SAMPLE;AURORA;&BWR)

&MAT    = SET('A.SETS'/SAMPLE;AURORA)          !Reads material set         !

$film   = PAR("($box-4*$p)/2")                 !Coolant film in BWR        !
$delw   = PAR("$wide/3")                       !Wide gap mesh              !
$deln   = PAR("$narrow/2")                     !Narrow gap mesh            !
$w1     = PAR("$film+$bwt")                    !Film + box wall            !
$w2     = PAR("$w1+$delw")                     !Wide: 1st mesh line        !
$w3     = PAR("$w2+$delw")                     !Wide: 2nd mesh line        !
$w4     = PAR("$w1+$wide")                     !Wide: 3rd mesh line        !
$n1     = PAR("$w1")                           !Film + box wall            !
$n2     = PAR("$n1+$deln")                     !Narrow: 1st mesh line      !
$n3     = PAR("$n1+$narrow")                   !Narrow: 2nd mesh line      !

$rdi    = PAR($deln)                           !Inner detector tube        !
$rdo    = PAR($narrow)                         !Outer detector tube        !

$c1     = PAR("$delw")                         !Centre piece: 1st mesh line !
$c2     = PAR("$c1+$delw")                     !Centre piece: 2nd mesh line !
$c3     = PAR("$c2+$delw")                     !Centre piece: 3rd mesh line !
$c4     = PAR("$c3+$w1")                       !Centre piece: 4th mesh line !

$rab    = PAR("$delw")                         !Radius of absorber pin      !
$wlong  = PAR("$w4+3*$p-$clong")               !Length of control wing      !
$aa     = PAR("$wlong/4")                      !Absorber cell: pitch        !
$d1     = PAR("$delw")                         !Absorber cell: 1st mesh line!
```

```
$d2     = PAR("2*delw")                       !Absorber cell: 2nd mesh line!
$ac     = PAR("$aa/2")                        !Absorber pin: centre       !

wg      = STR(!Wide gap side, including box and film!
              (0,0)     (0,$w4)        ($p,$w4)   ($p,0)               ! 1-4 !
                        ("$p/2",0)                                     ! 5-5 !
              (0,$film) ("$p/2",$film) ($p,$film)                      ! 6-8 !
              (0,$w1)   ("$p/2",$w1)   ($p,$w1)                        ! 9-11!
              (0,$w2)   ("$p/2",$w2)   ($p,$w2)                        !12-14!
              (0,$w3)   ("$p/2",$w3)   ($p,$w3)                        !15-17!
                        ("$p/2",$w4)                                   !18-18!
              / 4,gap //        1,6,7,5,film;     5,7,8,4,film;
              6,9,10,7,box;     7,10,11,8,box;    9,12,13,10,gap;
              10,13,14,11,gap;  12,15,16,13,gap;  13,16,17,14,gap;
              15,2,18,16,gap                      !Last is default gap!)

wgx     = STR(!Reduced wide gap side, including box and film!
              (0,0)     (0,$w2)        ($p,$w2)   ($p,0)               ! 1-4 !
                        ("$p/2",0)                                     ! 5-5 !
              (0,$film) ("$p/2",$film) ($p,$film)                      ! 6-8 !
              (0,$w1)   ("$p/2",$w1)   ($p,$w1)                        ! 9-11!
                        ("$p/2",$w2)                                   !12-12!
              / 4,gap //        1,6,7,5,film;     5,7,8,4,film;
              6,9,10,7,box;     7,10,11,8,box;    9,2,12,10,gap)

ng      = STR(!Narrow gap side, including box and film!
              (0,0)     (0,$n3)        ($p,$n3)   ($p,0)               ! 1-4 !
                        ("$p/2",0)                                     ! 5-5 !
              (0,$film) ("$p/2",$film) ($p,$film)                      ! 6-8 !
              (0,$n1)   ("$p/2",$n1)   ($p,$n1)                        ! 9-11!
              (0,$n2)   ("$p/2",$n2)   ($p,$n2)                        !12-14!
                        ("$p/2",$n3)                                   !15-15!
              / 4,gap //        1,6,7,5,film;     5,7,8,4,film;
              6,9,10,7,box;     7,10,11,8,box;    9,12,13,10,gap;
              10,13,14,11,gap;  12,2,15,13,gap)

ww      = STR(!Reduced wide-wide corner, including box and film!
              (0,0)     (0,$w2)        ($w2,$w2)  ($w2,0)              ! 1-4 !
              ($film,0) ($w1,0)        (0,$film)  ($film,$film)        ! 5-8 !
              (0,$w1)   ($w1,$w1)      ($w2,$w1)  ($w1,$w2))           ! 9-12!
              / 4,gap //        1,9,10,6,box;     1,7,8,5,film;
              6,10,11,4,gap;    9,2,12,10,gap)

wn      = STR(!Wide-narrow corner, including box and film!
              (0,0)     (0,$w4)        ($n3,$w4)  ($n3,0)              ! 1-4 !
              ($film,0) ($n1,0)        ($n2,0)                         ! 5-7 !
              (0,$film) ($film,$film)                                  ! 8-9 !
              (0,$w1)   ($n1,$w1)      ($n2,$w1)  ($n3,$w1)            !10-13!
              (0,$w2)   ($n1,$w2)      ($n2,$w2)  ($n3,$w2)            !14-17!
              (0,$w3)   ($n1,$w3)      ($n2,$w3)  ($n3,$w3)            !18-21!
              ($n1,$w4) ($n2,$w4)                                      !22-23!
              / 4,gap //        1,10,11,6,box;    1,8,9,5,film;
              6,11,12,7,gap;    7,12,13,4,gap;    10,14,15,11,gap;
              11,15,16,12,gap;  12,16,17,13,gap;  14,18,19,15,gap;
              15,19,20,16,gap;  16,20,21,17,gap;  18,2,22,19,gap;
              19,22,23,20,gap)

nw      = STR(!Narrow-wide corner, including box and film!
              (0,0)     (0,$n3)        ($w4,$n3)  ($w4,0)              ! 1-4 !
              ($film,0) ($w1,0)        ($w2,0)    ($w3,0)              ! 5-8 !
              (0,$film) ($film,$film)                                  ! 9-10!
              (0,$n1)   ($w1,$n1)      ($w2,$n1)  ($w3,$n1)  ($w4,$n1) !11-15!
              (0,$n2)   ($w1,$n2)      ($w2,$n2)  ($w3,$n2)  ($w4,$n2) !16-20!
              ($w1,$n3) ($w2,$n3)      ($w3,$n3)                       !21-23!
              / 4,gap //        1,11,12,6,box;    1,9,10,5,film;
              6,12,13,7,gap;    7,13,14,8,gap;    8,14,15,4,gap;
              11,16,17,12,gap;  12,17,18,13,gap;  13,18,19,14,gap;
              14,19,20,15,gap;  16,2,21,17,gap;   17,21,22,18,gap;
              18,22,23,19,gap)
```

```
nn      = STR(!Narrow-narrow corner, including box, film and detector!
              (0,0)     (0,$n3)        ($n3,$n3)  ($n3,0)              ! 1-4 !
              ($film,0) ($n1,0)        ($n2,0)                         ! 5-7 !
              (0,$film) ($film,$film)                                  ! 8-9 !
              (0,$n1)   ($n1,$n1)      ($n2,$n1)  ($n3,$n1)            !10-13!
              (0,$n2)   ($n1,$n2)      ($n3,$n2)                       !14-16!
              ($n1,$n3) ($n2,$n3)                                      !17-18!
              / 4,gap / dpin($n3,$n3)2 /          1,10,11,6,box;
              1,8,9,5,film;     6,11,12,7,gap;    7,12,13,4,gap;
              10,14,15,11,gap;  14,2,17,15,gap)

dpin    = CCS(!Detector: inner & outer ring!
              $rdi,$rdo // deti,deto)

cw      = STR(!Absorber cell of control wing!
              (0,0)     (0,$d2)        ($aa,$d2)  ($aa,0)              ! 1-4 !
              (0,$d1)   ($aa,$d1)      ($ac,$d1)                       ! 5-7 !
              !Tangential node nr 7 must be given!
              / 4,blade / apin($ac,$d2)1 /
              1,5,6,4,blade)

apin    = CCS($rab // abs) !One-region control pin !

wing    = CNX(!Control wing; k=0 => one space element!
              cw, cw, cw, cw  / (1,3,4)0(2,2,1) /
              (2,3,4)0(3,2,1) / (3,3,4)0(4,2,1) )

cc      = STR(!Centre piece of the control element!
              (0,0)     (0,$clong)     ($c2,$clong)                    ! 1-3 !
              ($c2,$c2) ($clong,$c2)   ($clong,0)                      ! 4-6 !
              ($c1,0)   ($c2,0)        ($c3,0)      ($c4,0)            ! 7-10!
              (0,$c1)   ($c1,$c1)      ($c2,$c1)    ($c3,$c1)          !11-14!
              ($c4,$c1) ($clong,$c1)                                   !15-16!
              (0,$c2)   ($c1,$c2)      ($c3,$c2)    ($c4,$c2)          !18-20!
              (0,$c3)   ($c1,$c3)      ($c2,$c3)                       !21-23!
              (0,$c4)   ($c1,$c4)      ($c2,$c4)    ($c1,$clong)       !24-27!
              / 6,blade //       1,11,12,7,blade;   7,12,13,8,blade;
              8,13,14,9,blade;   9,14,15,10,blade;  10,15,16,6,blade;
              11,17,18,12,blade; 12,18,4,13,blade;  13,4,19,14,blade;
              14,19,20,15,blade; 15,20,5,16,blade;  17,21,22,18,blade;
              18,22,23,4,blade;  21,24,25,22,blade; 22,25,26,23,blade;
              24,2,27,25,blade)

$peri   = PAR(!Peripheral nodes in a cell with 4 coolant regions!
              (0,0)      (0,$p)      ($p,$p)     ($p,0)                ! 1-4 !
              (0,"$p/2") ("$p/2",$p) ($p,"$p/2") ("$p/2",0))           ! 5-8 !

$cregs  = PAR(!Defining 4 coolant regions in a pin cell!
              1,5,9,12,8,cool;  5,2,6,10,9,cool;  6,3,7,11,10,cool)

$cell1  = PAR(!Pin cell with 4 coolant regions(pin radius $rp1)!
              $peri ("$p/2−$rp1","$p/2")  ("$p/2","$p/2+$rp1")         ! 1-10!
                    ("$p/2+$rp1","$p/2")  ("$p/2","$p/2−$rp1")         !11-12!
                    / 4,cool / pin1("$p/2","$p/2") /   $cregs)

$cell2  = PAR(!Pin cell with 4 coolant regions(pin radius $rp2)!
              $peri ("$p/2−$rp2","$p/2")  ("$p/2","$p/2+$rp2")         ! 1-10!
                    ("$p/2+$rp2","$p/2")  ("$p/2","$p/2−$rp2")         !11-12!
                    / 4,cool / pin2("$p/2","$p/2") /   $cregs)

$cell3  = PAR(!Pin cell with 4 coolant regions(pin radius $rp3)!
              $peri ("$p/2−$rp3","$p/2")  ("$p/2","$p/2+$rp3")         ! 1-10!
                    ("$p/2+$rp3","$p/2")  ("$p/2","$p/2−$rp3")         !11-12!
                    / 4,cool / pin3("$p/2","$p/2") /   $cregs)
```

```
$cell4  = PAR(!Pin cell with 4 coolant regions(pin radius $rp4)!
              $peri ("$p/2−$rp4","$p/2")  ("$p/2","$p/2+$rp4")         ! 1-10!
                    ("$p/2+$rp4","$p/2")  ("$p/2","$p/2−$rp4")         !11-12!
                    / 4,cool / pin4("$p/2","$p/2") /   $cregs)

$row    = PAR(!Coupling of 6 structures, the first and last are the
               wide and narrow gap sides, the others are pin cells.
               Rotate the wide side by(pi/2 and the narrow by pi/2.!
              / (1,4,1)1(2,2,1) / (2,3,4)1(3,2,1) / (3,3,4)1(4,2,1)
              / (4,3,4)1(5,2,1) / (5,3,4)1(6,1,4) )

top     = CNX(ww, wgx, wgx, wgx, wg, wn          !Rotate ww by pi/2!
              / (1,4,1)2(2, 2,1) / (2,3,4)0(3,2,1) / (3,3,4)0(4,2,1)
              / (4,3,4)2(5,12,1) / (5,3,4)2(6,2,1) )

bot     = CNX(nw, ng, ng, ng, ng, nn !Rotate nw & ng by pi; nn by pi/2!
              / (1,1,2)2(2,4,3) / (2,1,2)0(3,4,3) / (3,1,2)0(4,4,3)
              / (4,1,2)0(5,4,3) / (5,1,2)2(6,1,4))

row1    = CNX(wgx, $prow1, ng / $row)
row2    = CNX(wgx, $prow2, ng / $row)
row3    = CNX(wgx, $prow3, ng / $row)
row4    = CNX(wg,  $prow4, ng / $row)

system  = CNX(top, row1, row2, row3, row4, bot, cc, wing, wing
              /(1-1,2)(1-6,4)$k(2-1,3)(2-6,2)/(2-1,2)(2-6,3)$k(3-1,3)(3-6,2)
              /(3-1,2)(3-6,3)$k(4-1,3)(4-6,2)/(4-1,2)(4-6,3)$k(5-1,3)(5-6,2)
              /(5-1,2)(5-6,3)$k(6-1,4)(6-6,2)/     (1-1,2,4)$k(7,20,26)
              /       (7,2,3)$k(8-1,2,1)     /       (7,5,6)$k(9-4,4,3))

system  = NEWK(!Set corner-gap coupling k=2 (wgx with wg gets k=2)
                 and readjust the inter-gap coupling back to k=0.!
               2/(1-1,1,2)(1-6,4,1)(6-1,4,1)(6-6,1,2)/
               0/(2-1,1,2)(3-1,1,2)/          !Left !
               2/(4-1,1,2)        /           !Left !
               0/(2-6,3,4)(3-6,3,4)(4-6,3,4)) !Right!

system  = BDRY((7,1,1)$k(White))  !Diagonal albedo: all around from cc corner!

outside = OVLM('H2O-550'/*-*-0-*/                    !Water outside pins    !
                Zr      /*-*-0-box,*-*-*-clad/       !Zr in box and clad    !
                Steel   /*-*-dpin-deto/              !Steel of detector tube!
               'U-det'  /*-*-dpin-deti/              !U vapour of detector  !
fuels   = OVLM($fuels)                               !User's fuel materials !
fake    = OVLM('H2O-550'/*-*-0-blade,*-*-*-abs/      !Water in control rod  !
true    = OVLM( Steel   /*-*-0-blade/                !Steel of control rod  !
                B4C     /*-*-apin-*)                 !B4C of absorber pins  !

mos1    = OVSM(outside, fuels, fake)                 !Uncontrolled          !
mos2    = OVSM(mos1/true)                            !Controlled            !

dov40   = OVLD(1/*-*-**/ 0.6/*-*-0-(cool,film))      !40% coolant void      !
dov40   = OVLD(1/*-*-**/ 0.3/*-*-0-(cool,film))      !70% coolant void      !

dos40   = OVSD(dov40)                                !40% coolant void      !
dos70   = OVSD(dov70)                                !70% coolant void      !

tov1    = OVLT($tcool/*−*−**/  $tfuel/*-*-*-fuel)

tos1    = OVST(tov1)

v40u    = STAT(mos1, dos40, tos1, $pw)               !40% void, uncontrolled!
v40c    = STAT(mos2, dos40, tos1, $pw)               !40% void,   controlled!
v70u    = STAT(mos1, dos70, tos1, $pw)               !70% void, uncontrolled!
```

```
ng1      = GROUP(N/0)                                  !For whole energy range!
ng2      = GROUP(N/0.62,0)                             !For two-group output  !
ngall    = GROUP(N/)                                   !For fast fluence      !
gg1      = GROUP(G/0)                                  !For gamma smear       !
gg5      = GROUP(G/4E6,3E6,2E6,1E6,0)                  !For gamma detector    !

All      = AREA(<*–*–**>)
FuelMap = AREA(*-*-*-<fuel>)
DetArea = AREA(*-*-*-<deti>)                           !Detector              !
AllClad = AREA(*-*-*-<clad>)                           !For fast fluence      !
AllBox   = AREA(*-*-*-<box> )                          !For fast fluence      !

NW       = FACE((cc,7,11))
NE       = FACE((top-wn,23,21))
SE       = FACE((bot-nn,18,16))
SW       = FACE((bot-nw,23,20))
Nside    = FACE((cc,1,2)   (8-cw,2,3) (top-(wg,wn),2,3))
Eside    = FACE((1-wn,3,4) (2-ng,2,3) (3-ng,2,3)
                (4-ng,2,3) (5-ng,2,3) (6-nn,2,3))
Sside    = FACE((bot-(nw,ng),2,3)     (bot-nn,3,4))
Wside    = FACE((cc,6,1)   (9-cw,2,3) (5-ng,2,3) (6-nw,3,4))

XSset    = MACRO(ng2, All    / dn,tr,fi,kf)            !Two-group data        !
Maps     = MACRO(ng1, FuelMap/ bu,kf)                  !Burnup and power maps !
DetGam   = MACRO(gg5, DetArea/)                        !Gamma detector        !
EgAll    = MACRO(gg1, All    / ed)                     !For gamma smear       !
EgFuel   = MACRO(gg1, FuelMap/ ed)                     !For gamma smear       !
FlClad   = MACRO(T/ ngall, AllClad/)                   !For fast fluence      !
FlBox    = MACRO(T/ ngall, AllBox /)                   !For fast fluence      !

'1/v'    = MICRO(ng2, All     /1          / ab)
FuelIsos= MICRO(ng1, FuelMap /            / ab,fi)
XeSm     = MICRO(ng1, FuelMap /54635,62649/ ab)
DetNeu   = MICRO(ng1, DetArea /92235      / fi)        !For neutron detector  !

CurCorner = CUR(ng2, NW,NE,SE,SW)                      !Discontinuity factors !
CurSides  = CUR(ng2, Nside,Eside,Sside,Wside)          !Discontinuity factors !
```

The short input that is shown below first calls the above described sets and defines almost all the parameters. Then each of the four pin-cell types, $*cell∗*, gets a pin assigned: a thin pin, a normal pin, a burnable-absorber pin subdivided into six annular regions, and a water pin. Next, these $*cell∗* parameters are used to define suitably-named structures, whose pin regions are filled with the proper materials by defining the enrichment distribution $*fuels*. Now the definition of $*prow∗*, the four pin rows, produces a visual pin map. Finally, the PATH and TREE operators specify the burnup and branch-off calculations to be done.

The output to the HERMES data-base is further treated by the output processor ZENITH. Observe that information is sent to the data-base from which ZENITH can adjust the power map for gamma-smearing, and evaluate the fast-neutron fluence in clad and box.

Of course, in standard applications the input will differ from what is shown in this example. Most of the dimensioning parameters, along with the *pin∗* definitions, can be specified in the set '*bwr-4x4*'. Further, the detector should be either for neutrons or for gammas. And, last but not least, the system would have been a full BWR assembly.

```
'BWR-4x4' = CASE('~rjs/library/35/lib-adj.35'/'a.hrf'
                  'BWR-4x4,T=556,v=40,cr=0,pw=25.1,br=70=40(cr=1)')

&'BWR'    = SET('A.SETS'/SAMPLE;AURORA)              !Read the expert input  !

$k        = PAR( 1 )                                 !Coupling discretization!
$p        = PAR("1.5")                               !Pin pitch          [cm]!
$box      = PAR("6.2")                               !Box inner dimension    !
$bwt      = PAR("0.1")                               !Box wall thickness     !
$wide     = PAR("0.6")                               !Wide   gap, half-width !
$narrow   = PAR("0.4")                               !Narrow gap, half-width !
$clong    = PAR("1.4")                               !Length of centre piece !
$rp1      = PAR("0.50")                              !Thin   pin, clad radius!
$rp2      = PAR("0.56")                              !Normal pin, clad radius!
$rp3      = PAR("0.65")                              !Water  pin, clad radius!
$rp4      = PAR("0.56")                              !BA     pin, clad radius!

$tcool    = PAR(550)                                 !Coolant temperature [K]!
$tfuel    = PAR(925)                                 !Fuel    temperature [K]!
$pw       = PAR(25.1)                                !Power level      [W/gU]!

 pin1     = CCS(0.44,$rp1//fuel,clad)                             !Thin   pin!
 pin2     = CCS(0.50,$rp2//fuel,clad)                             !Normal pin!
 pin3     = CCS(0.55,$rp3//water,clad)                            !Water  pin!
 pin4     = CCS(0.20,0.30,0.35,0.40,0.45,0.50,$rp4//              !BA     pin!
                fuel,fuel,fuel,fuel,fuel,fuel,clad )

'1.8'     = STR($cell1)                               !Cell with a  thin  pin!
'2.0'     = STR($cell1)                               !Cell with a  thin  pin!
'2.4'     = STR($cell2)                               !Cell with a normal pin!
'2.6'     = STR($cell2)                               !Cell with a normal pin!
'3.0'     = STR($cell2)                               !Cell with a normal pin!
 WP       = STR($cell3)                               !Cell with a water  pin!
 BA       = STR($cell4)                               !Cell with a   BA   pin!

$fuels    = PAR('UO2-1.8'/*-'1.8'-*-fuel /
                'UO2-2.0'/*-'2.0'-*-fuel /
                'UO2-2.4'/*-'2.4'-*-fuel /
                'UO2-2.6'/*-'2.6'-*-fuel /
                'UO2-3.0'/*-'3.0'-*-fuel /
                'H2O-550'/*- WP  -*-water/
                'UO2-BA' /*- BA  -*-fuel )

$prow1    = PAR('1.8', '2.6', '2.6', '2.0')
$prow2    = PAR('2.6',  BA,   '3.0', '2.4')
$prow3    = PAR('2.6', '3.0',  WP,   '2.4')
$prow4    = PAR('2.0', '2.4', '2.4', '2.0')

 burn40   = PATH(/(v40u), 200, 500, 1 000, 2 000, 10 000/4, 30 000/5)

 branches = TREE(burn40/P,(v70u),(v40c)/
                              500, 2 000, 10 000/2, 22 000, 30 000)

'BWR-4x4' = RUN()
```

PATH(), THE BURNUP OPERATOR

Purpose

The PATH operator is used to specify what states (see STAT operator) are to be calculated at what burnup levels and/or after what decay steps. Although this operator can be used for burnup and most branch-off calculations, the latter should rather be specified by the TREE operator.

Usage

name = **PATH(** [*iis*] **/***b* [, *b*]… **)**

name	represents a calculational path to be executed by HELIOS.	
iis	specifies initial isotopics produced in the actual case, or it specifies restart isotopics from an earlier case,	*iis* = *bname*:*eb*, *iis* = *dbname*:*eb*.
b	specifies one or more calculational points,	*b* = [*opt*,] *stead*.

Where:

bname is the name of an earlier path in the actual case, along which the initial isotopics of *name* have been produced (blank or *name* if there is no earlier path);

dbname is the name of a DBMAT operator (see there) with unspecified burnup that retrieves restart materials from a data-base where they have been saved in an earlier case by a DUMP operator (see there);

eb is a burnup level on *bname*, or on the path in the case from which *dbname* gets the restart isotopics [MWd/t];

opt is any combination of the following four options, represented by letters, and not separated by delimiters:

N	-	negates *G*, only neutron calculations (default);
G	-	also gamma calculations;
P	-	negates *C* (default with burnup and decay);
C	-	negates *P*, extra flux calculations, but with updated equilibrium number densities (default after a state specification, does not affect decay);

stead is one of the following three specifications that activate a state, or one or more burnup or decay steps, with subsequent neutron (and gamma) flux calculations:

(*st*)	-	name of a state enclosed in parentheses;
ea[/*n*]	-	burnup to $\lvert ea \rvert$ in one [or *n*] steps (if $ea < 0$, no isotopes with instant equilibrium);
(*d*[/*n*])	-	*d* days of decay in one [or *n*] steps, enclosed in parentheses.

Rules for burnup and decay

1) Each burnup (|*ea*| or *eb*) must be in units of MWd/t (metric ton) relative to the beginning of its path, i.e., each path starts at zero burnup.

2) The burnups (|*ea*|) along the path *name* must be given in increasing order. The interpolation (*ea*/*n*) inserts *n*–1 equidistant burnups before |*ea*|. Together with |*ea*| they define the existing burnups.

3) A burnup step is defined as the difference between two successive existing burnups; if smaller than 0.001 MWd/t, it is neglected by HELIOS.

4) For the active state (*st*), HELIOS will calculate the number densities of the burnable isotopes after each burnup step, as well as the neutron (and gamma) fluxes. This is done with the predictor/corrector method described in Rules 5-7 below.

5) Predictor step: predictor number densities at the end of the step are evaluated from fluxes and number densities at the beginning of the step. With these densities, the predictor or P fluxes are calculated.

6) Corrector step: corrector number densities at the end of the step are evaluated from the P flux reaction rates—at the end of the step—and the number densities at the beginning of the step.

7) P fluxes and predictor-corrector averaged number densities are used as the final fluxes and number densities after the burnup step. These final P fluxes are not based on the final number densities.

   With option C in force, an extra flux calculation with the final number densities provides the corrected final C fluxes.

8) The equilibrium isotopes (e.g., Xe-135) are in instant equilibrium if $ea > 0$. In deviation from Rule 7, their number densities are not averaged. Instead, the number densities used during the corrector step are taken, which are based on the isotopic reaction rates of the P fluxes.

9) The equilibrium isotopes are treated as normal isotopes if $ea < 0$, and during decay; then they are not updated to instant equilibrium.

10) Decay must be specified by parentheses that enclose the total decay time *d* in days [and the number *n* of equal time steps], "(*d*[/*n*])". No time step must be smaller than 0.001 days.

11) During decay of the active state (*st*), it only matters which materials are present. Temperatures and density factors are irrelevant, while the power level is temporarily set to zero—10^{-10} W/g.

12) For the active state (*st*), HELIOS will calculate the number densities of the burnable isotopes after each decay step, as well as the neutron (and gamma) fluxes. In the flux calculations, the power will be that of *st*, whose temperatures and density factors are relevant again.

Rules for initial isotopics

1) The initial isotopics (*iis*) must be specified as the name of an earlier path (*bname*) or a DBMAT operator (*dbname*), followed by a colon and an existing burnup (*eb*) on that path or in the data-base from which *dbname* retrieves the restart isotopics; *eb* must be specified within 0.001 MWd/t.
2) If there is no earlier path, *bname*:*eb* must be omitted or the default *name*:0 must be specified, i.e. fresh—input—isotopics are used.
3) After a burnup step, the final number densities at *eb* (see Rule 7 for burnup and decay) define *iis*.
4) There can be multiple isotopics at *eb* because of update calculations of equilibrium number densities (Rule 7 for states, Rule 7 for options) or because of decay. In that case, the number densities prior to update calculations or decay at *eb* define *iis*.

Rules for states

1) A state (*st*) is activated by specifying its name in parentheses, "(*st*)". This assigns properties and a power level to the geometric system.
2) States with isotopic multipliers must not occur in the PATH operator (see the ISOX and STAT operators).
3) States at any nonnegative power level may be subjected to burnup. Even zero power (10^{-10} W/g) is allowed, although this is unphysical and almost always causes the case to fail in the B_1 calculations after the first burnup step.
4) The first point on the path *name* must activate a state.
5) At any point on the path, another state may be activated to replace the previous one. Thus, an uncontrolled fuel assembly may at any burnup be replaced by the same assembly with control rods.
6) At any point on the path, the isotopics (or burnup) of the materials that make up the active state are determined by *iis*, plus the changes accumulated along the path until that point.
7) If a state is activated by "(*st*)", HELIOS will first calculate its neutron (and gamma) fluxes twice, which provides the P and C fluxes. Both are based on initial number densities, but the equilibrium number densities in the latter have been updated with P flux reaction rates.
8) Branch-off calculations can be generated by activating extra states at the relevant burnups. This requires recalculations of states or multiple PATH operators, which can be avoided with the TREE operator (see there).

Rules for options

1) The options are local to a path; they activate/negate additional flux calculations and are represented by the four letters *N*, *G*, *P* and *C*.
2) The options may be given in any combination—not separated by delimiters—before a state, a burnup or a decay specification (*stead*). They must be separated from *stead* by a comma.
3) An option is in force either by default or since its specification until it is negated. Hence, "*G,ea/n, C,ea/p*" means that *G* is in force during *n+p* burnup steps, while *C* is in force during *p* steps.
4) Option *N* is default; it negates the gamma calculations of option *G*.
5) Option *G* activates the gamma calculations (see also the remark in Rule 6 below); it negates option *N*.
6) Option *P* negates the C flux calculations (Rule 7 for burnup, Rule 7 for states); it is default for burnup. It remains in force for burnups until negated, but it must be re-specified before each "(*st*)" if needed.

   If during burnup C fluxes are calculated, there will be neither gamma calculations nor output based on the P fluxes.
7) Option *C* activates extra C flux calculations. It is default with "(*st*)"; it can not affect "(*d*)"; it negates option *P*.

   After "(*st*)": the equilibrium number densities are updated with P flux reaction rates, which is followed by a flux calculation.

   During burnup: fluxes consistent with predictor-corrector averaged number densities are calculated.

Remarks

State burnup

Each path starts at zero burnup. However, at any point along a path, the burnup of the active state may differ from the burnup of the path. This depends on the burnups that the individual materials of the state have undergone in the present path and in possible earlier paths. Because in HELIOS the true burnup of a material is saved together with its composition, the burnup of an active state is known at any time.

C fluxes, and the options *P* and *C*

The main purpose of a C flux calculation is to update the equilibrium number densities, and then evaluate the C fluxes. In this updating, the latest available isotopic reaction rates are used, i.e., those of the P fluxes (Rule 7 for states, Rule 8 for burnup). With these C fluxes, improved equilibrium number densities can be obtained, together with improved C fluxes, and so on. Thus, the C flux calculation is merely an iteration step towards the true equilibrium densities and their fluxes.

When a state is activated its first flux calculation is based on its initial number densities and provides the P fluxes (Rule 7 for states). For equilibrium isotopes, these densities are often not consistent with the state. They could also belong to another state that represents the same fuel assembly, but before a change of power, control or void. Or they could represent a fresh assembly without equilibrium isotopes, while the actual state is at full power.

Therefore, the C flux calculation is default when a state is activated, even if there is no subsequent burnup or decay. This first iteration step towards instant equilibrium is usually sufficient. If the User wishes to iterate further, one or more very small burnup steps should be made, rather than repeated reactivations of the state (see the TREE operator, Remarks). On the other hand, such C flux calculations can be negated by the option *P*, i.e. by "*P*,(*st*)". Although this negates option *C* only for the activation of the actual state, it remains in force for the subsequent burnup calculations until negated.

During burnup, the equilibrium number densities are already based on the P fluxes (Rule 8 for burnup). The C flux calculation merely provides fluxes that are consistent with the predictor-corrector averaged number densities. The price of this consistence is a doubling of the number of flux calculations during burnup. Normally, the extra flux calculations improve only marginally on the results, which is why option *P* is default during burnup. If, for some reason, option *C* is in force—e.g., at burnups from which dumps for restarts will be made—there will be no output nor gamma calculations with the P fluxes. (This is not so with the P fluxes of a state specification; they will always have output and optional gamma calculations.)

Option *C* is ignored during decay. During burnup with $ea < 0$ it still causes C flux calculations with predictor-corrector averaged number densities. Instant equilibrium, however, is not enforced; the equilibrium isotopes are treated as normal isotopes, so that their build-up to equilibrium can be followed explicitly. If after decay or burnup with $ea < 0$ there is normal burnup, it is recommended to restore the instant equilibrium densities first, e.g., by redefining the state.

Examples

1a) *P*,(*st*), *C*,*ea*, …

In this example "(*st*)" gets no C fluxes. Until negated, the subsequent burnups and state activations—the latter by default—will have C flux calculations; the decays remain unaffected by *C*.

1b) *PNNG*,(*st*), *GC*,*ea*, … = *PG*,(*st*), *C*,*ea*, …

This example shows that any combination of options is accepted; the last ones are valid. The *N*'s are superfluous, since the second confirms the first, and both are negated by the *G* before "(*st*)". The *G* before *ea* is also superfluous; it is already in force since "(*st*)". Apart from *G*, this is the same as Example 1a). Until negated, *G* will generate gamma fluxes at all subsequent points on the path, decays included.

1c) $P, (st), \ldots$

In this example no C fluxes are calculated with "(*st*)", nor at later burnups (and decays). At all later state specifications, however, *C* remains the default, unless specifically negated by *P*.

1d) $C, (st), ea, \ldots = (st), C, ea, \ldots$

Both members are the same because the C flux calculation is default with "(*st*)". All the subsequent calculations will be as in Example 1a).

1e) $C, (d), ea, \ldots = (d), C, ea, \ldots$

Both members are the same because decay is not affected by *C*. All the subsequent calculations will be as in Example 1a).

1f) $(d_1), (d_2), \ldots \neq (\text{"}d_1 + d_2\text{"}), \ldots$

Both members ultimately produce the same number densities. The one on the left, however, implies two flux calculations, the one on the right only one.

1g) $(full), e_1/n_1, P, (half), -e_2/n_2, \ldots$

A fuel assembly is depleted at full power in n_1 steps to e_1. Next, at half power the approach to equilibrium is followed in n_2 steps to e_2. When changing to (*half*), the C flux calculation is negated to avoid instantaneous equilibrium.

2)
```
B1 = PATH( / (s), 5000)
```

State *s* is depleted in one step from 0 to 5,000 MWd/t; one (default) C flux calculation at the start; no gamma calculations. The default initial isotopics are *B*1:0, i.e., the fresh—input—isotopics at the very start of the run.

3)
```
B2 = PATH(  / G, (s), N, 5000)
B2 = PATH( B1 : 0  / G, (s), N, 5000)
B2 = PATH( B2 : 0  / G, (s), N, 5000)
```

Starting with the same isotopics as in Example 2, state *s* is depleted in one step from 0 to 5,000 MWd/t; one (default) C flux calculation at the start; two gamma calculations at the activation of *s*, with both the P and C fluxes. The initial isotopics may look different—default (*B*2:0), *B*1:0 and *B*2:0—but all refer to the same fresh isotopics.

4)
```
B3 = PATH( B2 : 5000 / CG, (s), N, 4000)
B3 = PATH( B2 : 5000 / G, (s), NC, 4000)
```

Starting with the final isotopics of path *B*2, state *s* is depleted in one step from 0 to 4,000 MWd/t. In fact, its physical burnup goes from 5,000 to 9,000 MWd/t. C flux calculations will be done throughout; gamma calculations only in connection with the P and C flux calculations of *s* at its activation. The two variants are equivalent, because the position of *C* before "(*s*)" is irrelevant, *C* being anyhow the default for "(*s*)".

5)
```
B4 = PATH( B3 : 0 / P, (s), CN,8000/8, G )
```

Starting with the same isotopics as in Example 4 (those at the end of path *B*2), state *s* is depleted in eight steps from 0 to 8,000 MWd/t—in fact from 5,000 to 13,000 MWd/t. No C flux calculation when *s* is activated, but all eight burnups will have *C* flux calculations. The *N* before the burnup steps is superfluous, it is default. The *G* at the end is illegal.

6)
```
                        1          2          3          4          5
B5 = PATH(B4:6000/(s), GC,2000/2,  P,(t),    11000/9,    (1.2),
                      (u), C,44000/6,  P,(0.5),    N,(s),  64000/20)
                        6          7          8          9          10
```

This example is summarized in Table 1, with *b* the points along the path and the burnups relative to its start—in fact 11,000 MWd/t for *s*.

Table 1. Calculational moments along the path of Example 6.

b	MWd/t	State	Flux	Gamma	Remarks
1	0	*s*	P/C	no/no	Activate *s*: *C* and *N* defaults at start.
2	1,000	*s*	P/C	no/yes	Two burnup steps, *C* in force;
2	2,000	*s*	P/C	no/yes	*G* in force but only with *C*-fluxes.
3	2,000	*t*	P/-	yes/-	Activate *t*: *P* negates *C*, *G* continues.
4	3,000	*t*	P/-	yes/-	Nine burnup steps:
:	:	:	:	:	*P* in force since *b* = 3;
4	11,000	*t*	P/-	yes/-	*G* with *P*-fluxes (no choice).
5	11,000	*t*	P/-	yes/-	Decay of 1.2 days.
6	11,000	*u*	P/C	yes/yes	Activate *u*: *C* local, *G* continues.
7	16,500	*u*	P/C	no/yes	Six burnup steps:
:	:	:	:	:	*C* has been redefined;
7	44,000	*u*	P/C	no/yes	*G* again with *C*-fluxes.
8	44,000	*u*	P/-	yes/-	Decay; *P* could have come at *b*=10.
9	44,000	*s*	P/C	no/no	Reactivate *s*: *C* is local, *N* negates *G*.
10	45,000	*s*	P/-	no/-	Twenty burnup steps:
:	:	:	:	:	*P* already in force since *b* = 8;
10	64,000	*s*	P/-	no/-	no more *G* fluxes since *b* = 9.

7)
```
B  = PATH(/(ab),E1/n1, (cd),E2/n2, (bd),E3/n3)
B1 = PATH(          / (ab),    E1 /n1)
B2 = PATH(B1: E1     / (cd),"E2-E1"/n2)
B3 = PATH(B2:"E2-E1" / (bd),"E3-E2"/n3)
```

The first operator *B* does the same as *B*1, *B*2 and *B*3 combined. This example could serve as a simple illustration of fuel reconstitution. First, the assemblies represented by the states *ab* and *cd* are depleted to *E*1 and *E*2−*E*1. Next, the assembly represented by the state *bd* and made up of parts of *ab* and *cd*, is depleted during *E*3−*E*2. The quotes indicate arithmetic; see the PAR operator.

Fig. 1 gives a schematic presentation of path *B*. The scale at the bottom shows the isotopics of the materials that make up the different parts of the three assemblies. These isotopics are denoted by the burnups to which the assemblies of which they were part have been exposed. In HELIOS, they are collectively stored in an isotopics depot at the points defined by: B:0, B:$E$1 and B:$E$2 (or at the points $B$1:0, $B$1:$E$1 and $B$2:"$E$2−$E$1" for the paths $B$1, $B$2 and $B$3).

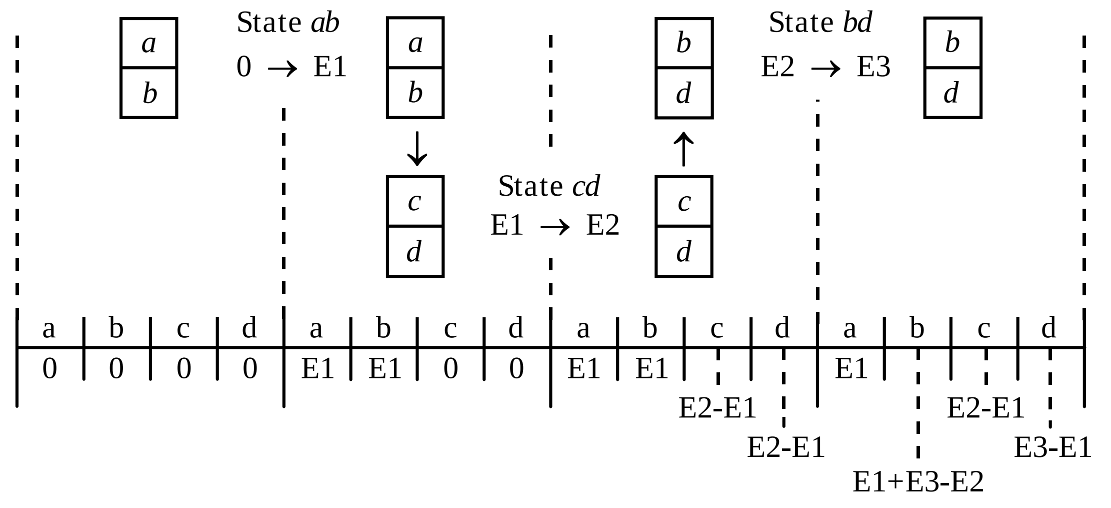


Fig.1: Schematic presentation of path *B* of Example 7.

8)
```
C1 = CASE(...)
...
Pa = PATH(/(st1),10000/10,(st2),20000/10)
Du = DUMP(df/area,Pa)
C1 = RUN()
C2 = CASE(...)
...
Db = DBMAT(df;C1;Du/(st2))
Bu = PATH(Db:12000/(St),E/n)
C2 = RUN()
```

First, case *C*1 is run. Dumps are made in the data-base *df* at 22 burnups, 11 for each state, including the first of *st*1 and *st*2 at 0 and 10,000 MWd/t, respectively. Next, case *C*2 is run whose burnup path *Bu* uses as initial isotopics those of state *st*2 on path *Pa* of case C1 at 12,000 MWd/t.

RUN(), THE END-OF-INPUT OPERATOR WITH OPTIONS

Purpose

The RUN operator signals the end of a case input. It can also be used to specify an input buckling and various options. The options can change default methods, accuracies, parameters of various iteration processes, trigger various test outputs, and adjust the output lines per page and the memory, if needed.

Usage

name = **RUN ([**OPT**:***opt* [,*opt*]… **[/**OPT:*opt* [,*opt*] … **]** … **])**

name	is the case identifier, which must be identical to *name* specified by the CASE operator.
OPT	specifies one of seven names representing an option type:
	BSQ signals an experimental input buckling;
	RES signals a User-defined choice of resonance categories;
	OUT signals User-requested test outputs;
	CP signals User-defined collision-probability accuracies;
	IT signals User-defined iteration parameters;
	MT signals User-defined methods;
	OM signals User-defined data for output and memory.
opt	specifies the values of the options of each type.

Rules in general

1) The sequence of the option types (OPT) is free, but no OPT may be specified more than once in a RUN operator.
2) Each OPT has a set of default options which can be changed by *opt*. Default values may be given explicitly, by their delimiters only, or not at all. Thus, "OPT:,,*opt*" changes the third option of OPT into *opt*; the preceding (and following) options keep their default values.

Rules for BSQ (one real number)

1) The BSQ option, the input buckling B^2, is in units of cm^{-2}. Its value must be in the range $|B^2| \leq 0.1$ cm^{-2}; its default value is 0.
2) HELIOS evaluates the leakage spectrum associated with B^2 (also if $B^2=0$) and its corresponding multiplication factor k,

$$k = \frac{\text{production rate from fission}}{\text{loss rates from absorption and leakage}}$$

where the leakage per energy group g is described by $D_g B^2 \phi_g$.

3) Depending on the second MT option (see there), the spectrum used for burnup and output reaction rates is that associated with the critical buckling, the input buckling, or $B^2 = 0$ (infinite medium).

Rules for RES (one integer)

1) The RES option specifies, either for the entire energy range or for nine energy sub-ranges, which resonance isotopes act together and which act independently in the spatial Dancoff-factor calculation.

2) Resonance isotopes that act together form a resonance category. A combination of categories is called a resonance set. The number of resonance categories can differ from one resonance set to another—see Table 1—but each set contains all the resonance isotopes exactly once.

Table 1. Sets of resonance categories in HELIOS.

Set	Cat	Description
1	1	All isotopes except natl Zr
	2	Natural Zr
2	1	Heavy-metal, fiss products
	2	Natural Zr
	3	Other isotopes
3	1	HM isotopes, FPs
	2	Natural Zr
	3	In-115, Hf, rest of Ag
	4	Ag-109, other isotopes
4	1	U-238
	2	Other HM isotopes, FPs
	3	Natural Zr
	4	Other isotopes
5	1	U-238
	2	Other HM isotopes, FPs
	3	Natural Zr
	4	In-115, Hf, rest of Ag
	5	Ag-109, other isotopes
6	1	Th-232
	2	U-238
6	3	Other HM isotopes, FPs
	4	Natural Zr
	5	Other isotopes
7	1	Th-232
	2	U-238
	3	Other HM isotopes, FPs
	4	Natural Zr
	5	In-115, Hf, rest of Ag
	6	Ag-109, other isotopes
8	1	Th-232
	2	U-238
	3	U-235
	4	Other HM isotopes, FPs
	5	Natural Zr
	6	Hf-177, Gd, Er
	7	In-115, other Hf, other Ag
	8	Ag-109, other isotopes
9	1-*n*	All isotopes individually, but all FPs together, and Pu-238 and Am-241 together

3) The mention of an isotope in Table 1 does not imply that HELIOS will treat it as a resonance isotope. To be treated as a resonance isotope, it must also have resonance data in the nuclear-data library.

4) The RES option assigns to the entire energy range one of the above nine sets of resonance categories, or to each of the nine intervals an individual set. The intervals, in lethargy and eV, are shown below.

```
u       9.21    10.13    11.11    12.21    12.79    13.35    13.82    14.73
 <---1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9--->
eV     1000      399      150     49.8     27.9     15.9     9.96     4.01
```

5) The sets per energy interval are converted into sets per library group. Each group gets the set whose lethargy interval contributes most to that group.
6) The RES option is a positive integer constant of one or nine digits. It represents the entire energy range or its nine intervals. Each digit represents one of the nine sets of resonance categories. The default value of RES is 4.

Rules for OUT (thirteen integers)

1) Output of the computer utilization per HELIOS module:
   - 0 - none (default);
   - ≥1 - cpu time, memory and working-file locations used.
2) Output of the resonance calculations:
   - 0 - none (default);
   - 1 - group-averaged fluxes, microscopic resonance cross sections (XSs) per group, flux-dip correction factors for out-scattering.
   - 2 - 1 & subgroup XSs, weights, and statistics of their fits per isotope;
   - 3 - 2 & macroscopic subgroup XSs and sources per category, used to calculate the equivalence XSs (Dancoff factors);
   - ≥4 - 3 & macroscopic subgroup fluxes and their corresponding equivalence XSs per category.
3) Output of the material XSs per energy group:
   - 0 - none (default);
   - 1 - nine sets of neutron XSs:
     1) total transport-corrected XS, absorption included
     2) removal XS, i.e. absorption + out-scattering
     3) ν*fission XS (ν is neutrons released per fission)
     4) fission XS
     5) κ*fission XS (κ is watt-seconds per fission)
     6) P_0-scattering XS
     7) P_1-scattering XS
     8) absorption XS
     9) fission spectrum, χ;
   - 2 - 1 & P_0-scattering matrices (in-scattering from all groups);
   - ≥3 - 2 & P_1-scattering matrices (in-scattering from all groups).
4) Output of the gamma data per group and per material:
   - 0 - none (default);
   - 1 - gamma sources;
   - 2 - 1 & (n, γ)-production XSs;
   - 3 - total transport, absorption and removal XSs;
   - 4 - 3 & gamma sources;
   - 5 - 3 & P_0-scattering matrices (in-scattering from all groups);
   - ≥6 - 2 & 5, i.e., all the above data.

5) Output of response-method data per group and per space element:

   100∗*opt*(resonance subgroups) + 10∗*opt*(gammas) + *opt*(neutrons)

   0 - none (default);

   1 - response fluxes due to sources (region-to-region) and in-currents in the angular surface sectors (sector-to-region);

   2 - response currents due to sources (region-to-sector) and in-currents in the angular surface sectors (sector-to-sector).

   ≥3 - 1 & 2, i.e. all the above data.

6) Output of the iteration performance of the transport calculations:

   100∗*opt*(resonance subgroups) + 10∗*opt*(gammas) + *opt*(neutrons)

   0 - none (default);

   1 - eigenvalue iteration report (about two lines per iteration);

   2 - 1 & thermal-rebalancing (fundamental-mode) factors;

   3 - 2 & inner iteration report (one line per iteration per group).

   ≥4 - 3 & final (converged) fluxes and fission sources.

7) Output of geometric collision-probability data per element type:

   0 - none (default);

   1 - exact and chord-integrated volumes and surfaces per sector;

   2 - 1 & macrobands and integration angles (see CP Rules);

   ≥3 - 2 & chord-integration information.

8) Output of collision probabilities per group and per space element:

   100∗*opt*(resonance subgroups) + 10∗*opt*(gammas) + *opt*(neutrons)

   0 - none (default);

   1 - weight factors used to normalize the collision probabilities;

   2 - first-flight collision, escape and transmission probabilities;

   ≥3 - 1 & 2, i.e. all the above data.

9) Output of symmetry data per space-element type:

   0 - none (default);

   1 - its representative element, symmetry fraction, number of independent region-to-region, sector-to-region and sector-to-sector probabilities, and the number of symmetric region and segment types;

   ≥2 - 1 & arrays of region, sector and collision-probability types.

10) Output of the local transport calculations. The exact calculation in a space element is a numerical integration of first-flight probabilities, followed by their conversion into multiple-flight responses, using the flat-flux approximation. If possible, one of the following faster methods is chosen: **(a)** exact for a full space element; **(b)** exact, utilizing existing symmetries; **(c)** exact for a geometrically symmetric part of a space element whose XSs cause a slight asymmetry, using XSs averaged over its symmetric parts, followed by a variational expansion to the full element; **(d)** variationally from earlier-calculated exact multiple-flight probabilities of the element itself; **(e)** varia-

tionally from earlier-calculated exact probabilities of another element of the same geometry type; **(f)** copying earlier exact probabilities of the element itself (only if its XS variations are zero or very small); **(g)** copying earlier exact probabilities of another element of the same geometry type.

- 0 - none (default);
- 1 - total number of times that each of these methods has been used in all elements and groups;
- ≥2 - method used and total-XS error per element and group.

11) Output of most of the arrays of the input treatment. It is a composite six-digit number; for each digit 0 is the default and >0 triggers the output:

    100,000∗A + 10,000∗G +1,000∗O +100∗M + 10∗R + Z

    - A - User input, read from the data-base (AURORA's output);
    - G - intermediate and final geometric arrays;
    - O - intermediate and final overlay arrays;
    - M - intermediate and final material-property arrays;
    - R - intermediate and final reflector-albedo arrays;
    - Z - intermediate and final administrative arrays for the output (the latter go to ZENITH).

12) Output of the array that controls the calculational phases:

    - 0 - none (default);
    - ≥1 - The destiny array with eight entries per action step:
      1) one of 11 actions:

1 - new state	2 - neutron flux	3 - gamma flux
4 - predictor step	5 - corrector step	6 - equilibrium step
7 - decay step	8 - output	9 - time-averaging
10 - dump isotopics		11 - dump for restart;

      2-8) state, dumps to read/write, burnup/decay, outputs.

13) Output of the output arrays:

    - 0 - none (default);
    - 1 - output lists and their catalog paths that go to the data-base;
    - ≥2 - 1 & administrative output arrays.

Rules for CP (three real numbers, one integer)

1) There are two types of first-flight probabilities. One is for source neutrons (or gammas) born in the regions of a space element; the other for in-currents in the angular sectors at its peripheral segments—see Fig. 2 of the CNX operator.

2) In each space element, both types are numerically integrated along sets of parallel chords, $y = y_i$. Each set corresponds to another direction φ between the element and the chords, or between each of the surface segments and

the chords; see Fig. 1.

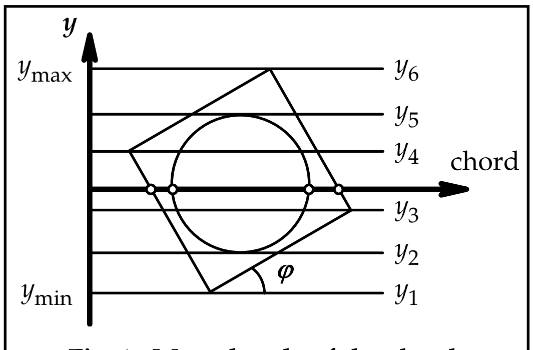


Fig. 1: Macrobands of the chords.

3) Each type has its own chords and angles. In fact, the sector probabilities have for each peripheral surface segment their own chords and angles.
4) To avoid numerical instabilities, the y-range is for each φ subdivided into macrobands, which are determined by the singularities in that direction; see Fig. 1. The chords within a macroband are equidistant and cross the same regions
5) In space elements with many regions, the amount of macrobands can become prohibitively large.
6) Therefore, there are two accuracies that control the chord spacing. The first specifies the maximum chord spacing within each macroband; the second specifies the minimum width of the macrobands.
7) There are also two accuracies that control the number of angles for the two types of probabilities. They define the maximum angular intervals. (The in-currents may have more angles, because all the regions and surfaces must be "visible" from each surface segment.)
8) An element's total XSs may be near to the XSs that have been used to integrate the CPs of an element of the same type, e.g. the same element at an earlier burnup. Then the CPs are obtained variationally from the integrated CPs at much reduced cpu times.
9) So there is also an accuracy that determines how to obtain the CPs, by numerical integration or variationally.
10) These five default accuracies can be changed with four options:
    - *opt1* - the maximum distance between two successive chords in cm, a real number whose default value is 0.1;
    - *opt2* - the minimum allowed width of the macrobands in cm, a real number whose default value is 0.001;
    - *opt3* - the relative XS difference below which variational CP calculations will be done, a real number whose default value is 0.01;
    - *opt4* - 1,000$*$*N1* + *N2*, an integer with a default value of 16,024, where *N1* and *N2* are the minimum number of integration directions φ in the interval $0 - \pi$ for sources and in-currents.

Rules for IT (nine real numbers, six integers)

1) The transport calculations are solved by two different types of iteration. The <u>inner iterations</u> couple, in one energy group at a time, the space elements by

interface currents. The outer iterations couple the groups by fission and in-scattering sources.

2) The outer iterations are subdivided into three levels, shown in Fig. 2. These levels are discussed in Rules 3-5.

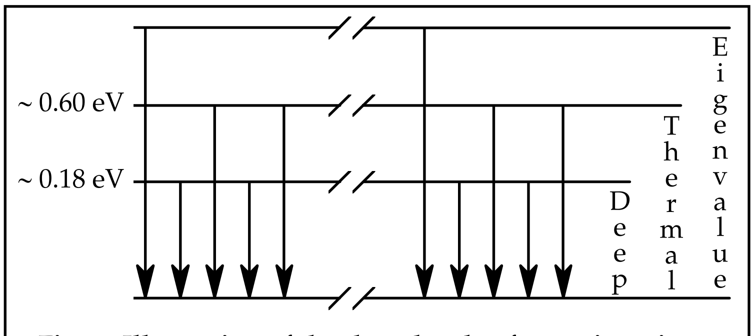


Fig. 2: Illustration of the three levels of outer iterations.

3) The eigenvalue iterations start with the normalization of the fission sources. This is followed by the inner iterations per group, which go from the first energy group to the last, i.e. from high to low energies. The eigenvalue iterations are repeated until convergence.

4) To accelerate convergence, additional thermal iterations are made per eigenvalue iteration, i.e. group-to-group outer-iteration sweeps that start at ~0.6 eV, where up-scattering is still worthwhile.

5) Further acceleration is achieved by a few extra deep-thermal iterations per thermal iteration, i.e. group-to-group outer iterations that start at ~0.18 eV, below which up-scattering is strong.

6) The relaxation of each fission source, current (inner iterations) and thermal flux (outer iterations) accelerates convergence as follows:

$$x^{(t)} = x^{(t-1)} + \omega[x' - x^{(t-1)}], \qquad \text{with } 0 < \omega < 2.$$

$x^{(t)}$ and x' are a source, current or thermal flux in the t'th iteration with and without relaxation, and ω is the relaxation factor.

7) Convergence is reached as soon as $| x^{(t)} - x^{(t-1)} | \leq \varepsilon x^{(t)}_{\max}$, for each x. Here x is any current in the iterations, or the eigenvalue k (multiplication factor of the system), while $x_{\max}$ is the maximum of all the currents in its iterations, or k.

8) The critical or material buckling (B_m^2), the migration area, the diffusion coefficients and the criticality (leakage) spectrum are evaluated by the B_1-method. This method involves an iteration process without relaxation.

9) The normalization weights of all the first-flight probabilities are obtained iteratively to preserve their balance and reciprocity relations.

10) The User can change the following four convergence criteria, five relaxation factors, and six limits on the number of iterations (see also Remarks):

opt1	-	$\varepsilon 1$:	inner iterations (currents),	default is 2.5E-4;
opt2	-	$\varepsilon 2$:	eigenvalue iterations (k),	default is 2.0E-5;
opt3	-	$\varepsilon 5$:	B1-iterations (B_m^2),	default is 1.0E-5;
opt4	-	$\varepsilon 11$:	normalization weights (colprobs),	default is 1.0E-5;

opt5	-	$\omega 1$:	inner iterations (currents),	default is 1.0;
opt6	-	$\omega 2$:	eigenvalue iterations (fission sources),	default is 1.3;
opt7	-	$\omega 3$:	thermal iterations (fluxes),	default is 1.4;
opt8	-	$\omega 4$:	deep-thermal iterations (fluxes),	default is 1.4;
opt9	-	$\omega 11$:	normalization weights (colprobs),	default is 0.7;
opt10	-	$it1 = 1{,}000 * it_{\text{extra}} + it_{\max}$:	inner iterations (currents),	default is 100 (CP) or 30 (CX);
opt11	-	$it2 = 1{,}000 * it_{\text{extra}} + it_{\max}$ (CP); $it_{\max}$ (CX):	eigenvalue iterations (k, fission sources),	default is 75;
opt12	-	$it3$:	thermal iterations (fluxes),	default is 4;
opt13	-	$it4$:	deep-thermal iterations (fluxes),	default is 2;
opt14	-	$it5$:	B1-iterations (B_m^2),	default is 50;
opt15	-	$it11$:	normalization weights (colprobs),	default is 14.

Rules for MT (three integers)

1) The symmetry of a space element depends on its geometry and its materials. Even regions with initially the same burnable material may differ with burnup. Insofar as possible, these symmetries are utilized to speed up the evaluation of the first-flight probabilities.

2) The system calculated will be part of a critical reactor. Thus, instead of the spectrum inherent in the transport calculations ($k \neq 1$), a criticality spectrum will be imposed ($k = 1$) before obtaining the burnup reaction rates and output XSs from energy-group condensation.

3) This criticality spectrum is calculated with the B_1-method, in which criticality is obtained by the group-dependent leakages $D_g B_m^2 \phi_g$.

4) The microscopic P_0 and P_1-scattering cross sections (XSs) in energy group g, are related to the P_0 and P_1-scattering matrices as follows:

$$\sigma_{sg}^{n} = \sum_{g'} \sigma_{g' \leftarrow g}^{n} \;, \quad (n = 0, 1).$$

5) The transport correction reduces the P_0-scattering XS and the diagonal of the P_0-scattering matrix by the P_1-scattering XS, i.e.

$$\sigma_{sg}^{tr} = \sigma_{sg}^{0} - \sigma_{sg}^{1} \text{ and } \sigma_{g' \leftarrow g}^{tr} = \sigma_{g' \leftarrow g}^{0} - \delta_{g'g} \sigma_{sg}^{1} \;,$$

with $\delta_{g'g}$ the Kronecker delta, which is 1 if $g' = g$ and 0 otherwise.

6) In some libraries the transport correction is not fully implemented, in particular in the highest energy groups. This has been done to avoid negative diagonal elements that might cause negative fluxes. In practical cases, however, this problem has never been observed.

7) HELIOS enforces the exact transport correction for isotopes with P_1 data. For the other isotopes, the transport-corrected scattering XS of the library is used to construct a diagonal P_1-matrix, i.e.

$$\sigma^1_{g'\leftarrow g} = \delta_{g'g}(\sigma^0_{sg} - \sigma^{tr}_{sg}) .$$

HELIOS libraries since 1994 have the exact transport cross sections built in.

8) Three options can switch off the defaults for use of: existing space-element symmetries, the criticality spectrum and the exact transport correction.

opt1 - 0: symmetry is used for first-flight probabilities (default);
- $\geq$1: symmetry is not used for first-flight probabilities.

opt2 - 0: criticality spectrum for burnup and output (default);
- 1: the spectrum associated with the input buckling, or its default value zero, is used for burnup and output;
- $\geq$2: the zero-buckling (infinite-medium spectrum) is used for burnup and output.

opt3 - 0: the exact transport correction is enforced (default);
- $\geq$1: the transport correction from the nuclear-data library is used, without attempting to make it exact (this option has no effect if the library has an exact transport correction built-in).

Rules for OM (six integers)

1) The number of output lines printed per page. The default is 56.

2) A rather conservative estimate is made of the memory requirements for the geometric input, which includes a safety factor whose default value is 2. In the unlikely event that this factor is insufficient, an error message will request an increased value. This has nothing to do with option 3 below.

3) The size of the working area in memory, *MMRY*. The default is 50,000,000. This is the maximum size reserved for the active arrays used in the calculations in any module. If it is not sufficient, an error message is printed.

   The User may economize the use of the memory reducing *MMRY*. Setting the first OUT option to 1 will cause a printout of the memory used by each module.

4) The size of the character working area in memory, *CMMRY*. The default is 5,000,000. This is the maximum size reserved for the active character arrays used in the calculations in any module. If it is not sufficient, an error message is printed.

   The User may economize the use of the memory reducing *CMMRY*. There is no OUT option that prints this number, so it will have to be done by trial and error.

5) The size of (part of) the scratch area that is in memory, *SMMRY*. The default is 75,000,000, its minimum value is 256. The scratch area contains data (arrays) that are re-used in other modules, or in the same module at a later time. Part or all of this area can be simulated in memory to reduce clock time; the remainder is stored in a scratch file.

The introduction of *SMMRY* has caused significant reductions in clock time on networks. If the User wishes to customize *SMMRY*, its actual value in any module can be found by setting the first OUT option to 1. This causes a printout of the total size of the scratch file used by each module. If not the entire scratch file could be simulated in memory, a message will appear at the end of the run that shows the size of the scratch file.

6) After HELIOS-1.8: Once the input of a case has been treated, the data-base with the results is kept open until the case is finished. This reduces clock time for data-bases with very many output lists because it avoids unnecessary opening and closing of the data-base each time that output must be written into it. If there is a DUMP operator, the data-base is closed and the dump file is opened whenever number densities must be saved in the dump file. Once the number densities are saved, the dump file is closed again and the data-base is re-opened.

   With this default strategy (option = 0) it is not possible to peek into the data-base with HFCARE to see how far the case has proceeded. By setting this option to a positive integer number, the default is switched off and the old strategy is used—the data-base is only open when output is written into it—which allows the User to peek into the data-base while the case is run.

Remarks

For most of the options, their default values ought to be sufficient in standard HELIOS applications. Only the BSQ input, and maybe the OM option for the number of output lines might be of interest to the User.

The RES option should only be used to study the effects of mutual shielding of resonance isotopes on the Dancoff factor. The default resonance set 4 corresponds to independent spatial-shielding (Dancoff) calculations for all the heavy resonance isotopes on the one hand, and all the other resonance isotopes on the other. Thus, e.g. Ag-In-Cd control pins will not be affected by fuel pins and *vice versa*. For MOX and UO_2 lattices, RES=4 is advised; for lattices with Th, RES=6.

The OUT options are chiefly meant as a help for the programmer in finding errors, or in explaining an unexpected behaviour of HELIOS. Some of the outputs are esoteric and will be difficult to understand. Most of the options will also produce vast amounts of output and should only be invoked for tests on small systems without burnup.

The CP defaults have been chosen to give good-quality results without unduly taxing the cpu time. They may have to be modified for odd geometries or complex space elements.

If applicable, the IT options are used for neutron, gamma and resonance-subgroup calculations alike. Their defaults should be adequate for most cases. The thermal and deep-thermal iteration limits merely affect the efficiency of the eigenvalue iterations. If exceeded, they will not break off the calculations.

Exceeding the other iteration limits often signals an input error, e.g. absurd material densities or an incorrect geometry. It may also happen in unusually large or complex cases, or when libraries with more than about fifty groups are used. If it is not an input error, the User should try to modify relaxation factors and convergence criteria before trying to change iteration limits. Particularly, if the inner iterations do not converge, the relaxation factors *ω3* and *ω3* should be set to 1. If that does not help, under-relaxation should be tried for *ω1, e.g.* 0.8.

If convergence is very slow, the difference between two successive iterands can become so small that the criterion of Rule 7 for IT is met before a sufficiently converged solution is obtained. This can be remedied by either sharpening the convergence criterion *ε*, or enforcing an extra number of iterations after the convergence criterion is satisfied.

Examples

```
1) 'KRITZ-KO544B' = RUN(IT:0.001,,,,0.8, ,,,,, ,,4/BSQ:0.00039)
```

In this example, a critical experiment is evaluated, which is why an experimental buckling has been specified. Further, the convergence criterion of the inner (current-coupling) iterations has been loosened to 0.001, and an under-relaxation factor of 0.8 will be used. Finally, there will be four deep-thermal iterations per thermal iteration instead of two.

```
2) 'Q96.250E,40V' = RUN(OUT:,4,1,,, 32/RES:122 222 331 )
```

In this example, full test output of the resonance calculations and output of the materials' neutron XSs are requested. For gamma calculations there will be inner and outer-iteration reports, while for neutron calculations there will be an outer-iteration report including output of the thermal-rebalancing factors.

Further, different resonance sets have been specified for the resonance calculations in the different energy ranges. It will now be explained how this option is interpreted in the case of a library with the five resonance groups shown below in lethargy and energy.

```
u    7.01    9.90   11.51   12.72   13.82   14.73
 <----|---1---|---2---|---3---|---4---|---5---|---->
eV  9000     500     100      30      10       4
```

According to Rules 4 and 5 for RES, the intervals that contribute most to the five group-lethargy intervals above are 1, 3, 4, 6 and 8. Thus, the resonance-group/resonance-set assignment becomes 1/1, 2/2, 3/2, 4/2 and 5/3.

```
3) xx = RUN(OM:,, 1 500 000,  !  MMRY: Work area                !
                    700 000,  ! CMMRY: Work area for characters !
                 20 000 000)  ! SMMRY: Scratch area in memory   !
```

This example is explained by its own comments. If any of the entries is not present, the following default value will be used:

```
MMRY  = 50 000 000; CMMRY =  5 000 000; SMMRY = 75 000 000
```

SET(), THE INCLUDE OPERATOR

Purpose

The SET operator has a dual purpose. As HELIOS input, it retrieves sets of input operators from a data-base and merges them with the User's input, thereby transforming it into the expanded input. As set-management input, the SET operator accesses a data-base to create, delete, list and restore these sets. The SET operator plays the same role for the input of the output processor ZENITH.

Usage

&*name*	= **SET (** *df* **/ [***ct* [; *ct*] … **])**	HELIOS/ZENITH input
&*ADD*	= **SET (** *df* **/ [***ct*; [*ct*;] … **]** &*name* **)**	Set management
&*DELETE*	= **SET (** *df* **/** *csp* **[** / *csp***]** … **)**	Set management
&*LIST*	= **SET (** *df* **/** *op* **/** *cs* **[** / *cs***]** … **)**	Set management
&*RSTORE*	= **SET (** *df* **/** *uf* **/** *csp* **[** / *csp***]** … **)**	Set management

&*name*	represents an earlier defined set of input operators.	
df	specifies a data-base file with the set &*name*.	
ct	is the name of a catalog in *df*.	
csp	specifies either a catalog path in *df*,	*csp* = **[***ct* [; *ct*] … **]**
	or a set path in *df*,	*csp* = **[***ct*; [*ct*;] … **]** &*name*
op	is 0/1/2 for a short/full/path list.	
cs	specifies:	
	op < 2 - a path to a catalog or to a set,	*cs* = *csp*
	op = 2 - the name of a catalog or of a set,	*cs* = [*ct*] or &*name*
uf	is the name of a User-specified file.	

Rules in general

1) The first character of the left-hand side of a SET operator must be an ampersand (&).

2) A set name (&*name*) must begin on an ampersand, which must be outside possible apostrophes that enclose *name*. &*name* must not exceed 80 characters, including the ampersand, but excluding the possible apostrophes.

3) A file name, i.e., *df* or *uf*, must not be given as a parameter ($*file*).

4) A set may contain either HELIOS or ZENITH input operators, but not the CASE and RUN operators.

5) &*name* reads from the data-base (*df*) the set &*name* defined by its set path ([*ct*; [*ct*;] …] &*name*). This set expands the User's input with its own input operators, which are inserted directly after the "&*name* = SET" operator.

6) As a HELIOS/ZENITH input operator, the "&*name*=SET" operator itself can be part of a set, thereby including another set. This can be repeated to at most nine levels: set 1 includes set 2, which includes set 3, … which includes set 9. All the involved sets must be unique.
7) <u>&*ADD*</u> creates in a specified data-base a set with a specified set path.
8) All the HELIOS or ZENITH input operators that follow after &*ADD*, until the next &*ADD*, &*DELETE*, &*LIST* or &*RSTORE*, or until the end of the input, will be stored in the set.
9) <u>&*DELETE*</u> deletes from a specified data-base either all the sets and catalogs below a catalog defined by its catalog path ([*ct* [; *ct*] …]), or a single set defined by its set path.
10) <u>&*LIST*, *op* = 0</u>, prints short lists of either all the sets in a specified data-base below a defined catalog, or just one defined set. A short list consists of the name of a set and its creation date.
11) <u>&*LIST*, *op* = 1</u>, prints full lists of either all the sets in a specified data-base below a defined catalog, or just one defined set. A full list consists of the name of a set, its creation date and the input operator(s) stored in the set.
12) <u>&*LIST*, *op* = 2</u>, lists for a specified data-base the paths of either all the sets below all the catalogs of a specified name (*ct*), or all the sets of a specified name (&*name*).
13) <u>&*RSTORE*</u> writes the contents (input operators) of either all the sets in a specified data-base below a defined catalog into a User-specified file *uf*, or just one defined set.
14) The &*ADD*, &*DELETE*, &*LIST* and &*RSTORE* SET operators must be used only in the set-management input.

Rules for catalogs and sets

1) A catalog in a data-base is defined by a catalog path, which is a chain of catalog names (*ct*) separated by semicolons (;). The last name in the chain must be that of the catalog itself.
2) A catalog is said to be, or to reside, below its predecessors in its defining chain.
3) The root (main) catalog of a data-base has no name and no path.
4) A set in a data-base may reside below any catalog. Its location is defined by its set path which is the catalog path, if any, followed by the set name. A semicolon must separate the possible catalog path from the set name.
5) Any number of catalogs and/or sets may be defined directly below a catalog if their names are unique. Otherwise, catalogs below different catalogs may have the same name, and so may sets. (Note that set and catalog names will always differ because of the ampersand.)

Remarks

The concept of predefined sets of input operators offers the User a possibility to facilitate the preparation of the HELIOS and ZENITH input. For example, sets of predefined geometry operators can be used in all calculations on the same type of fuel assembly, or a set of MAT operators could define all the commonly used materials.

While many sets can be included in the same input—even sets containing SET operators—the input-operator sequence in the expanded input must be correct. So, one should distinguish between sets for material, geometry, overlay and path input. These then can be combined freely when specifying the HELIOS or ZENITH input.

The flexibility of the sets is improved if used together with parameters (see the PAR operator). For instance, a set could contain the entire geometry input of an 8×8 assembly, leaving the pin radii and the pitch undefined as parameters. In her/his input, the User then has to define only these parameters.

Example

```
&ADD    = SET('db.dat'/&Pcell1)
pin1    = CCS(0.41//)
&Pcell2 = SET('db.dat'/Cat1)
cell1   = STR((0,0)(0,1.1)(1.1,1.1)(1.1,0)/4/pin1(0.55,0.55))

&ADD    = SET('db.dat'/Cat1;&Pcell2)
pin2    = CCS(0.42//)
&Pcell3 = SET('db.dat'/Cat1;Cat2 )
cell2   = STR((0,0)(0,1.2)(1.2,1.2)(1.2,0)/4/pin2(0.60,0.60))

&ADD    = SET('db.dat'/Cat1;Cat2;&Pcell3)
pin3    = CCS(0.43//)
cell3   = STR((0,0)(0,1.3)(1.3,1.3)(1.3,0)/4/pin3(0.65,0.65))

&ADD    = SET('db.dat'/Cat1;Cat3;&Pcell4)
pin4    = CCS(0.44//)
cell4   = STR((0,0)(0,1.4)(1.4,1.4)(1.4,0)/4/pin4(0.70,0.70))

&ADD    = SET('db.dat'/Cat1;&Pcell5)
pin5    = CCS(0.45 // )
cell5   = STR((0,0)(0,1.5)(1.5,1.5)(1.5,0)/4/pin5(0.75,0.75))

&LIST   = SET('db.dat'/1/)
&RSTORE = SET('db.dat'/'set.dat'/Cat1/Cat1;Cat2;&Pcell3)
&LIST   = SET('db.dat'/2/Cat1/&Pcell4)
&DELETE = SET('db.dat'/Cat1;Cat2/Cat1;&Pcell2)
&LIST   = SET('db.dat'/0/Cat1)
```

This example of set-management input uses all the SET operators. The database file used throughout is '*db.dat*'. First, five sets are put into an empty database, which results in the data-base structure of Fig. 1. The set &*Pcell1* is stored directly below the root catalog, while the set &*Pcell2* is stored directly below the catalog *Cat1* which is located below the root catalog. The set &*Pcell3* is stored below the catalog *Cat2* which in turn is located below *Cat1*. Further, &*Pcell4* is below *Cat3* which is another catalog below *Cat1*. Finally, &*Pcell5* is below *Cat1*.

Observe that the set &*Pcell1* contains a SET operator that includes &*Pcell2* among its operators. In its turn, &*Pcell2* includes &*Pcell3*. Thus, when used in the HELIOS input, &*Pcell1* will be three levels deep.

Next, a full list is printed for all the sets below the root catalog, i.e. all the sets of the data-base '*db.dat*'. Thereafter, the contents of all the sets below *Cat1* (&*Pcell2*, &*Pcell3*, &*Pcell4* and &*Pcell5*) and of &*Pcell3*, this time explicitly specified by its set path, are stored in the User-specified file '*set.dat*'.

The second &*LIST* causes a printout of the set paths of all the sets below all the catalogs named *Cat1* and of all the sets named &*Pcell4*.

&*DELETE* causes all the sets and catalogs that are below *Cat2*—defined by the catalog path "*Cat1*; *Cat2*"—to be deleted, as well as the explicitly defined set "*Cat1*; &*Pcell2*". Notice that in this example there are no catalogs below *Cat2* to be deleted.

The final &*LIST* will produce a short list of all the sets that have remained below *Cat1* after the &*DELETE*, i.e. &*Pcell4* and &*Pcell5*.

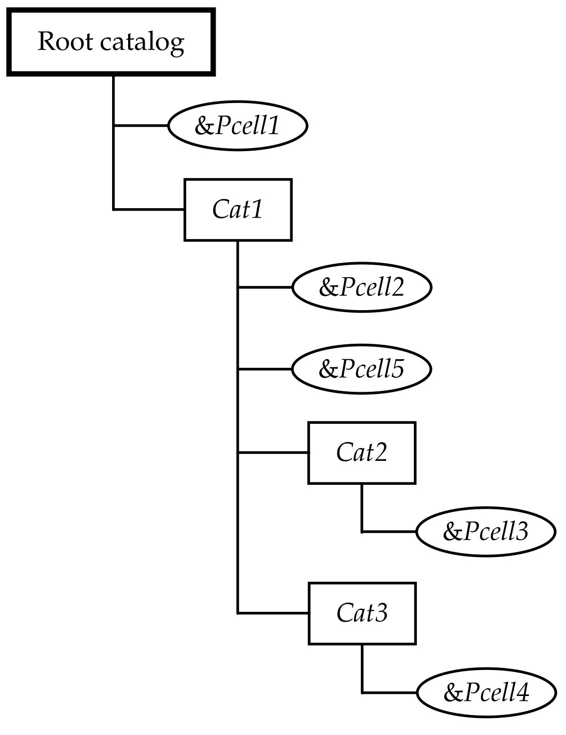


Fig. 1: Data base with the five sets of the example.

STAT(), THE STATE OPERATOR

Purpose

The STAT operator is used to define a state. A state is a complete set of property configurations that fill the system, so that it can be calculated by HELIOS. It also includes—default or User-specified—isotopic multipliers and the power level. States are used by the PATH and TREE operators to specify the properties of the system to be calculated. States with User-specified isotopic multipliers, however, may not be used in PATH operators.

Usage

name = **STAT (** *mos* , *dos*, *tos* [, *ix*] [, *pw*] **)**

name	represents a state that can be calculated by HELIOS.
mos	is the name of a material-overlay set—see OVSM operator.
dos	is the name of a density-overlay set—see OVSD operator.
tos	is the name of a temperature-overlay set—see OVST operator.
ix	is the name of a combination of isotopic multipliers, defined by the ISOX operator.
pw	is the power level of the system in W/g (initial heavy isotopes).

Rules

1) Apart from the burnup level, the state *name* assigns to the geometric system all the properties that HELIOS needs to calculate it.
2) The overlay sets for the materials (*mos*), density-correction factors (*dos*) and temperatures (*tos*), as well as the combination of isotopic multipliers (*ix*) must all be specified by name.
3) The *ix* may be omitted. Isotopes without multiplier get the default value 1.0.
4) States with an *ix* must not occur in PATH operators, nor must they be used for decay in TREE operators.
5) The power level ($pw \geq 0$) must be in units of W/g of initial heavy isotopes. If omitted, or if $pw < 10^{-5}$, the default value of 10^{-10} W/g will be used; this is also the built-in minimum value of *pw*.
6) It is not necessary to specify a special state at zero power, room temperature and zero void for the purpose of decay only, i.e., without flux calculations (see the PATH and TREE operators).

Remarks

Non-burnable materials always have the same composition. Density-correction factors and isotopic multipliers are defined by *dos* and *ix*. For burnable materi-als, however, the number densities of their burnable isotopes vary with time. In that case, it is not enough to know which material is assigned to which region; its burnup level must also be specified. This additional, and final, information is provided in the PATH and TREE operators—see there.

Examples

```
1) 'HFP,  CR=1'  = STAT('CR=1', D40, '950/559', 25.2)
2) 'HFP,  CR=0'  = STAT('CR=0', D40, '950/559', 25.2)
3) 'COLD, CR=0'  = STAT('CR=0', Dcld, TRoom, 0)
4) 'Sm149=0'     = STAT('CR=0', Dcld, TRoom, 'No Sm149')
```

The first two examples are hot-full-power states with and without control rod, at 40% void, operating temperatures (T-fuel = 950 K, T-coolant = 559 K) and at a power of 25.2 W/g. In the third example, the state represents a cold fuel assembly at room temperature and zero power—HELIOS will set the power to the built-in minimum value of 10^{-10} W/g. The fourth example could be the same fuel assembly, but now with the Sm-149 set to zero by the ISOX operator '*No Sm149*'.

STR(), THE STRUCTURE OPERATOR

Purpose

The STR operator is used to define a geometric structure, like a pin cell, a water gap and a control blade. The structures are the building blocks, which can be combined by the CNX operator to describe the geometry of the system, e.g., a fuel assembly.

Usage

name = **STR (** *n* [[,] *n*]… **/** *p* **/** [*c* [, *c*]…] **/** [*r* [; *r*]…] **)**

name	represents a geometric structure that is defined by its closed periphery of straight-line segments and its contents of CCS-es and (flat-flux) regions or meshes.	
n	specifies a node,	$n = (x, y)$
p	specifies the periphery,	$p = p_n$ [, *bid*]
c	specifies a CCS,	*c* = *cname* (*cx*, *cy*, *rot*) [n_p]
r	specifies a non-CCS region,	*r* = *j*,…*k* [, *rid*]

Where:

(x, y)	are the coordinates (cm) of a node, in parentheses;
p_n	is the number of nodes that define the periphery;
bid	is the name of the background region that initially fills the entire inside of the periphery;
cname	is the name of a CCS inside the structure;
(*cx*, *cy*, *rot*)	are the centre coordinates (cm) of the CCS and the rotation angle, in parentheses;
n_p	is the number of CCS regions inside the structure (only for a partial CCS and then it is compulsory);
j,…*k*	are the sequence numbers of the nodes that delimit a non-CCS region inside the structure, i.e., nodes $n_j, \ldots n_k$;
rid	is the name of a non-CCS region defined by *j*,…*k*.

Rules for nodes

1) Each structure has an individual perpendicular coordinate system; its location with respect to the origin is up to the User.

2) The minimum allowed distance between any two nodes in the list is 10* AQRACY; presently, AQRACY = 1.0E-04.

3) The p_n nodes that define the periphery must be the first in the list of nodes, i.e., $n_1, \ldots, n_{p_n}$.

4) The sequence of the remaining nodes is irrelevant. Together with the first p_n nodes, they may be used to define non-CCS regions.

5) Nodes may be defined anywhere on a segment, a CCS radius, or at a CCS centre.

6) Nodes that are not used by the STR operator can also be defined. This practice is not recommended, unless such nodes have to be used later, e.g., by a CNX operator; see also Rule 6 for CCS-es, and Fig. 2.

Rules for the periphery

1) Any periphery made up of non-intersecting line segments is allowed. The periphery may be re-entrant—some of the interior angles may be greater than π; see Fig. 1.

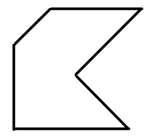


Fig. 1: Example of a re-entrant polygon.

2) Two consecutive nodes must define the begin and end points of a line segment. The periphery must change direction at each of these nodes. The peripheral segments may be further subdivided into more (flat-current) segments, i.e., they may contain more nodes.

3) The structure must lie to the right of the peripheral line segments.

4) Because the list cannot contain equal nodes, HELIOS closes the periphery by assuming that the last segment goes from n_{p_n} to n_1.

5) If *bid* is omitted, HELIOS will use '*1*' as name.

6) The background region *bid* can be partly replaced by CCS-es and/or non-CCS regions (see below).

Rules for CCS-es

1) CCS-es are dominant, i.e., they must not contain alien regions while they replace part of the non-CCS region inside which they occur; see Rule 8 for non-CCS regions, and Figs 3d-e and 4.

2) A partial CCS can be defined by intersecting a CCS with at most two peripheral line segments that must pass through its centre. Thus, while a half CCS is cut by one segment, any other partial CCS will have its centre at one of the peripheral nodes; see Figs 2 and 6.

3) The number of regions actually present in a CCS, n_p, must be specified if and only if the CCS is partial.

4) In a partial CCS, the region names are the *cid*'s of its generic CCS. For example, the partial CCS in Fig. 1b of the CCS operator has as default *cid*'s the region numbers '1', '2', '3', '4', '9' and '10'. However, their sequence numbers in HELIOS will be from 1 to 6.

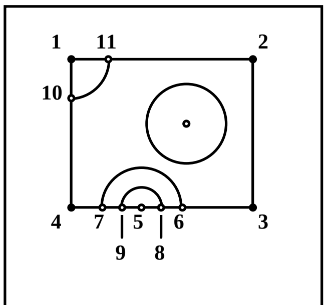


Fig. 2: CCS-es and nodes.

5) Only a partial CCS can have nodes inside its outer radius that can be used. Such nodes must be located at its centre or at the intersections of its radii with the periphery. This is shown in Fig. 2, where $p_n = 4$. The solid nodes 1 through 4 must be specified—they define the periphery. Although the hollow nodes 5 through 11 are not used here, they can be used later, e.g., in a CNX operator—of course, nodes on the outer radius of a CCS can always be used. The centre of an entire CCS may be specified but can never be used.
6) If the centre of a partial CCS, or any of the intersections of its radii with the periphery of the structure, will be needed later on, e.g., in a CNX operator, they must be specified in the STR operator.
7) The same generic CCS may be assigned to various positions inside a structure.
8) Rotation angles are specified in degrees clockwise from north. For complete CCS-es, the angle rotates the north pole clockwise; for partial CCS-es, the azimuthal segmentation is rotated, but north is unaffected.

Rules for non-CCS regions

1) A region outside a CCS is a non-CCS region.
2) The border segments of a non-CCS region must be line segments and/or CCS arcs.
3) The border nodes $j,\ldots,k$ must be unique for each non-CCS region.
4) Two consecutive nodes among $j,\ldots,k$ define the begin and end point of a border segment. In these points, the border must change direction or the border segment must be an entire arc segment.
5) The region must lie on the right of the border segments.
6) Because the nodes $j,\ldots,k$ must be unique, HELIOS closes the periphery by assuming that the last segment or arc goes from n_k to n_j.
7) Any non-CCS region but *bid* must be defined inside another non-CCS region, replacing part but not all of it, i.e., non-CCS regions must not intersect or disappear; see Figs 3a-c and 4.
8) Apart from the background region *bid*, a non-CCS region may overlap only entire CCS-es, which remain intact and replace part of the non-CCS region; see Rule 1 for CCS-es, and Figs 3e, 4 and 6.
9) If *rid* is left out, HELIOS uses the sequence number among the non-CCS regions as name, where *bid* is the first non-CCS region.

10) The User-defined *bid* and *rid*'s do not have to be unique. In fact, for the overlay and output operators, it will be helpful to give regions that will contain the same type of material the same name, i.e., the same *bid* and/or *rid*.

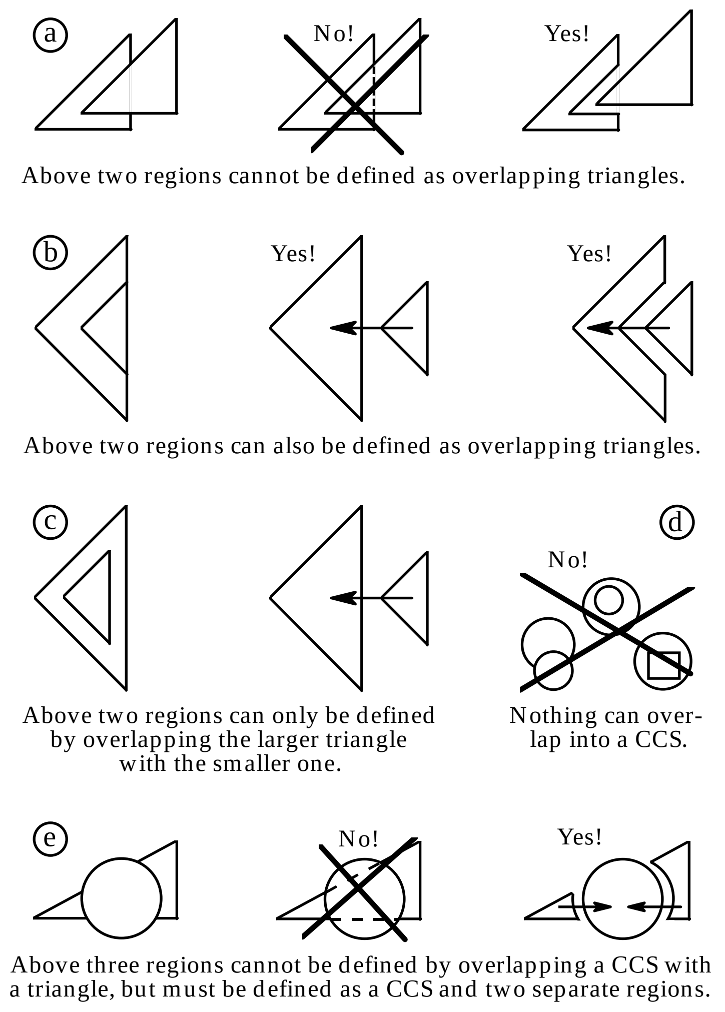
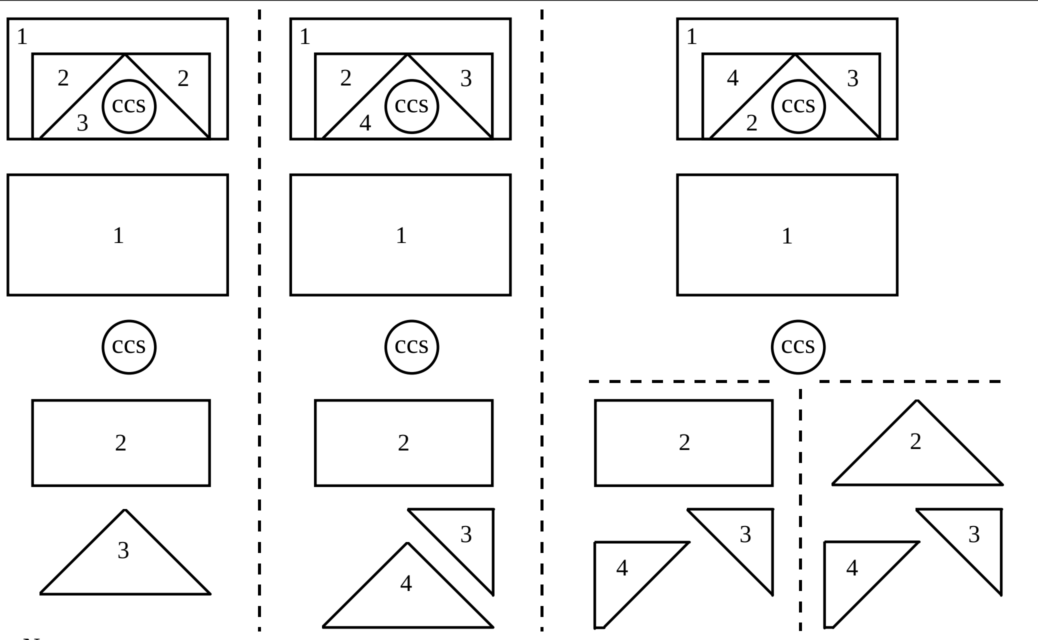

a
No!
Yes!
Above two regions cannot be defined as overlapping triangles.
b
Yes!
Yes!
Above two regions can also be defined as overlapping triangles.
c
d
No!
Above two regions can only be defined by overlapping the larger triangle with the smaller one.
Nothing can overlap into a CCS.
e
No!
Yes!
Above three regions cannot be defined by overlapping a CCS with a triangle, but must be defined as a CCS and two separate regions.

Fig. 3: Examples of the rules for overlapping– CCS Rule 1, and non-CCS Rules 7 and 8.

Remarks

The structures are the basic geometric building blocks that make up the system to be calculated by HELIOS. The CNX operator combines these structures into subsystems, and the subsystems and/or structures into the system itself. Examples of structures are a pin cell, a control blade, a sandwich of water gap, flowbox wall and coolant film, and an arm of the SVEA water cross.

When building up a structure, HELIOS starts with the periphery, which defines the first region. Next, possible CCS-es are inserted and, finally, the other regions. This is illustrated in Fig. 4.



Notes:
- Only an entire CCS can be overlapped by non-CCS regions (twice in variants 1 and 2).
- In the first variant, region 3 splits region 2 in two parts that are treated as one region.
- In variants 3 and 4, the structures are identically the same but built-up differently.

Fig. 4: The build-up of the same geometric structure with different flat-flux regions.

Within the rules for the structures and their coupling, the User has full freedom to build up the system. For example, the whole system might be just one structure, or each of the four mini-bundles of a SVEA assembly and its entire water cross can be structures.

Inside the structures HELIOS uses collision probabilities, while—depending on the coupling between the structures—the (sub)systems are treated by collision probabilities, or are coupled by interface currents. To evaluate the collision probabilities, HELIOS relies on the flat-flux approximation in the regions.

Obviously, closed material areas define regions. In analogy with the CCS regions, these may be subdivided into more regions to obtain a satisfactory spatial mesh for the flat-flux approximation. For example, it is advised to describe the coolant of the pin cells by at least four flat-flux regions, as shown in Fig. 5.

Examples

1) Consider a square pin cell, called *cell*. Let the half-pitch be given by the parameter *$hp* (see PAR operator) and let the outer radius of the CCS, called *pin*, be given by *$rc*. Further, let the coolant around *pin* be divided into four flat-flux regions; see Fig. 5. The STR operator describing this cell is:

```
Cell = STR(("-$hp","-$hp")("-$hp",$hp)($hp,$hp)($hp,"-$hp")   ! 1-4 !
           ("-$hp",   0  )(   0  ,$hp)($hp, 0 )( 0 ,"-$hp")   ! 5-8 !
           ("-$rc",   0  )(   0  ,$rc)($rc, 0 )( 0 ,"-$rc")   ! 9-12!
           /4,cool/pin(0,0)/ 1,5,9,12,8,SW; 3,7,11,10,6,NE;
                                            5,2, 6,10,9,NW)
```

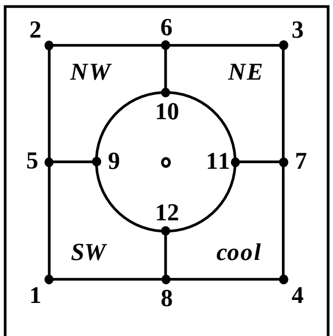


Fig. 5: The structure *cell.*

This example is illustrated in Fig. 5. Notice that $p_n = 4$, while the sequence of the remaining nodes is arbitrary. Further, the SE region is what is left of the background region and still has the name *cool*. If no *bid* and *rid*-names had been specified, the regions *SW, NW, NE* and *cool* would have their default names, i.e. the sequence numbers of their definitions: '*2*', '*4*', '*3*' and '*1*'. Also, a minus sign followed by a parameter is considered an arithmetic expression and must be enclosed in quotes ("...").

2)
```
STR((5.9,2.8)(5.9,0)(0,0)(0,2.8)(0,1.7)(2.3,0)(3.9,0)(1.4,0)(0.3,0)
    (4.8,0) (3.1,1.7) (3.1,2.8) (3.1,0) / 4, bid / pin (3.1,0)1 /
     6,9,12,2,7, a;   6,8,11,10,7, b )
```
or:
```
STR((5.9,2.8)(5.9,0)(0,0)(0,2.8)(0,1.7)(2.3,0)(3.9,0)(1.4,0)(0.3,0)
    (4.8,0) (3.1,1.7) (3.1,2.8) (3.1,0) / 4, bid / pin (3.1,0)1 /
     8,9,12,2,10,11, a;   6,8,11,10,7, b )
```
or:
```
STR((5.9,2.8)(5.9,0)(0,0)(0,2.8)(0,1.7)(2.3,0)(3.9,0)(1.4,0)(0.3,0)
    (4.8,0) (3.1,1.7) (3.1,2.8) (3.1,0) / 4, bid / pin (3.1,0)1 /
     6,8,11,10,7, b;   8,9,12,2,10,11, a )
```

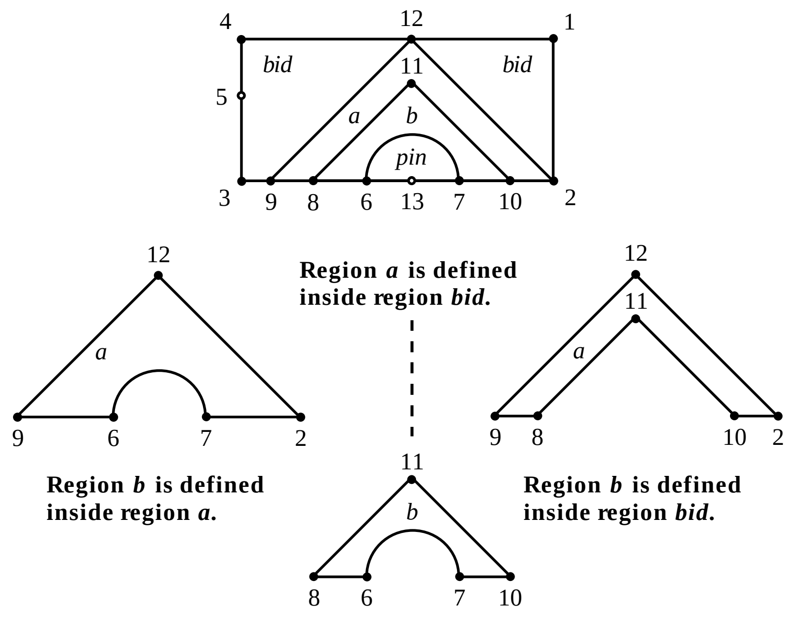

4
12
1
bid
11
bid
5
a
b
pin
3
9
8
6
13
7
10
2
12
Region *a* is defined inside region *bid.*
12
11
a
a
9
6
7
2
9
8
10
2
11
Region *b* is defined inside region *a.*
b
Region *b* is defined inside region *bid.*
8
6
7
10
Fig. 6: Example of a structure with a partial CCS and three non-CCS regions: *bid, a* and *b*.

Fig. 6 illustrates this example. Again, $p_n = 4$, while the order of the other nodes is arbitrary. Because $n_p = 1$ in the STR operator, the azimuthal subdivision of the generic CCS is *na* = 2 or *na* = 1. (The figure shows that *na* = 1.) Notice that according to Rule 8 for non-CCS regions, regions *a* and *b* cannot overlap the partial CCS. That is why the CCS arc must be defined as part of their borders.

Observe also that the background region *bid* has been split into two regions. Being originally defined as one region, HELIOS will treat them as one (flat-flux) region. Further, region *a* can be specified in two ways, depending on whether or not region *b* overlaps *a*. If they overlap, their sequence in the STR operator must be: first *a*, then *b*. If they do not overlap, their sequence is irrelevant, i.e., the last two variants of the example are equivalent. Finally, Rule 6 for nodes and Rule 6 for CCS-es permit the specification of the unused nodes 5 and 13.

TREE(), THE BRANCH-OFF OPERATOR

Purpose

The TREE operator can be used to specify branch-off calculations with various states at a given set of burnups in a PATH operator. Decay may be used to set equilibrium isotopes to their zero-power concentrations, but burnup is not possible. Although most branch-offs can also be done with the PATH operator, the TREE operator is better suited for this purpose.

Usage

name = **TREE(** *bname***/***t* [, *t*]… **/***eb*[/*n*] [, *eb*[/*n*]]… **)**

name	represents a set of branch-off calculations to be done by HELIOS.
bname	is the name of a path in the actual case whose isotopics will be used for the branch-off calculations; or it is the name of a DBMAT operator (see there) with unspecified burnup that retrieves restart materials from a data-base where they have been saved in an earlier case by a DUMP operator (see there).
t	specifies a branch-off state in the set *name*, *t* = [*opt*,] *std*.
eb[/*n*]	specifies one [or *n*] burnups on the path *bname* or among the isotopic dumps accessible via the DBMAT operator *bname* [MWd/t].

Where:

opt is any combination of the following two options, represented by letters, and not separated by delimiters:

- *G* - also gamma calculations;
- *P* - negates, after a state activation, the extra C-flux calculation that updates the equilibrium concentrations (default with decay);

std is one of two specifications that define a state for which neutron (and gamma) fluxes are to be calculated:

- (*st*) - name of a state enclosed in parentheses;
- (*st*,*d*) - state *st* after *d* days of decay, in parentheses.

Rules for burnups

1) Each burnup (*eb*) must be in units of MWd/t (metric ton) relative to the beginning of the path *bname* of the actual case, or the path referred to via the DBMAT operator *bname,* which starts at zero burnup (see PATH operator, Rule 1 for burnup and decay).

2) The burnups *eb* in the set *name* must be given in increasing order. The interpolation (*eb*/*n*) inserts *n*–1 equidistant burnups before *eb*. Together with *eb* they define the branch-off burnups.

3) Branch-off burnups must lie within 0.001 MWd/t from the existing burnups in *bname* (see PATH operator, Rule 2 for burnup).

Rules for states

1) A state (*st*) is activated by specifying its name in parentheses, "(*st*)". This assigns properties and a power level to the geometric system.
2) A state can also be activated after *d* days of decay by specifying, in parentheses, its name followed by a comma and *d*, i.e., "(*st*,*d*)". By varying *d*, the number densities of the equilibrium isotopes during zero-power decay can be followed until equilibrium.
3) States with isotopic multipliers cannot be subjected to decay (see the ISOX and STAT operators), i.e., they cannot be used in a state specification "(*st*,*d*)".
4) The activated state "(*st*)" will have the isotopics of *bname* at *eb*; the activated state "(*st*,*d*)" will have the same isotopics but after *d* days of decay at zero power, after which the state's power is restored.
5) If in *bname*, *eb* has multiple isotopics due to update calculations of equilibrium isotopes or decay (see PATH operator, Rule 4 for initial isotopics), the isotopics prior to the update or decay are taken. If *bname* is a DBMAT operator, only one dump is accessible per burnup, in agreement with this Rule (see DUMP operator, Rules 7 and 8).
6) If a state is activated by "(*st*)", HELIOS will first calculate its neutron (and gamma) fluxes twice, which provides the <u>P and C fluxes</u>. Both are based on initial number densities, but the equilibrium number densities in the latter have been updated with P-flux reaction rates.
7) If a state is activated by "(*st*,*d*)", HELIOS first lets the isotopics of the state decay at zero power during *d* days. Thereafter, the state's power is restored and its neutron (and gamma) fluxes are calculated. No C fluxes can be calculated.
8) All the branch-off states will be calculated at all the branch-off burnups.

Rules for options

1) The options, represented by the two letters *G* and *P*, are local to one state at a time; they activate or negate additional flux calculations.
2) The options may be given in any combination—not separated by delimiters—before a state specification (*std*) in the set. They must be separated from *std* by a comma.
3) <u>Option *G*</u> activates gamma calculations. A state that has also C-flux calculations (see Rule 6 above), will have two gamma calculations, with sources based on the P and C fluxes, respectively.

4) Option *P* negates the extra C-flux calculation of Rule 6 above.

Remarks

The TREE operator can be replaced by the PATH operator, so long as no states with isotopic multipliers are involved. However, the results will not always be rigorously the same when equilibrium isotopes are involved. That is because in a TREE operator the states use the isotopics at *eb*, defined by Rule 5 for states, while in a PATH operator they use the latest available isotopics. Consider the example of path *B* and three branch-off states at three branch-off burnups:

```
B = PATH( /(s), e1/n1, e2/n2, e3/n3)
T = TREE(B/(x), (y), (z) / e1, e2, e3)
```

This can also be achieved with PATH operators alone, as shown in the two variants below. In the first, everything is contained in a single path *Bt*; in the second, four paths are used, the basic burnup path *B*, and the branch-off paths *B*1, *B*2 and *B*3.

```
Bt = PATH(/(s), e1/n1, (x), (y), (z),
           (s), e2/n2, (x), (y), (z),
           (s), e3/n3, (x), (y), (z))
B  = PATH(/(s), e1/n1, e2/n2, e3/n3 )
B1 = PATH(B:e1/(x), (y), (z) )
B2 = PATH(B:e2/(x), (y), (z) )
B3 = PATH(B:e3/(x), (y), (z) )
```

In *Bt*, state *s* has to be reactivated at *e*1 and *e*2 before continuing with burnup. This causes extra P and C-flux calculations (Rule 7 for states of the PATH operator), which makes this variant more costly. The extra C-flux calculations may be suppressed if none of the states *x*, *y* or *z* has materials with equilibrium isotopes in common with *s*. In that case, their calculations do not change the isotopics of *s*. But the P fluxes of the reactivated state *s* will be the same as its C fluxes would have been (had they been calculated) at *e*1 and *e*2 before the reactivation. Both are based on the same predictor-corrector averaged number densities and the same equilibrium densities. The latter are based on the latest available isotopic reaction rates, those of the P fluxes at *e*1 and *e*2 (Rule 7 for states and Rule 8 for burnup in the PATH operator). Thus, unless burnup is done with the C option, the fluxes and isotopics of *s* in the two variants will differ slightly, even if there are no equilibrium isotopes.

In the case that *x*, *y* or *z* have materials with equilibrium isotopes in common with *s*—e.g., if they are branch-offs of *s* with the same fuel—their C-flux calculations will change the equilibrium number densities of *s*. These number densities should then be restored by C-flux calculations when *s* is reactivated at *e*1 and *e*2.

In both variants the P fluxes of *y* and *z* in *Bt*, *B*1, *B*2 and *B*3 will differ from those in *T*—at least if they have equilibrium isotopes in common with *x*. In the PATH operators the P fluxes of *y* will be based on the equilibrium densities of *x*, while in *T* they are based on those of *s*.

Examples

1)
```
VH4    = PATH(VH4:0/(V4),  5000/10, 25000/5 ) ! Burn at 40% void !
VH4CC  = TREE(VH4/P,(VC,8)/5000/5,  25000/5 ) ! Cold-clean       !
VH4V00 = TREE(VH4/  (V0)  /5000/5,  25000/5 ) ! Branch at 0%     !
VH4V40 = TREE(VH4/  (V4)  /5000/5,  25000/5 ) ! Superfluous?     !
VH4V70 = TREE(VH4/  (V7)  /5000/5,  25000/5 ) ! Branch at 70%    !
```

The four void branches are equivalent with

```
VBRA4  = TREE(VH4/P,(VC,8), (V0), (V4), (V7) / 5000/5, 25000/5 )
              ! Cold-clean, 0%,  40%,  70% void !
```

The path *VH*4 (void history, 40%) specifies depletion of a BWR assembly at 40% void (state *V*4). The burnup to 25,000 MWd/t is in ten steps of 500 MWd/t and five steps of 4,000 MWd/t. The branch-off calculations are specified for four states at ten branch-off burnups. The branch-off states are: cold-clean, 0%, 40% and 70% void; the first five burnups lie 1,000 MWd/t apart, the other five 4,000 MWd/t.

The cold-clean state (room temperature, no Xe-135) is represented by the cold state *VC* after eight days of decay. The voided states are represented by *V*0, *V*4 and *V*7, respectively. Option *P* before the cold-clean state negates the unnecessary C-flux calculation. Its specification is not needed; it is default with decay. After eight days, the equilibrium isotopes should have reached their zero-power equilibrium concentrations, e.g., Xe-135 should be negligible.

The P fluxes during burnup and the P fluxes of the branch-off states are based on predictor and predictor-corrector averaged number densities, respectively. Therefore, while the branch-off at 40% void recalculates some of the points on *VH*4, the two calculations will give slightly different results. Although it is consistent to calculate in this way, it is superfluous if the often small differences are acceptable.

2)
```
C1 = CASE(...)
...
Pa = PATH(/(st1),10000/10,(st2),20000/10)
Du = DUMP(df/area,Pa)
C1 = RUN()

C2 = CASE(...)
...
Db = DBMAT(df;C1;Du/(st2))
Tr = TREE(Db/(V0),(V4),(V7)/10000,20000/2)
C2 = RUN()
```

First, case *C*1 is run. Dumps are made in the data-base *df* at 22 burnups, 11 for each state, including the first of *st*1 and *st*2 at 0 and 10,000 MWd/t, respectively. Next, case *C*2 is run whose branch-off tree *Tr* uses the isotopics of state *st*2 on path *Pa* of case *C*1 at 10,000, 15,000 and 20,000 MWd/t. Observe that only burnups from 10,000 MWd/t and above can be used in *Tr*. That is because *Db* retrieves the isotopics of *st*2 along the path *Pa* of case *C*1, and this state was only active at 10,000 MWd/t and above.

3) <u>Common beginner's mistake in connection with the TREE operator</u>

To understand this example one should know that HELIOS uses two isotopic arrays in its calculations. One array, ISOTOP(), contains the library sequence numbers of all the isotopes in the case to be run. The other, ISODNS(), which is parallel to ISOTOP(), contains at any time in the calculations their actual number densities.

Assume the case of a PWR fuel assembly with three coolants with three different boron contents. Call the fuel materials *Fuel*, the coolant materials *ppm*1, *ppm*2 and *ppm3*, and the remaining materials *X*. Assume one wishes to deplete with *ppm*2 until 10 000 MWd/t in equal steps of 1 000 MWd/t, followed by branch-off calculations with *ppm*1 and *ppm*3 at 0, 5 000 and 10 000 MWd/t. The three states then must contain the three material overlay sets:

```
osm1 = OVSM(ovFuel,ovX,ovppm1)   ! used in st1 = STAT(osm1, ...) !
osm2 = OVSM(osm1/ovppm2)         ! used in st2 = STAT(osm2, ...) !
osm3 = OVSM(osm1/ovppm3)         ! used in st3 = STAT(osm3, ...) !
```

where *ovFuel, ovX, ovppm*1, ... are the materials' overlays (OVLMs). The burnup path and the two branch-offs could be defined like:

```
pa2  = PATH(/(st2),10000/10)       ! depletion with coolant ppm2 !
tr1  = TREE(pa2/(st1)/0,10000/2)   ! branchoff with coolant ppm1 !
tr3  = TREE(pa2/(st3)/0,10000/2)   ! branchoff with coolant ppm3 !
```

The arrays ISOTOP() and ISODNS() will then look like:

	all states		*st*1	*st*2	*st*3
ISOTOP()	*Fuel*	*X*	*ppm1*	*ppm2*	*ppm3*
ISODNS()	*Fuel*	*X*	*ppm1*	*ppm2*	*ppm3*

where the blocks *Fuel*, *X*, *ppm*1, ... contain library sequence numbers and number densities of the isotopes of the fuels, the remaining non-coolant materials, and the three coolants. During burnup, the blocks *Fuel*, *X* and *ppm*2 are active, while the content in ISODNS() changes insofar as the burnable isotopes of *Fuel* are concerned.

HELIOS also has an isotopics depot in the scratch file, where space has been reserved to stack as many copies of ISODNS() as required for the branch-off calculations, plus one extra copy with ISODNS() at the start of the calculations. As burnup proceeds, HELIOS will dump successive copies of ISODNS() , in this example at 5 000 and 10 000 MWd/t. At the end of the burnup path the isotopics depot then looks like:

all states		*st*1	*st*2	*st*3
Fuel(0)	*X*	*ppm*1	*ppm*2	*ppm*3
Fuel(5,000)	*X*	*ppm*1	*ppm*2	*ppm*3
Fuel(10,000)	*X*	*ppm*1	*ppm*2	*ppm*3

This is the correct way to set up the calculations. In the TREE calculations the blocks *Fuel*, *X*, and either *ppm*1 or *ppm*3 will be active at the three burnups 0, 5 000, and 10 000 MWd/t. That means that all the time the same *Fuel* is used, but at three different burnups. This is the correct way of setting up the data.

The mistake often made by the unsuspecting beginner is to overlook Rule 2 for materials of the OVLM operator. According to this rule, if an overlay is referenced by name in more than one OVSM operator, then the materials of that overlay are defined once again; one could say that they are "cloned". The error consists in defining the three OVSMs as follows:

```
osm1 = OVSM(ovFuel,ovX,ovppm1)   ! used in st1 = STAT(osm1, ...) !
osm2 = OVSM(ovFuel,ovX,ovppm2)   ! used in st2 = STAT(osm2, ...) !
osm3 = OVSM(ovFuel,ovX,ovppm3)   ! used in st3 = STAT(osm3, ...) !
```

The arrays ISOTOP() and ISODNS() will then look like:

	<------ *st*1	----->		<------ *st*2	----->	<------ *st*3	----->	
ISOTOP()	*Fuel*	*X*	*ppm1*	*Fuel*	*X ppm2*	*Fuel*	*X*	*ppm3*
ISODNS()	*Fuel*	*X*	*ppm1*	*Fuel*	*X ppm2*	*Fuel*	*X*	*ppm3*

There are now three sets of fuel materials, the second of which will be depleted because during burnup *st*2 is active. The isotopics depot at the end of the burnup path will then look like:

<------ *st*1	----->		<--------- *st*2	--------->		<------ *st*3	----->	
Fuel(0)	*X*	*ppm*1	*Fuel*(0)	*X*	*ppm*2	*Fuel*(0)	*X*	*ppm*3
Fuel(0)	*X*	*ppm*1	*Fuel*(5,000)	*X*	*ppm*2	*Fuel*(0)	*X*	*ppm*3
Fuel(0)	*X*	*ppm*1	*Fuel*(10,000)	*X*	*ppm*2	*Fuel*(0)	*X*	*ppm*3

This is obviously not what was intended. The TREE calculations will use the first and third sets, whose fuel materials are still fresh, i.e., they have not undergone any burnup. The unpleasant surprise is that for each of the two TREEs, the branch-offs at 5 000 and 10 000 MWd/t produce the same results!

Studsvik Scandpower

HELIOS Methods
(version 1.10)

01 April 2008

DOCUMENT CONTROL

This document will be updated periodically, whenever revisions are made to the Manual. This front page, together with a page specifying the pages to be deleted and/or added, will be provided with each new addition. The User should check that the Manual corresponds to the Program Version in use and that all revisions are included.

PROGRAM MANUAL		PROGRAM		REMARKS	Rev'd by	Appr'd by
Rev. No	Date	Version	Date			
0	18Oct93	HE931A helios-1.0	11Oct93	14Sep06		
1	05Dec94	HE941A helios-1.1	05Dec94	Minor changes.		
1	-	helios-1.2	-	Unchanged.		
2	15Dec95	helios-1.3	15Dec95	Represent gadolinia by 64000; general restart option added.		
2	-	helios-1.4	-	Unchanged.		
3	11Sep98	helios-1.5	11Sep98	Manual converted from Macintosh; library description added.		
4	01Apr00	helios-1.6	01Apr00	Minor changes; new library structures.		
5	31Oct01	helios-1.7	31Oct01	New restart; improved mutual resonance shielding; energy-dependent energy per fission; resonance-scattering data in library.		
6	20Nov03	helios-1.8	20Nov03	Improved resonance treatment; more heavy-metal isotopes in burnup; library: new resonance scatter data, 47 instead of 45 groups, and more isotopes		
7	01Dec05	helios-1.9	01Dec05	No longer leakage from balance; outflow and inflow transport XS; typos and minor errors		
8	01Apr08	helios-1.10	01Apr08	Represent dysprosia and erbia by 66000 and 68000; improved formula for $\kappa(E)$; improved resonance treatment with temperature profile; add anonymous nodes on outer ring of segmented CCS; rotation angle for CCS in structure; added new angular coupling options; P_2 and P_3 output; corrected various typos, minor errors, and clarified several explanations.		

REV. 1: 05DEC94

Completion of Chapter V; addition of chapters VI-VIII.
Minor changes in the other chapters.

Pages replaced:

Entire document

REV. 2: 15DEC95

Represent gadolinia by 64000.
Introduce dump/restart option.

Pages replaced:

Cover page
Document control .... pages 1–2
Contents .... pages 1–4
Chapter I .... pages 1–5
Chapter II .... pages 1–26
Chapter VIII .... pages 1–19

REV. 3: 11SEP98

Chapter IX on the library added.
Converted from Macintosh, with many minor changes.

Pages replaced:

Entire document

REV. 4: 01APR00

Section in Chapter II on (n,2n) and (n,3n) added.
Xe-128 and Xe-130 added in Fig. VII.2/2.
Appendices of Chaper IX updated for new libraries.
Typing errors corrected.

Pages replaced:

Cover page
Document control .... pages 1–2
Contents .... pages 1–4
Chapter II .... pages 1–26
Chapter VII .... pages 1–19
Chapter VIII .... pages 1–19
Chapter IX .... pages 1–26

REV. 5: 31OCT01

Chapter II: section 5.1.3 on restart materials added.
Chapter II: sections 5.4-6 changed to 5.5-7; new section 5.4 on energy release per fission.
Chapter III: section 5.2 improved formula for resonance cross sections and mutual shielding.
Chapter IV: section 2.1 has changed a reference from II.5.4 to II.5.5.
Chapter VII Figure 3 improved (minor).
Chapter VIII: sections 4 and 5 modified because of the new restart option.
Chapter IX: sections 2, 3 and 8 modified: energy-dependent fission energy and resonance scattering.

Pages replaced:

Cover page
Document control .... pages 1–3
Contents .... pages 1–5
Chapter II .... pages 1–28
Chapter III .... pages 1–26
Chapter IV .... pages 1–21
Chapter VII .... pages 1–19
Chapter VIII .... pages 1–19
Chapter IX .... pages 1–26

REV. 6: ***20NOV03***

Chapter III: improved calculation of subgroup constants; resonance scattering.
Chapter VII: seven heavy-metal isotopes added to the burnup chains.
Chapter VIII: sentence added in section 5; typo removed on page 11.
Chapter IX: More isotopes; 47- instead of 45-group group library.

Pages replaced:

Cover page
Document control .... pages 1–3
Contents .... pages 1–5
Chapter III .... pages 1–28
Chapter VII .... pages 1–19
Chapter VIII .... pages 1–19
Chapter IX .... pages 1–26

REV. 7: ***01DEC05***

Contents: chapter II page numbering.
Chapter I: remark on removal of leakage calculation from balance.
Chapter II: changed Avogadro number and introduction of inflow transport correction.
Chapter III: errors on page 1 (total energy range).
ChapterVI: typo after Eq. (14), change $\mu/2$ to $\pi/2$.
Chapter VIII: updated remarks on removal of leakage calculation from balance; typo on p.3.
Chapter IX: inflow transport correction; typos in Eq. (9), p. 24 (Sm150 has n,2n), and p. 26 (Zircaloy-2).

Pages replaced:

Cover page
Document control .... pages 1–3
Summary .... pages 1–5
Chapter II .... pages 1–28
Chapter III .... pages 1–28
Chapter VI .... pages 1–13
Chapter VIII .... pages 1–19
Chapter IX .... pages 1–26

REV. 8: ***30JUL2007***

Chapter II: Dy_2O_3 = 66000 and Er_2O_3 = 68000; add P2 and P3 to Table 1; correct Eq.(45) for $\kappa(E)$.
Chapter III: resonance treatment of a temperature profile.
Chapter VIII: added output types *p2* and *p3*; updated text of section 3.1.
Chapter IX: updated text in section 9; appendix B shows isotopes with P_n ($n = 1, 2, 3$) data.

Pages replaced:

Cover page
Document control .... pages 1–4
Chapter II .... pages 1–28
Chapter III .... pages 1–28
Chapter VIII .... pages 1–19
Chapter IX .... pages 1–26

REV. 9: 23JAN08

Chapter I: corrections for characteristics.
Chapter II: added anonymous points generation for azimuthal subdivision of outer ring in CCS; added rotation angle for azimuthal segmentation; converted equations from MathType for compatibility.
Chapter III: updated description of subgroup constants calculation, converted equations from MathType for compatibility.
Chapter IV: added new angular coupling options; added method of characteristics description as Section IV.6; converted equations from MathType for compatibility.
Chapter V: added description of MOC changes to ray-tracing routines; converted equations from MathType for compatibility.
Chapter VI: converted equations from MathType for compatibility.
Chapter VII: various minor corrections; converted equations from MathType for compatibility.
Chapter VIII: various minor corrections; converted equations from MathType for compatibility.
Chapter IX: converted equations from MathType for compatibility.

This will be HELIOS-2.0.

<u>Pages replaced:</u>

Cover page
Document Control.........pages 1-4
Contents ....pages 1-5
Chapter I.......................pages 1-5
Chapter II......................pages 1-29
Chapter III.....................pages 1-27
Chapter IV.....................pages 1-21
Chapter V......................pages 1-23
Chapter VI.....................pages 1-13
Chapter VII....................pages 1-20
Chapter VIII...................pages 1-19
Chapter IX.....................pages 1-26

CONTENTS

I. HELIOS: SUMMARY

I.1 INTRODUCTION

This report describes the methods of the neutron and gamma transport code HELIOS for lattice burnup, in general two-dimensional geometry. Its input and output processors are the separate codes AURORA and ZENITH. The data flow between these codes is via a data-base that is accessed and maintained by the subroutine package HERMES. Fig. 1 shows the interrelationship of these codes.

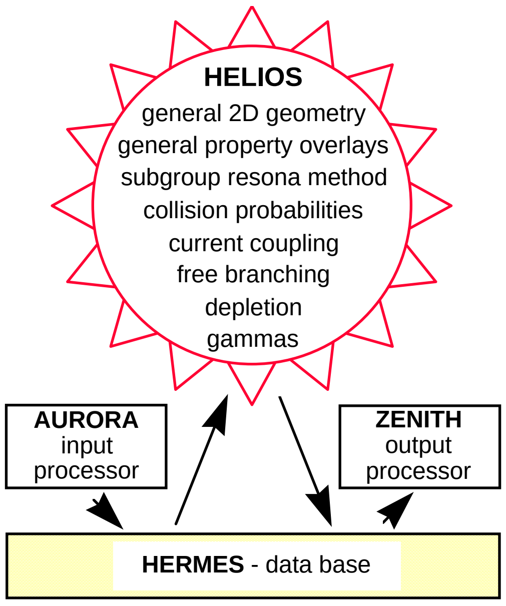


Fig. 1: Data flow between HELIOS and its input/output processors AURORA/ZENITH.

AURORA reads, processes, and then saves the User's input. The result is a number of "User arrays" that start with the letter "U". These arrays, information on their sizes, and the User's input, are written into a HERMES data-base for retrieval by HELIOS and/or ZENITH.

For each case HELIOS retrieves the input from this data-base and executes the calculations specified therein. The type and amount of the output depend on what has been specified by the User in the input. This output is a number of "ZENITH arrays" that start with the letter "Z", which are written into the same data-base for retrieval and further processing by ZENITH.

If isotopic dumps for later restart calculations are requested, they are also saved as "Z" arrays. These arrays, however, are not for ZENITH, nor do they have to be saved in the data-base used for input and output. The restart dumps can go to other data-bases. In fact, each dump may have its own data-base in which the number densities are saved.

The input, except for the basic nuclear data in the library which are not User-specified, is detailed in the AURORA manual. It consists of the following data types:

- the nuclear-data library with the basic nuclear data, which also defines the energy discretizations (group structures) of the particle* transport calculations;
- data that define the (initial) number densities of materials, and the elements of the albedo matrix elements;
- data that define the geometry of the system, including the spatial and angular discretizations to be used in the transport calculations;
- data that assign one or more property sets to the geometric system, thus defining one or more "states" of that system;
- data that define the execution sequence of the calculations;
- data that define what output will be saved in the data-base;
- optional data that define an experimental buckling, change default iteration parameters, accuracies and methods, and produce outputs.

At each calculational point (also called reactivity point), the essential results are particle fluxes and currents, and in case of a burnup or a time step, the new material number densities. Together with the nuclear-library data and the User's input, this is all that is needed to obtain an output. It is, however, not feasible to save at each calculational point, in all energy groups and all regions, fluxes, number densities and isotopic resonance-shielded cross-sections or, in all energy groups and at all interfaces, the currents in all angular sectors. Therefore, the User has to define in the input what output must be saved in the data-base. For example, two-group homogenized data of a fuel assembly, one-group fluxes and macroscopic energy-production cross-sections (XSs), $\kappa\Sigma f$ for all the fuel pins (to obtain power maps), et cetera.

I.2 THE METHODS USED BY HELIOS

The methods described in this document cover three areas:

- the geometric buildup of the system and its properties (chapter II);
- the physical methods used to obtain fluxes, currents, and number densities, which in turn consist of five parts:
    - calculating resonance-shielded microscopic XSs (chapter III)
    - calculating fluxes and currents by the current-coupling collision-probability (CCCP) method for particle transport (chapter IV)
    - evaluating first-flight probabilities (chapter V)
    - evaluating, with the B_1 method, the criticality spectrum which is used to rebalance the spectrum of the CCCP solution (chapter VI)
    - solving the burnup chains to get new number densities (chapter VII)

* Throughout this document, the term *particles* refers to neutrons or gammas.

- the output processing to obtain the "ZENITH arrays" with number densities and other information for restart calculations (chapter VIII).

I.3 PROGRAM DESCRIPTION PER MODULE

HELIOS consists of a main module (LINK00), which calls nineteen computational modules, LINK01-11 and LINK13–20. The first eleven take care of the input treatment, they are called once per case. In fact, the first one is called only once per run of one or more cases, so long as the same library is used. Six of the remaining modules do the physics calculations, while the last two do the processing for the regular output and for the restart dumps. Most of these eight modules are called at least once per reactivity point.[*] The depletion module is only called if there is burnup or decay, while the module for restart dumps is only called when such dumps are requested.

The following is a very brief description of the tasks/contents of each of the modules of HELIOS (LINK12 does not exist):

LINK00:	It starts the calculations, initiates the files and calls the modules. It contains some general routines, e.g., for random access of disk files, and for obtaining the date and the elapsed CPU time.
LINK01:	It reads and rearranges some basic cross section library data.
LINK02:	It locates the next case in the HERMES data-base, sets dimensioning data and options that come from AURORA, and estimates dimensioning data for the geometry treatment. It also contains the subroutines that interact with the HERMES data-base.
LINK03:	It sets up the calculational path.
LINK04:	It interprets the geometric input structures, as described in chapter II.
LINK05:	It builds up the geometric system from the structures, as described in chapter II.
LINK06:	If specified in the input, structures can be fusioned into space elements, which is done in this module. Otherwise, each structure is treated as a separate space element in the CCCP calculations. After possible fusioning, the boundary conditions are assigned to each boundary segment, as described in chapter II.
LINK07:	It interprets the overlay input to define one or more overlay sets for materials, temperatures and density-correction factors. Each set must assign data to the entire geometric system.
LINK08:	It interprets material and albedo input. Burnable materials that occur in more than one region become more than one material. These materials are completed to contain all the isotopes of their burnup chains.

[*] A reactivity point is the same as a calculational point. It represents a neutron (and gamma) transport calculation, and possibly the preceding burnup or decay step, with output.

Materials with resonance isotopes that are assigned to different regions, and materials with scattering isotopes lighter than AWMAX that are assigned to regions whose temperatures differ by more than DTMIN, temporarily become "multiple" materials. (Presently, AWMAX = 50 amu and DTMIN = 20 K.)

LINK09: It sets up administrative arrays for the output treatment, and creates in the HERMES data-base those catalog chains and output data lists that do not depend on the reactivity points, as described in chapter VIII.

LINK10: It analyses the space-element types for symmetries to be used in the first-flight-probability calculations, as described in chapter V.

LINK11: It draws in each space-element type the chords for the evaluation of the first-flight probabilities, as described in chapter V. The chord intersections with the region boundaries are stored in the scratch working file.*

LINK13: In the first stage, it prepares XSs for the transport calculations from which the equivalence XSs are determined. In the second stage, it evaluates the effective resonance XSs and the modified out-scattering XSs due to the flux dip. This is described in chapter III.

LINK14: It sets up the macroscopic neutron or gamma XSs of the regions. For the gamma calculations, it also determines the sources in the regions from neutron fluxes and the microscopic (n,γ) matrices in the library. This is described in chapters II.5 and IV.2.

LINK15: It evaluates, per space element and per group, first-flight probabilities for the calculations of the resonance, neutron, or gamma fluxes, as described in chapter V. It converts these probabilities into response fluxes, and multiple-flight transmission and escape probabilities, which are used in the CCCP iteration process. This is described in chapter IV.

LINK16: It solves the fluxes and partial currents of the particles according to the CCCP method described in chapter IV. There was also a subroutine that used the particle balance to obtain the net leakage in all groups and regions. Since version 1.7 it did not calculate the leakage any more but replaced it by the flux. It has been removed altogether since version 1.9.

LINK17: It uses the B_1 method to evaluate the criticality (or leakage) spectrum of the neutron flux. It enforces this spectrum on the neutron flux obtained from the transport calculation. This is described in chapter VI.

LINK18: It does the burnup calculations as described in chapter VII, using a predictor-corrector method. The basic results are the burnups of the individual materials, their new number densities and the time-averages of these number densities during the actual burnup step.

* Depending on the problem size, the scratch file, or at least part of it, is contained in the memory.

LINK19: It saves number densities and other information for restart calculations, as described in chapter VIII. These data are dumped (saved) as data lists in a HERMES data-base. Each dump may have its own data-base, which may be the same as, or different from, the I/O data-base.

LINK20: It processes the data to obtain the User-specified output, as described in chapter VIII. These results are saved as data lists in the I/O HERMES data-base.

II. HELIOS: GEOMETRY AND PROPERTIES

II. 1 INTRODUCTION AND DEFINITIONS

HELIOS can calculate almost any two-dimensional (2D) system. This chapter deals with the buildup of such a system from basic structures, the properties that must be assigned to its various regions, and the limitations that apply. Section 2 deals with the basic structures and how they can be subdivided into regions (LINK04 of the program). Section 3 describes how these structures are combined into the final geometric system (LINK05-06 of the program). Section 4 discusses the boundary conditions that can be assigned to the system (LINK06 of the program). Section 5, finally, describes the properties that must be assigned to the regions, and defines the cross-section (XS) types used in the calculations (LINK07-08 and LINK14 of the program).

First, some definitions are given—details follow in the text:

- A node is a point in the 2D space, characterized by its (*x*,*y*) coordinates.
- A segment is a straight line or a circular arc between two nodes. It has the dimension of square centimetres[1].
- A region is a volume[1] enclosed by one or more segments.
- A CCS is a circular cylindrical system which may be subdivided into regions.
- A structure is a polygon which may be subdivided into regions. It is an input concept that facilitates the buildup of the system to be calculated.
- A space element is a computational concept: inside, first-flight probabilities are used, while it is coupled by interface currents with the outside. It is either a single structure, or more structures fused into a superstructure by specifying a coupling order of zero (see below) at the interfaces.
- A boundary or reflector element is a computational concept. To each boundary segment of the system a reflector element is assigned. This element, whose volume is zero, defines the properties of the outside.
- A sector is a surface sector of the directional half-sphere.
- A coupling order is a number (*k*) that defines a certain discretization of the partial currents that couple a space element with its surroundings.

The regions represent the spatial discretization of the fluxes and the sources in the system. Within a region, material properties are constant, fluxes are assumed to be flat, and particle sources are assumed to be flat and isotropic. The segments represent the spatial discretization of the currents in the system. Along a segment, currents are assumed to be flat. Currents are only calculated at the interface segments between space elements and between space and reflector elements. The sectors represent the angular discretization of the currents crossing an interface segment.

[1] Throughout this document, surfaces and volumes are per 1 cm height.

II. 2 THE STRUCTURES

The structures are the basic building blocks of the geometric system to be calculated. They facilitate the buildup of the system for the User. At the same time, they are the smallest space elements used in the calculations. This is no limitation, because two or more structures can be fusioned into a "superstructure", which computationally will be treated as a space element. This permits large space elements, which may be complicated to describe by a single structure, to be built up from smaller structures. In HELIOS, the structures are treated in LINK04.

II. 2.1 The periphery

The first step in the definition of a structure is the definition of its periphery. This periphery must be made up of straight line segments, i.e., it must describe a polygon. The shape of the polygon is arbitrary; it may be irregular and reentrant, as shown in Fig. 1. The polygon must be specified by "supernodes", i.e., nodes at which the periphery changes direction. Thus, a triangle is described by three nodes and a cross by twelve, as can be seen in Fig. 1.

The choice of the (x,y) coordinate system in which to describe a structure is free; each structure has its own independent coordinate system. The nodes must be specified such that, when going from node *1* to node *2* to node *3* ... and eventually back to node *1*, the structure is on the right, as in Fig. 1. With the nodes thus specified, it is easy to find the volume and the centre of gravity of the polygon, quantities that are needed later on. This will be demonstrated in section 2.4.

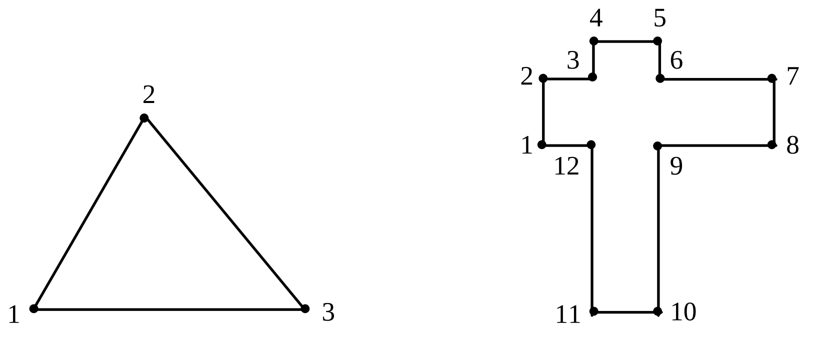


Fig. 1: Peripheries and supernodes of two structures; the cross is reentrant.

II. 2.2 CCS-es and their regions

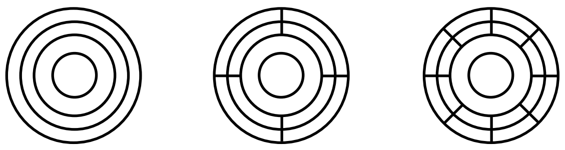
The next step is to specify none, one or more CCS-es inside the structure. The CCS-es inside a structure can either be entire or partial, as illustrated in Fig. 2. An entire CCS fits inside the structure, nowhere intersecting with the periphery, although it may be tangential to the periphery. A partial CCS either has its centre on a peripheral segment, or in a supernode of the periphery and must only intersect the peripheral segment(s) on which its centre is located. Whether a CCS will be entire, partial, or both, it is always generically defined as an entire CCS.

Internally, a CCS can be subdivided into regions; see section 2.3 for the definition of a region. First, a subdivision into one or more concentric annuli defines one or more annular regions. Second, one or more adjacent annular regions can be further subdivided into a number of equal azimuthal sections. In a partial CCS, rotation of the azimuthal segmentation allows partial section to be defined. Figure 2 shows various CCS region definitions.

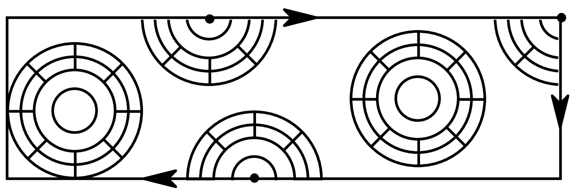
In each annulus, the azimuthal regions are ordered in clockwise direction, starting from the radius pointing to the north. Since HELIOS-1.10, for an entire CCS, the CCS itself can be oriented such that the north corresponds to a desired direction via use of a rotation angle. In a partial CCS, the north is the centre-to-intersection direction that has the structure on the right (see Fig. 2) and the rotation angle applies to the azimuthal subdivision; the north direction is not changed.

Since HELIOS-1.10, in the instances depicted in the upper right picture in Fig. 2, **anonymous nodes** (i.e., nodes not accessible by the user) are defined for the points on the outer boundary of the CCS where the azimuthal segment lines intersect the outer radius of the CCS. Also, segments incorporating these anonymous nodes are defined. This permits proper calculation of the surface areas between regions.

These are the only regions possible; thus, the choice of CCS regions is not free. An annular cluster around fuel pins must be approximated by a regular polygon, while the rounded box corners in a BWR assembly must be approximated by straight-line segments.



Three generic CCS-es with 4, 10 and 18 regions



Two entire and three partial CCS-es (the arrows show the north directions of the partial CCS-es)

Fig. 2: CCS examples.

II. 2.3 Non-CCS regions and nodes

The regions are the spatial discretization of the transport calculations. Their choice, together with that of the other discretizations—energy by groups, interfaces by segments, and directions by angular sectors—determines the quality of the transport calculations. There are two types of regions: CCS regions and non-CCS regions. The former, which lie inside the CCS-es, have already been described in section 2.2.

The non-CCS regions, which lie outside the CCS-es, must be defined in the same way as the structures, by specifying the supernodes of their periphery. However, this periphery can consist of straight-line *and* arc segments. The arc segments must correspond to parts of the periphery of the CCS-es present in the structure. Again, the region to be specified must be on the right of the segments when going from node n to node $(n+1)$. The nodes that define a non-CCS region must lie inside the structure or on its periphery, but not inside a CCS, except that they may lie on its circular periphery.

Nodes that are not used to define non-CCS regions may be specified, too; for example, to subdivide a peripheral face of the structure into two or more segments to improve the flat-current approximation, or the extra node is used later to couple the actual structure with another structure; see section 3. Some nodes can lie even inside a partial CCS, provided they lie on one of the two radii that coincide with the periphery of the structure; then they must lie at the CCS centre or at the intersections of the annuli inside the CCS with the periphery of the structure. Actually, if not specified, HELIOS creates these CCS nodes as anonymous nodes, i.e., they cannot be referred to by the User. Finally, nodes that are never used also may be specified.

Initially, when defining the periphery of the structure, the entire structure is considered as one region, the background region. Inserting one or more (partial) CCS-es reduces the volume of the background region by the CCS volumes. The volumes of the successively defined non-CCS regions are also subtracted from the volume of the background region. The background region that finally remains must have a finite volume; it does not have to be a coherent area (all other regions are coherent areas).

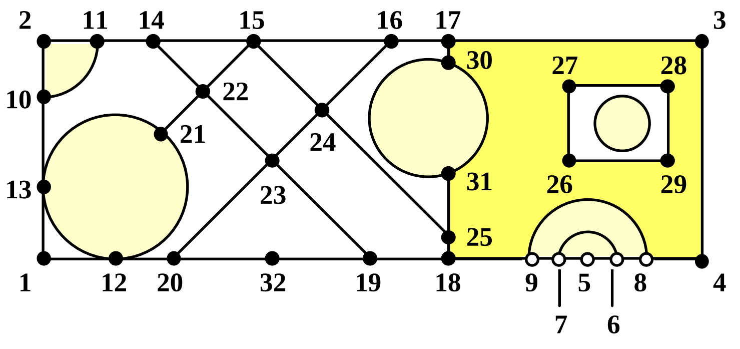


Fig. 3: Example of nodes and regions.

Figure 3 shows various aspects of non-CCS regions and nodes. First, the periphery of the structure must be defined by the nodes *1-2-3-4-1*; at that moment, the original background region (the first non-CCS region) corresponds to the entire structure. Next, the CCS-es must be inserted by specifying their centre coordinates. Finally, the peripheries of the remaining non-CCS regions must be specified. In Fig. 3, the non-CCS regions could have been defined as follows:

1-13-12-1,	*13-10-11-14-22-21-13,*	*12-21-22-23-20-12,*
22-14-15-22,	*22-15-24-23-22,*	*20-23-19-20,*
19-23-24-25-18-19,	*25-24-16-17-30-31-25,*	*26-27-28-29-26.*

Observe: (1) Nodes *5*, *6*, *7*, *8* and *9* are not used but may be specified; otherwise they are anonymous. (2) Node *32* is not used but may be specified; it is not a supernode, so it cannot be used in a region definition. (3) The shaded region is the final background region; its periphery, *9-31-30-17-3-4-8-9*, must not be specified. (4) The small CCS inside the last region remains intact because CCS-es cannot be overlapped by non-CCS regions, nor by other CCS-es.

II. 2.4 Volumes, surfaces and points of gravity

The region volumes (areas x 1 cm height) and the surfaces (lengths x 1 cm height) of the coupling segments are needed for the transport calculations. Moreover, the output needs the gravity points of the regions and of the coupling segments, too.

First, general formulas are given for the sum (difference) of two volumes or surfaces, and for the corresponding gravity point. Let D denote a volume or surface, and z_c the x or y-coordinate of a gravity point. Then one has:

$$D = D_1 \pm D_2 \quad \text{with } D = V \text{ or } S\,; \tag{1}$$

$$z_c = \frac{z_{1c}D_1 \pm z_{2c}D_2}{D_1 \pm D_2} \quad \text{with } z = x \text{ or } y\,. \tag{2}$$

Because a coupling segment always is a straight line between two nodes, *1* and *2*, its surface (i.e., length) and gravity point are given by:

$$S = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\;; \tag{3}$$

$$x_c = \tfrac{1}{2}(x_1 + x_2) \quad \text{and} \quad y_c = \tfrac{1}{2}(y_1 + y_2)\,. \tag{4}$$

First, the volumes are evaluated as if all segments between the supernodes that define a region were straight lines. Next, the volumes of partially or entirely enclosed CCS-es and of enclosed non-CCS regions are subtracted, using Eq. (1). The same is done for the gravity points, using Eq. (2).

II. 2.4.1 Polygons

A polygon can be constructed by successively adding or subtracting triangles. The *n*'th triangle is made up of the nodes *1*, (*n*+1), and (*n*+2), as illustrated in Fig. 4. There, the polygon, which is defined by *1-2-3-4-5-1*, can be made up of the triangles: *1-2-3-1*, *1-3-4-1* and *1-3-4-1*. The latter must be subtracted from the other two; this subtracts the light and dark-shaded areas, by which the first and second triangle overestimate the size of the polygon. Using vector products, the volume is readily found from:

$$\vec{v}_1 \times \vec{v}_2 + \vec{x}_1 \times \vec{v}_3 + \vec{x}_2 \times \vec{v}_4 = -2V\,, \tag{5}$$

where

$$\vec{x}_1 = \vec{v}_1 + \vec{v}_2 \quad \text{and} \quad \vec{x}_2 = \vec{x}_1 + \vec{v}_3 (\vec{x}_3 = \vec{x}_2 + \vec{v}_4 = -\vec{v}_5)\,. \tag{6}$$

Observe that the result is negative if the region is always to the right of the defining segments.

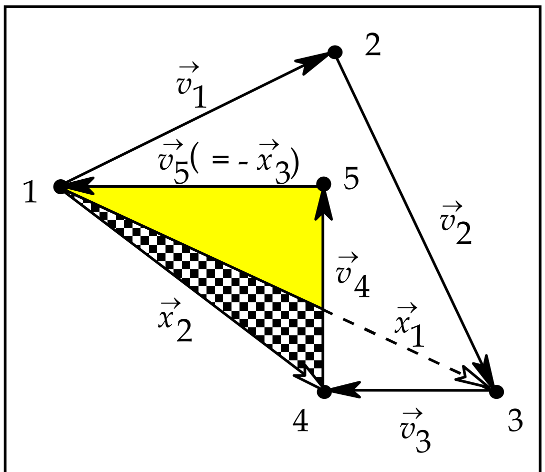


Fig. 4: Volume of a polygon.

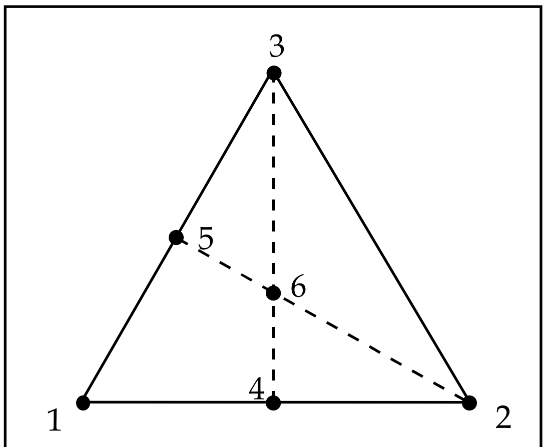


Fig. 5: Gravity point of a triangle.

The point of gravity is found by adding the gravity points of the triangles that make up the polygon according to Eq. (2). The individual vector products in Eq. (5) automatically subtract or add the contributions of the three triangles.

The gravity point of a triangle lies at the intersection of any two lines that go from a corner to the midpoint of its opposite side. In Fig. 5, these are the lines *3-4* and *2-5*; their intersection is at point *6*. Thus, to obtain this gravity point, three ingredients are needed.

- The midpoint, (x_m,y_m), of a line segment between the points *1* and *2*:

$$(x_m, y_m) = \left(\tfrac{1}{2}(x_1 + x_2), \tfrac{1}{2}(y_1 + y_2)\right). \tag{7}$$

- The equation of a line through two points *1* and *2*:

$$px + qy = r\,, \tag{8}$$

With

$$\left.\begin{array}{l} p = (y_1 - y_2)/d, \quad q = (x_2 - x_1)/d, \quad r = (x_2 y_1 - x_1 y_2)/d, \\ \text{where} \quad d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{array}\right\}. \tag{9}$$

- The intersection, (x_c, y_c), of two lines, $p_1 x + q_1 y = r_1$ and $p_2 x + q_2 y = r_2$:

$$(x_c, y_c) = \big((q_2 r_1 - q_1 r_2)/(p_1 q_2 - p_2 q_1), (p_1 r_2 - p_2 r_1)/(p_1 q_2 - p_2 q_1)\big)\,. \tag{10}$$

If the denominator, $(p_1 q_2 - p_2 q_1)$, is less than 2.5×10^{-5}, the points *1, 2,* and *3* lie on one line, and the point of gravity coincides with point *5*.

II. 2.4.2 CCS-es

A circle has two types of regions: a segment defined by an arc and its chord, and a sector defined by an azimuthal section of an annular shell; both are shown in Fig. 6. The segments must not be confused with the line or arc segments enclosing a region, nor must the sectors be confused with the angular sectors of the current discretization.

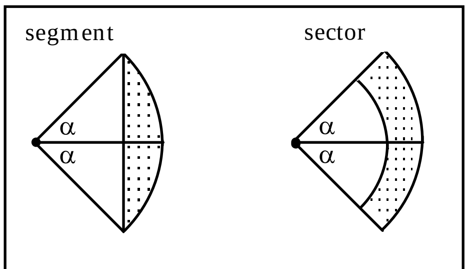


Fig. 6: Volumes and points of gravity in CCS-es.

A segment occurs when an arc segment is originally represented by a line segment. The volume and gravity point of the circle segment then have to be subtracted from the non-CCS region. A sector occurs when the volume and gravity point of a partial CCS have to be subtracted from a non-CCS region. Otherwise, sectors are the basic CCS regions—for an annular shell, $\alpha = \pi$, and for the central annulus the inner radius is zero.

First, for the segment and the sector of Fig. 6, formulas are given for the area and the distance between the gravity point and the CCS centre, *A* and *d*. Let the outer and inner radii be *R* and *r*, respectively. The formulas for the segment and sector supplements are also given; they are marked with an asterisk.

- For the segment of Fig. 6 and its supplement, one has:

$$\left.\begin{array}{lll} A = R^2(\alpha - \sin\alpha\cos\alpha) & \text{and} & d = \frac{2}{3}R^3\dfrac{\sin^3\alpha}{A} \\ A^* = R^2\big(\pi - (\alpha - \sin\alpha\cos\alpha)\big) & \text{and} & d^* = -\frac{2}{3}R^3\dfrac{\sin^3\alpha}{A^*} \end{array}\right\}. \tag{11}$$

- For the sector of Fig. 6 and its supplement, one has:

$$\left.\begin{array}{ll} A=(R^2-r^2)\alpha \quad \text{and} & d=\tfrac{2}{3}(R^3-r^3)\dfrac{\sin\alpha}{A} \\ A^*=(R^2-r^2)(\pi-\alpha) \text{ and} & d^*=-\tfrac{2}{3}(R^3-r^3)\dfrac{\sin\alpha}{A^*} \end{array}\right\}. \qquad (12)$$

Equations (11-12) give the distance between the gravity point and the CCS centre. To obtain the coordinates of the gravity point, consider three different situations. First, replace an earlier assumed straight-line segment by an arc; then its segment must be subtracted, and d is given by Eq. (11). Second, reduce the volume of a region by the volume of an entire or partial CCS; then an entire sector is subtracted, and d is given by Eq. (12) with $r=0$. Third, consider a CCS region that is a section of an annular shell (entire shells have their gravity point in the CCS centre).

The first two cases are discussed first. Referring to Fig. 7, the question is: knowing the points *0*, *1*, and *2*, and the distance d between the points *0* and *3*, what are the coordinates of point 3?

Line l_1, which passes through the points *1* and *2*, is given by Eqs. (8-9). Line l_2, which is perpendicular to line l_1 and passes through point *0*, is given by:

$$qx - py = qx_0 - py_0 . \qquad (13)$$

Point *3* also lies on line l_2. Its coordinates, (x_3,y_3), are found from the following set of equations:

$$\left.\begin{array}{l} (x_3-x_0)^2+(y_3-y_0)^2=d^2 \\ qx_3-py_3=qx_0-py_0 \end{array}\right\}. \qquad (14)$$

Depending on the sizes of p and q, the solution is:

$$\left.\begin{array}{ll} \left.\begin{array}{l} x_3 = x_0 \pm sd/\sqrt{s^2+1} \\ y_3 = y_0 \pm d/\sqrt{s^2+1} \end{array}\right\} & s=p/q \text{ and } |q|>|p| \\ \\ \left.\begin{array}{l} x_3 = x_0 \pm d/\sqrt{t^2+1} \\ y_3 = y_0 \pm td/\sqrt{t^2+1} \end{array}\right\} & t=q/p \text{ and } |p|>|q| \end{array}\right\}. \qquad (15)$$

The sign to be used in Eq. (15) follows from Fig. 8, which can be verified by inspection—note that p/q is negative in the shaded areas. The application is based on the position of the midpoint between *1* and *2* on l_1, (x_m, y_m), with respect to the centre of the circle. Thus: if s is used and $y_m \ge y_0$, or if t is used and $x_m \ge x_0$, the sign is plus; otherwise the sign is minus. The same Eq. (15) is used for the supplement, now with d negative.

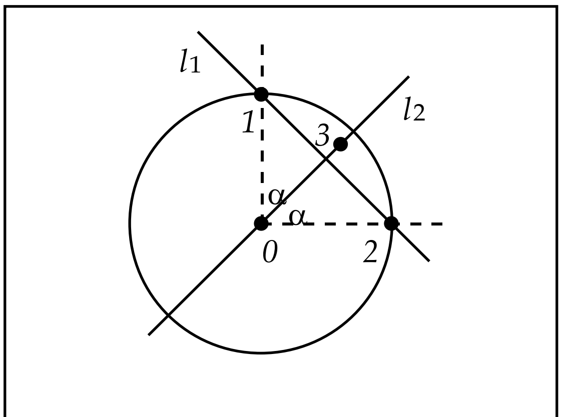


Fig. 7: Coordinates of gravity point.

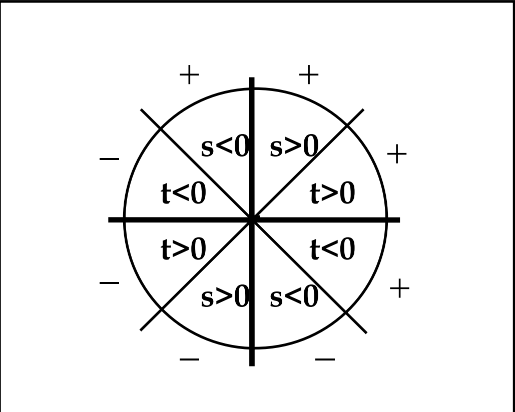


Fig. 8: Sign to be used in Eq. (15).

If the midpoint lies very near to the CCS centre, the wrong half of the CCS could be taken with the correct area but a wrong gravity point. To avoid this, a remedy has been built in. The points *1* and *2* are moved parallel to l_2 (along the lines l_3 and l_4) to the locations *4* and *5*, such that the line segment *4-5* (along l_5) has the CCS centre on the same side as the segment *1-2* has it. Its midpoint is then used instead of the old (x_m, y_m). The algorithm is:

$l_3 // l_2$:	$q\,x - p\,y = q\,x_1 - p\,y_1$.
$l_4 // l_2$:	$q\,x - p\,y = q\,x_2 - p\,y_2$.
Move *m*:	$(x_m, y_m) \rightarrow (x_m + p, y_m + q)$.
$l_5 // l_1$:	through new midpoint $\rightarrow p\,x + q\,y = p\,x_m + q\,y_m + p^2 + q^2$.
$l_5 \cap l_3$:	this yields (x_4, y_4).
$l_5 \cap l_4$:	this yields (x_5, y_5).
Test:	is (x_0, y_0) on the same side of *4-5* as of *1-2*?
Yes:	use $(x_m, y_m) \rightarrow (x_m + p, y_m + q)$.
No:	use $(x_m, y_m) \rightarrow (x_m - p, y_m - q)$.

The third case is the gravity point of a CCS region. Recalling section 2.2, the azimuthal subdivision of an annular shell starts with respect to the north direction, and the sections are equal. Let *a* be the north direction (for an entire CCS, $\alpha = 0$) and let there be *N* sections per 2π radians. The gravity point of the *n*th azimuthal region in the annulus between *r* and *R*, then lies at:

$$(x_c, y_c) = (x_0 + d\sin\varphi_n, y_0 + d\cos\varphi_n); \varphi_n = \alpha - \frac{2\pi}{N}(n - \tfrac{1}{2}). \tag{16}$$

With regard to this third case, two problems still have to be solved for a partial CCS: what is the angle α of the north direction, and what is the angular sector φ of the part of the CCS that lies inside the structure.

Reference is made to Fig. 9 and to section 2.2. The north direction, N, is along the radius from the centre to the second intersection, i.e., along *0-2*. Let α be measured clockwise from the positive y axis, and let the radius be R, then α (= φ_2) is obtained from:

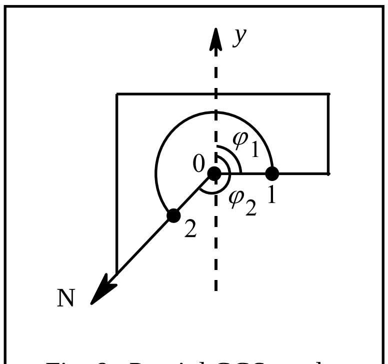


Fig. 9: Partial CCS angle inside a structure.

$$\left.\begin{aligned}\alpha &= \cos^{-1}\left(\frac{y_2 - y_0}{R}\right) && \text{if } x_2 - x_0 \geq 0\\ \alpha &= 2\pi - \cos^{-1}\left(\frac{y_2 - y_0}{R}\right) && \text{if } x_2 - x_0 < 0\end{aligned}\right\}. \quad (17)$$

The angle of the CCS sector inside the structure is the difference between φ_1 and φ_2. Both these angles, which are measured clockwise with respect to the positive y axis, are obtained according to Eq. (17). The sought sector angle then is found to be:

$$\left.\begin{aligned}\varphi &= \varphi_1 - \varphi_2 && \text{if } \varphi_1 \geq \varphi_2\\ \varphi &= \varphi_1 - \varphi_2 + 2\pi && \text{if } \varphi_1 < \varphi_2\end{aligned}\right\}. \quad (18)$$

II. 2.5 Notes on LINK04

The structures are treated in LINK04. Apart from purely administrative parts, its main five subprograms are:

PRESTR: It tests the nodes and sets up the line and arc segments.

TESTST: It tests for illegal segment intersections and creates additional, anonymous CCS nodes; see section 2.2.

VOLTRT: It tests that the regions don't overlap or intersect and that the regions have been specified to the right of their perimeters. It calls the function VOLUME and the subroutine CCSTRT for non-CCS and CCS regions, respectively.

VOLUME: It does the calculations described in section 2.4.1 to obtain the volumes and gravity points of the non-CCS regions. It calls the function VECPRO and the subroutine TRICGR to evaluate the area and gravity point of a triangle, and combines triangles into non-CCS regions. For the non-CCS regions, it calls the function CHAREA to evaluate volume and gravity point of a circle segment—these are needed when replacing line segments by arc segments.

CCSTRT: It obtains volumes and gravity points of the CCS regions and corrects the non-CCS regions for the presence of CCS-es; see section 2.4.2. It calls the functions CHAREA and ANGLEY, the latter to obtain the angles α and φ given by Eqs. (17-18).

Apart from the above-mentioned subprograms, there is a welter of smaller subroutines and functions that are used in the various geometric calculations. Most of these deal with trivialities, such as: the intersection of two segments; the distance between two nodes; or the overlapping of two CCS-es. Below, a couple of maybe less trivial methods are briefly sketched, and the involved subroutines and functions mentioned.

The logical function RIGHTX is often called to test if a node 3, (x_3,y_3), lies to the right of a segment defined by the nodes (x_1,y_1) and (x_2,y_2). Referring to Eq. (5) and Fig. 4, node 3 lies to the right of segment *1-2* if

$$\vec{v}_1 \times \vec{v}_2 = (x_2 - x_1)(y_3 - y_2) - (x_3 - x_2)(y_2 - y_1) < 0\,. \tag{19}$$

The logical function NODLIN checks if the node *3*, (x_3,y_3), lies on a line defined by the nodes (x_1,y_1) and (x_2,y_2). For node 3 to lie on the line, its coordinates must satisfy the equation of that line, which is given by Eqs. (8-9), i.e., $px_3 + qy_3 = r$. The logical function NODSEG also verifies if node 3 lies on the segment defined by these nodes 1 and 2. For node 3 to lie on the segment, its coordinates must lie between those of nodes 1 and 2. To avoid loss of precision, both tests depend on the slope of the line; they are summarized below:

$$\left.\begin{array}{lll} \underline{|p|<|q|}: & \text{on line if} & y_3 - (r - px)/q| < \varepsilon \\ & \text{on segment if also} & x_1 < x_3 < x_2 \\ & & \\ \underline{|p|\geq|q|}: & \text{on line if} & |\,x_3 - (r - qy)/p| < \varepsilon \\ & \text{on segment also if} & y_1 < y_3 < y_2 \end{array}\right\}. \tag{20}$$

NODSEG calls the logical function BETWTH for the inequality tests of the x and y coordinates. Presently, $\varepsilon = 4*$ AQRACY, where AQRACY $= 2.5\times10^{-4}$.

The function NODINC checks where a node lies with respect to a region: inside, outside, or on the periphery. In words, the test goes as follows. An arbitrary line is drawn through the node, and its intersections with peripheral segments to the left of the node are counted. If the number of intersections is odd, the node lies inside the region. If the arbitrary line coincides with a segment, the test is done with another arbitrary line through the node. If after five trials this still does not work, HELIOS terminates with an error message.

II. 3 CONNECTING STRUCTURES

Having defined the basic structures of the system, the next step is to combine them into the system. This buildup can be done gradually, by first connecting structures into <u>subsystems</u>. Initially, a subsystem is defined as any number of structures coupled together. However, as soon as at least one such a subsystem has been specified, the definition can be broadened to: a subsystem is the combination of any number of structures and/or earlier-defined subsystems. The final system, the one to be calculated, is called the system. It is the only subsystem to which boundary conditions are assigned; see section 4. The coupling of structures and subsystems is done in LINK05 and LINK06.

Figure 10 illustrates the system buildup. In the second case, *Cell* and *Gaps* are structures, *Row*, *Fuel* and *Box* are subsystems, and *BWR* is the system. Of course, the two systems may be built up differently. For instance, each system can be specified directly as one structure. However, this would destroy flexibility. First, the input is more complicated when the nodes and regions of an entire system rather than of its repetitive parts have to be specified. Second, because structures are the smallest space elements, the use of first-flight probabilities is enforced in the entire system. On the other hand, in a system that is built up of smaller structures the User has the choice to use first-flight probabilities in these structures (and couple them by currents), or in space elements made up of two or more structures.

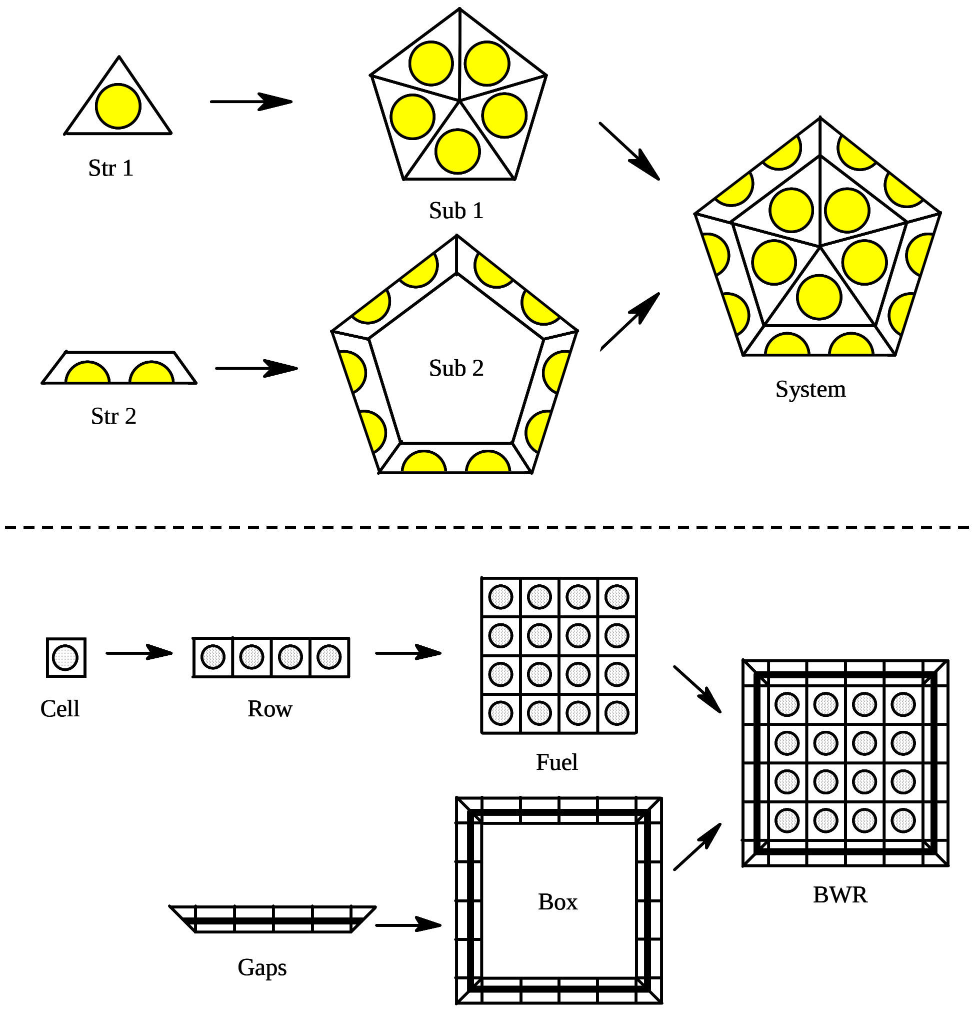


Fig. 10: Buildup of two different systems from structures and subsystems.

A subsystem is defined by its components—structures or subsystems—and for each coupling of two components by the two peripheral node pairs where they are connected. These nodes do not have to exist in the structure; their coordinates may be defined with the coupling, in which case they cannot be part of a region definition.

The coordinate system of the new subsystem is that of the first component. That is because the buildup of the subsystem starts by connecting the second component to the first one. Any next component is coupled to the cluster of already connected components. Thus, the coordinates of each component but the first are transformed into the coordinate system of the first component.

Let the two pairs of coupling nodes be *1*, *2*, *3* and *4*, as shown in Fig. 11a. The lines drawn between the node pairs do not have to represent peripheral segments of the two components to be coupled, but each of the nodes must lie on the periphery. First, translate nodes *3* and *4* such that nodes *3* and *1* coincide (see Fig. 11b); second, move

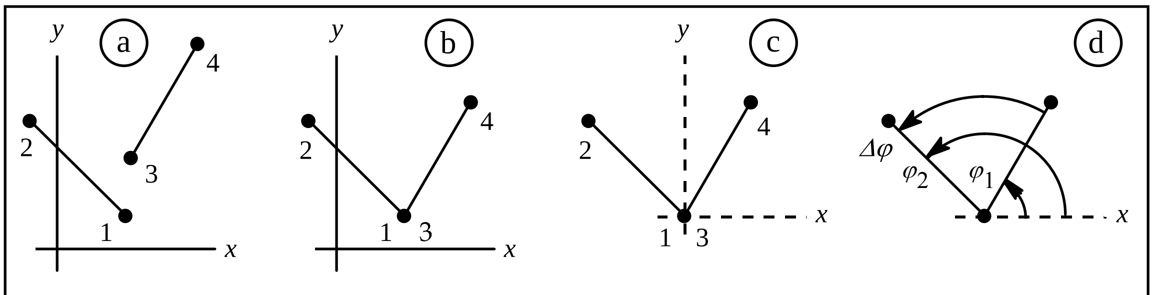


Fig. 11: The transformation of nodes 3 and 4 on nodes 1 and 2.

the coordinate system to node 1 (see Fig. 11c); third, rotate node 4 to node 2 (see Fig. 11d); and fourth, restore the coordinate system. If $\vec{v}_i$ is the vector to node *i*, the transformations are:

$$\left.\begin{aligned}
&1)\quad \vec{v}' = \vec{v} + \vec{v}_1 - \vec{v}_3 \\
&2)\quad \vec{v}'' = \vec{v}' - \vec{v}_1 \quad = \vec{v} - \vec{v}_3 \\
&3)\quad \vec{v}''' = \mathrm{R}\vec{v}'' \quad = \mathrm{R}(\vec{v} - \vec{v}_3) \\
&4)\quad \vec{v}'''' = \vec{v}''' + \vec{v}_1 \quad = \mathrm{R}\vec{v} - (\mathrm{R}\vec{v}_3 - \vec{v}_1) \equiv \mathrm{R}\vec{v} + \vec{t}
\end{aligned}\right\}, \tag{21}$$

where **R** is the rotational transformation over the angle $\Delta\varphi = \varphi_1 - \varphi_2$, given by

$$\mathrm{R} = \begin{pmatrix} \cos\Delta\varphi & -\sin\Delta\varphi \\ \sin\Delta\varphi & \cos\Delta\varphi \end{pmatrix}. \tag{22}$$

Thus, the transformation related to a single coupling consists of a rotation matrix and a translation vector. But there can be successive transformations - once becoming part of a subsystem, a structure may be transformed again when that subsystem is used in a later coupling. Each additional transformation is also applied to the previous transformation to yield the latest, total transformation.

When, as the result of a coupling, all the nodes of a component have been transformed, all the common peripheral segments (the interface between the components) are registered. This may result in the creation of anonymous nodes, each of which splits the segment where it is created into two segments, as shown in Fig. 12. There, two subsystems are coupled by the node pairs (*1*, *2*) and (*3*, *4*). The coupling interface consists of the two heavy lines. In the second subsystem, each pin-cell side consists of two segments, so the interface consists of four peripheral segments. In the first subsystem, each pin-cell side consists of only one segment. However, the coupling has created two anonymous nodes, dividing each of the original segments into two.

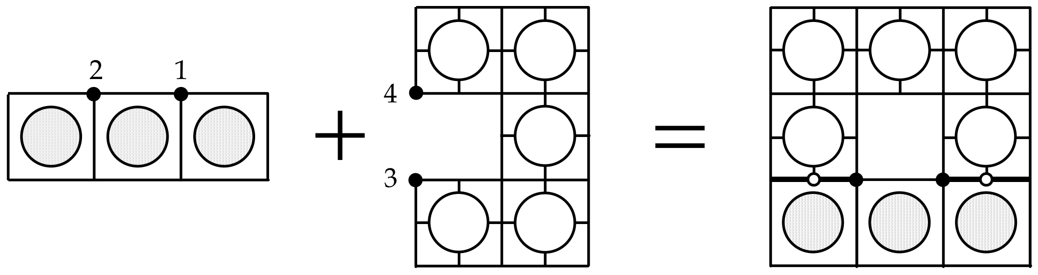


Fig. 12: Coupling interface and the creation of anonymous nodes.

Notice that the interface does not have to be coherent. In Fig. 12, a "hole" has been created. This hole may later be filled by another connection, or it remains a hole to which boundary conditions must be assigned.

Together with specifying the node pairs, a coupling order (k) must be specified. It assigns to the interface an angular discretization of the coupling currents that will be used in the transport calculations of HELIOS. The discretization is done by partitioning the directional half-sphere into sectors. The polar angle at each surface element is subdivided into a number of θ levels, and each θ level into a number of azimuthal φ intervals. Figure 13 shows a case of two θ levels, both of which are subdivided into the same four φ intervals. More details about the angular discretization are given in the chapters IV and V of this document.

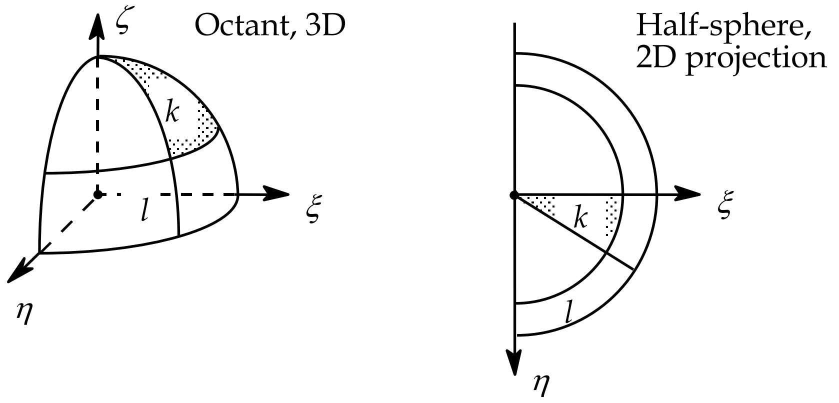


Fig. 13: Angular sectors in the local coordinate system of a segment.

Among the discretizations, $k = 0$ plays a special role. It means *no discretization*, i.e., the structures on either side of the segment are not coupled by currents. Instead, they are fusioned into one space element, inside which collision probabilities are used.

Initially, the same value of k is assigned to all the segments of a coupling interface. However, at any moment during the system buildup, the User can redefine k for each individual segment.

II. 3.1 Notes on LINK05 and LINK06.

The system buildup is treated in LINK05 and LINK06. Apart from purely administrative parts, the main six subroutines of the coupling are:

PRECNX:	It builds up the geometric information of a new subsystem by copying the nodes, segments, volumes, CCS data, gravity points, et cetera, of all their involved structures.
NODCNX:	It tests that the coupling nodes lie on peripheral, free (not-yet coupled) segments, and that the new nodes defined with the coupling do not lie inside CCS-es; see section 2.2.
VOLCNX:	It obtains and applies the transformations of Eqs. (21-22), both to nodes and to earlier transformations, by calling the subroutines TRANSF, NODTRA and DBLTRA. It tests that there are no overlapping regions.
SEGCNX:	It finds the coupling segments and gives them their k values.
CHANGK:	It redefines individual k values.
FUSION:	It rearranges the administration of structures that have been combined by $k = 0$ into superstructures. This is done in LINK06 after the boundary conditions have been assigned.

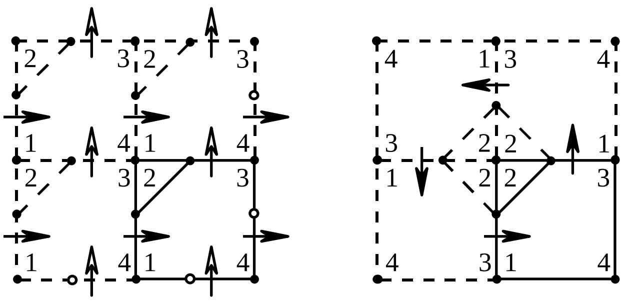
II. 4 BOUNDARY CONDITIONS

Once the system has been built up, a number of peripheral structure segments have not yet been coupled. They are arranged in one or more closed sets of segments. One set represents the outer boundary of the system. If there are more, the others represent internal boundaries, i.e., holes.

To each of these segments, defined by its first and second node, a boundary condition is assigned. If the same condition is assigned to several coherent segments then it suffices to specify the first node of the first segment and the second node of the last segment, keeping the system on the right.

Three different types of boundary condition can be assigned to a boundary segment: <u>Specular reflection</u> is incorporated at the level of the collision probabilities; see chapter V of this document. <u>Albedo reflection</u> defines the fraction of the out-currents that is reflected and into which energy group; see also section 5.6 and chapter IV of this document.

Boundary coupling prescribes that out-currents through a certain segment are in-currents through another segment, and vice versa. Of course, both of the involved segments must have the same length. This boundary condition allows for the description of periodic and rotational symmetries. In Fig. 14a, coupling faces *1-2* and *2-3* with faces *4-3* and *1-4*, respectively, results in periodicity. Here, each face consists of two boundary segments. In Fig. 14b, coupling face *1-2* with face *2-3*, results in rotational symmetry.



a: Periodic b: Rotational

Fig. 14: Boundary coupling.

II. 4.1 Notes on LINK06.

The boundary conditions are treated in LINK06. Apart from purely administrative parts, the main three subroutines of the boundary conditions are:

PRELBC: It finds the boundary segments and arranges them into one or more closed sets — the boundary groups.

NODEBC: It traces the input boundaries, and verifies that the nodes of a given boundary condition belong to the same group.

SOLVBC: It assigns the boundary conditions to the boundary segments.

II. 5 CROSS-SECTIONS AND ALBEDOS

To calculate particle fluxes and currents throughout the system, the properties that define its interactions with these particles must be known. Therefore, to each region a material composition, a temperature, and a density-correction factor are assigned. Together with the data of the nuclear-data library, this information is sufficient to evaluate the region properties, i.e., macroscopic cross-sections (XSs), fission spectra, and (n,γ) production matrices. Moreover, if there is albedo reflection, then as many albedo matrices must be specified as there are boundary segments with different albedos.

The assignment of material, temperature and density-correction overlays is done in LINK07. Since this is purely a programming matter it does not belong to *HELIOS*

Methods; it is discussed in chapter X of *HELIOS Program*. The other aspects of the properties are the topic of the present section.

In HELIOS, macroscopic XSs, fission spectra and (n,γ) matrices are evaluated in LINK14. The interpretation of the material and albedo input is done in LINK08. This includes the possible creation of multiple materials if an input material occurs in more than one region. In that case the material must be burnable, or it must have resonance isotopes, or it must have light isotopes and the regions must have different temperatures. Restart materials are constructed from dumps made by earlier calculations. AURORA retrieves their identifiers from these dumps. Their number densities are either retrieved by AURORA or by HELIOS—see section II.5.1.3.

II. 5.1 Materials and number densities

The materials that are read from the HERMES data-base, are specified by their components and the number densities of these components. A component is defined as an isotope (e.g., B-10), an element (e.g., boron), a molecule or an alloy (e.g., H_2O or steel), a tabulated burnable absorber (see below), or a special component like the $1/v$ absorber.

The isotope identifiers are of the type $1{,}000{*}Z + A$, where Z is the atomic number and A is the mass in atomic mass units, e.g., 92235 for U-235. The element identifiers are of the type 1,000*Z, e.g., 13000 for natural Al. Alloys and molecules have the same identifier type as isotopes, but now Z is the atomic number of the most important isotope, and A is the mass, or a special indicator, e.g., 1018 for H_2O and 26347 for a certain type of stainless steel. The other component types have identifiers of the type 99∗∗∗.

A material with burnable components is burnable. However, the User can specify a burnable material to be non-burnable, e.g., for boron dissolved in the coolant. There are three categories of burnable components:

- Fuel and fission-product isotopes of fissionable materials.
- Isotopes with strong absorption, often including their predecessors/successors in an absorption-decay chain, such as B-10 and B-11, and certain Gd, Hf and Er isotopes.
- Tabulated burnable absorbers (BAs) which represent a BA chain by a single component with library tables of effective absorption XSs. Their identifiers are 991∗∗ for Gd_2O_3, and 992∗∗ for arbitrary BA chains.

At zero burnup, a burnable material does not necessarily contain all the isotopes that will be present once it has been exposed to burnup. For example, fresh fuel initially does not contain transuranic isotopes or fission products. Such materials are completed to include all the isotopes of the burnup chains, even if no burnup will take place. Each added isotope gets a zero initial number density. The burnup of materials is discussed in chapter VII of this document.

A burnable material gives rise to as many individual burnable materials as regions to which it has been assigned. Each of these materials becomes a permanent material. A non-burnable material assigned to different regions also can give rise — temporarily—to multiple materials. Although its number densities do not change, its XSs can differ because of resonance shielding or scattering. The effective XSs of a resonance isotope depend through the Dancoff factor on its location in the system. Also, materials with scattering isotopes lighter than AWMAX in regions whose temperatures differ by more than DTMIN, have different scattering XSs. (Presently, AWMAX = 50 atomic mass units, and DTMIN = 20K.)

For reasons of computer efficiency, multiple burnable materials may share the same scattering matrix, so long as their temperatures differ by less than DTMIN. This would apply to UO_2, where it is a good approximation. The only light scatterer is oxygen, and its number density does not change with burnup. Further, U-238 is by far the predominant inelastic scatterer at high energies. Although subject to burnup, its concentration changes little during the lifetime of the fuel.

The above approximations can be overridden by defining for each region a material with a unique name; see "User's Manual for AURORA."

II. 5.1.1 Number densities

Although AURORA evaluates the input number densities to HELIOS and saves them in the HERMES data-base, their evaluation is described here, too. AURORA accepts number-density and weight-percentage (wt%) input. For the User's convenience, oxygen in oxide fuel may be specified with zero wt%. The wt% of a heavy isotope then refers to the aggregate of the heavy isotopes. Oxygen is then added according to AO_2 (A is the aggregate) plus that of Gd_2O_3 for BA of type 991**. Let ρ and c_i be the material density [g/cm^3] and the wt% of component i, then the number density N_i [atoms/barn-cm] is

$$N_i = 0.01 c_i \frac{\rho}{A_i} N_{Avo} \ , \tag{23}$$

where A_i is the atomic weight of i in atomic mass units, and N_{Avo} is the Avogadro number in units of 10^{24} nuclei per mole,

$$1 \text{ amu} = 1.660302\text{x}10^{-24} \text{ g} \ ; \quad N_{Avo} = 0.6023 \ . \tag{24}$$

If oxygen is specified with zero wt%, there are three possible component types: oxygen (indexed by O), BA, and heavy isotopes and the other components. Their number densities are given by (for 991**, see section 5.1.2):

$$\left.\begin{array}{ll} N_i = 0.01\, c_i \dfrac{\rho}{A_i} N_{Avo} & i \notin \text{heavy isotopes} \\ N_i = 0.01\, c_i \dfrac{\rho_h}{A_i} N_{Avo} & i \in \text{heavy isotopes} \\ N_O = 2 \sum\limits_{i \in \text{heavy}} N_i + 4.9180 \sum\limits_{i \in 991**} N_i & \text{for the oxygen} \end{array}\right\} , \tag{25}$$

where the density of the aggregate, ρ_h, is given by:

$$\rho_h = \frac{\left(1 - 0.01 \sum_{i \notin \text{heavy}} c_i\right)\rho}{1 + 0.02 A_O \sum_{i \in \text{heavy}} c_i / A_i} \ . \tag{26}$$

II. 5.1.2 Burnable absorbers (BAs)

There are two types of tabulated burnable absorbers, with the identifiers 991** and 992**. The BA of type **991**** represents only the two strongly absorbing isotopes of Gd_2O_3, Gd-155 and Gd-157. Because no other Gd-isotopes are treated, the residual Gd-absorption is not represented in the tables of effective absorption cross-sections of this type. The BA of type **992**** can represent arbitrary chains of one or more absorbing isotopes. For both types, the interpolation parameter in the tables is the fraction f that is left of the BA number density at time t, i.e.

$$f(t) = \frac{N_{BA}(t)}{N_{BA}(0)} \ , \tag{27}$$

where $N_{BA}(t)$ is the number density at time t. It is interpreted as the number of absorptions still available. With this number density, the interpolated effective cross-section gives the correct macroscopic absorption cross-section,

$$\hat{\Sigma}_{BA}(t) = N_{BA}(t)\hat{\sigma}_{BA}(f(t)) \ . \tag{28}$$

Because BA of type 991** represents only the isotopes Gd-155 and Gd-157, $N_{BA}(t) = N_{155}(t) + N_{157}(t)$. Once this BA is burnt out, $f(t) = 0$. For BA of type 992**, $N_{BA}(t)$ represents the number of absorptions still available, i.e., all the chain locations the nuclei can assume during the life of the BA. For example,

$$N_1(t) \to N_2(t) \to N_3(t) \tag{29}$$

is represented by the cumulative number density

$$N_{BA}(t) = 3N_1(t) + 2N_2(t) + N_3(t) \ . \tag{30}$$

If such a BA chain ends on a non-burnable isotope, the "burnt-out" BA has a non-zero pseudo-equilibrium value $f(t) > 0$. This represents the residual absorption which very slowly will approach zero if the last isotope has a finite absorption cross-section.

To evaluate number densities from wt% input, the atomic masses of the BAs in the library have been adjusted. BA of type 991** represents the two isotopes Gd-155 and Gd-157 in gadolinia, whose natural abundances are 0.148 and 0.157. So, per

Gd_2O_3 molecule there are 0.610 BA nuclei, which represent the total atomic weight of the gadolinia molecule of 362.475 amu. Hence, the weight assigned to a BA component of type 991** is 594.221 amu[1].

BA of type 992** represents an arbitrary chain of I isotopes i with natural abundances a_i. The atomic weight that produces the correct initial BA number density is given by

$$A_{992**} = \frac{\sum a_i}{\sum (I-i+1)a_i} \sum a_i A_i \quad \text{with } i = 1, \cdots I \ . \tag{31}$$

The numerator and denominator are proportional to the true and cumulative number densities of the chain isotopes combined; the factor on the right is their true initial atomic weight.

Adjusted atomic weights can only be used with fresh BA. Thus if $f<1$, number-density input must be used, not wt% input. Further, since oxygen in 991** already is accounted for in Eq. (26), it must not be included in the wt% of the oxygen in the rest of the material. However, if a BA of type 992** contains oxygen, its wt%'s must be explicitly specified. (Although in this case the option of zero wt% oxygen is allowed, it is not recommended because it neglects the BA's oxygen.)

For the striped BA of Fig. 15, the density and number-density input of HELIOS must be axially averaged according to the formula

$$x = (L_{BA}\, x_{BA} + L_{fuel}\, x_{fuel}) / (L_{BA} + L_{fuel}) \ . \tag{32}$$

HELIOS cannot use the same library as PHOENIX for BA of type 991**, because PHOENIX expects that the atomic masses in the library have been axially adjusted.

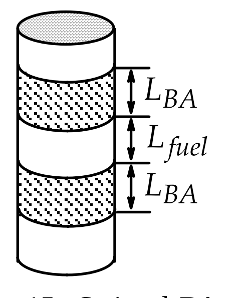


Fig. 15: Striped BA.

Representing a BA chain by a single component with pre-calculated tables of effective absorption XSs is one way to treat BA. Another way is to treat the isotopes of the chain explicitly, on-line instead of off-line. The former is the most computer-efficient, the latter is more exact but cannot be used for striped BA.

If gadolinia (Gd_2O_3), dysprosia (Dy_2O_3), or erbia (Er_2O_3) are to be treated on-line, they may be specified by the input identifiers 64000, 66000, or 68000. Their wt% c_i then refers to the entire material. This adds $0.1323c_i$, $0.1286c_i$, or $0.1254c_i$ to the wt% of the oxygen. The remaining wt% is assigned to seven Gd isotopes (Gd-152, Gd-154, Gd-155, Gd-156, Gd-157, Gd-158, Gd-160), five Dy isotopes (Dy-160, Dy-161, Dy-162, Dy-163, Dy-164), or four Er isotopes (Er-166, Er-167, Er-168, Er-170) according to their natural abundances—which are atomic fractions, not weight fractions.

II. 5.1.3 *Restart materials*

[1]The BA tables in the library are based on the abundances 0.1473 and 0.1568, so the weight assigned to a BA component of type 991** in the library still is 596.021 amu.

The TREE operator in AURORA allows HELIOS to do branch-off calculations in the same run as the burnup calculation, provided the geometry of the system does not change. Even inserting/extracting control blades is possible. In that case, the geometry description includes the control-blade regions, which can be filled with either gap water or control-blade materials. However, if the geometry changes, this is no longer possible—e.g., spent fuel in a storage rack, or a fuel assembly that is controlled during its life by geometrically different control blades. In such cases, the (fuel) compositions of the basic burnup calculations must be dumped in a HERMES file to be retrieved later for branch-off calculations in the changed geometry. These dumped materials are called restart materials; they are defined by the input operator DBMAT.

AURORA can represent such materials in two slightly different ways in the "User arrays". The old way, present since version 1.3, allows retrieval of a set of restart materials at one burnup level at a time. Their material compositions are stored as if they were defined by MAT operators. However, if the same set of restart materials is needed at more burnup levels, more DBMAT operators are required, each of which creates an extra set of restart materials. This taxes memory so heavily that branch-offs in changed geometries become virtually impossible.

Since version 1.7, the DBMAT operator can also define a set of restart materials at several burnups. In the "User arrays" they still look like materials defined by MAT operators, but at only one burnup level and with all number densities equal to -1. In that case, HELIOS retrieves their number densities at the desired burnup levels and stores them in the scratch file for use by the branch-off calculations as if they were produced from burnup calculations in the same run.

II. 5.2 Neutron cross-sections (XSs) and fission spectra

The macroscopic XSs that are used in the transport calculations must be evaluated whenever a new reactivity point is to be calculated. In principle, for a material made up of *I* components *i*, the macroscopic XS of type *x* is:

$$\Sigma_x = \sum_{i=1}^{I} N_i \sigma_{xi} \quad , \tag{33}$$

where, for component *i*, σ_{xi} is the microscopic XS of type *x*, and N_i its number density, including the density-correction factor of the region.

For most components, the microscopic XSs of the library can be inserted directly into Eq. (33). Not so for the resonance isotopes; in the energy groups of the resonance range, their microscopic XSs depend on their concentrations. Chapter III of this document describes how their effective microscopic XSs are calculated.

For a given component *i*, the library may contain XS tabulations at more than one temperature. In fact, for neutron XSs, HELIOS accepts two different temperature tabulations, one for P_1 energy-transfer XSs, and one for all the other XS types. Let *T* be the temperature of the region in which the material occurs, and let one of its isotopes i have more than one temperature tabulation. Then $\sigma_{xi}(T)$, its microscopic XS of type x at temperature T, is obtained by linear interpolation according to:

$$\sigma_x(T) = \frac{T_j - T}{T_j - T_{j-1}} \sigma_x(T_{j-1}) + \frac{T - T_{j-1}}{T_j - T_{j-1}} \sigma_x(T_j) \ , \tag{34}$$

where $T_{i\text{-}1}$ and T_j are the nearest tabulation temperatures below and above T. If T is smaller than the lowest tabulation temperature, T_1, or greater than the highest, T_J, then Eq. (34) becomes an extrapolation, with either $j\text{-}1 = 1$, or $j = J$. Of course, the library should be set up such that extrapolation is avoided.

As discussed in section 5.1.2, a BA chain can be treated by providing tables of effective absorption XSs. Interpolation in these tables uses the fraction that is left of the BA at zero burnup; see Eq. (27). This interpolation, upon which the temperature interpolation of Eq. (34) can be superimposed, is given by:

$$\hat{\sigma}_a(f) = \frac{f_j - f}{f_j - f_{j-1}} \sigma_x(f_{j-1}) + \frac{f - f_{j-1}}{f_j - f_{j-1}} \sigma_x(f_j) \ . \tag{35}$$

Extrapolation in Eq. (34) can be avoided if the two extreme values of f are included in the tables. These are $f = 1$ for fresh BA and $f = f_{min}$ for completely burnt BA, respectively.

Table 1 shows the macroscopic XSs and other neutronic data that are calculated at each reactivity point, for each material, in each energy group, g. Except for the fission spectrum, all are macroscopic XSs. They are built up according to Eq. (33), with the microscopic XSs duly interpolated or obtained from resonance calculations.

Table 1: Neutronic data calculated by HELIOS.

Σ_g	total, transport-corrected	χ_g	fission spectrum
Σ_{ag}	absorption	Σ_{0g}	P_0 scattering
Σ_{rg}	removal (= $\Sigma_g - \Sigma_{tr,g\leftarrow g}$)	$\Sigma_{\text{tr},g'\leftarrow g}$	P_0-scattering matrix, transport-corrected
Σ_{fg}	fission		
$(\nu\Sigma_f)_g$	fission-neutron production	Σ_{ng}	P_n scattering ($n = 1, 2, 3$)
$(\kappa\Sigma_f)_g$	fission-energy production	$\Sigma_{n,g'\leftarrow g}$	P_n-scattering matrix

Some quantities in Table 1 need an explanation. Whereas the microscopic $(\nu\sigma_f)_g$ is given in the library, $(\kappa\sigma_f)_g$ is found by combining κ_g and σ_{fg}, i.e., κ is energy-dependent. The P_1 data are used in the B_1 method to evaluate the criticality spectrum (see chapter VI) and the default transport correction (see below). In the transport calculations isotropic scattering is used. Linear anisotropic scattering is approximated by reducing the self-scattering elements of the P_0 matrix by the transport correction $\Delta\Sigma_g$,

$$\Sigma_{tr,g'\leftarrow g} = \Sigma_{0,g'\leftarrow g} - \delta_{g'g}\Delta\Sigma_g \quad \text{and} \quad \Sigma_g = \Sigma_{ag} + \Sigma_{0g} - \Delta\Sigma_g, \tag{36}$$

where, since HELIOS-1.9, the library has two definitions of the transport correction:

$$\left.\begin{array}{ll} \Delta\sigma_g = \sigma_{1g} = \sum_{g'} \sigma_{1,g'\leftarrow g} & \text{the outflow correction} \\ \Delta\sigma_g = \sum_{g'} \sigma_{1,g\leftarrow g'} J_{g'} / J_g & \text{the inflow correction} \end{array}\right\}. \tag{37}$$

In Eq. (37), the J_g are the group currents found by solving the P_1 equations in a finite homogeneous system of the isotope at a number density of $0.01x10^{-24}$ cm^{-3}. Leakage is described by a small buckling, $B^2 = 2.5x10^{-5}$ cm^{-2}, to simulate an almost critical system. These calculations are done by the code HEBE, when constructing the library.

The outflow correction, which is nothing but the P_1-scattering XS, is the default for most isotopes in the library. However, a few light isotopes—H and D bound in water, Be, C, O and Na—have the inflow correction. Yet, HELIOS enforces the outflow correction for all isotopes with P_1 data in the library, unless it has been specified in the input that the library's transport correction must be used.

For reasons of computer efficiency, there may be fission-product isotopes in the library with only absorption XSs. For these isotopes, it is assumed that $\sigma_i = \sigma_{ai}$. Otherwise, the library provides P_0-scattering XSs, transport-corrected P_0 matrices, and P_1-scattering matrices, though the latter are not always given. For components without P_1 data, the P_1-scattering matrix is assumed to be diagonal and is found from

$$\sigma_{1,g'\leftarrow g} = \delta_{g'g}\sigma_{1g} \quad \text{with} \quad \sigma_{1g} = \sigma_{0g} - \sum_{g'} \sigma_{tr,g'\leftarrow g} \ . \tag{38}$$

Finally, the fission spectrum of a mixture of isotopes is evaluated from

$$\chi_g = \sum_i \left(\chi_{ig} \frac{N_i w_i}{\sum_{i'} N_{i'} w_{i'}} \right) \quad \text{with} \quad w_i = \sum_g \nu\sigma_{fi,g}\phi_g \ . \tag{39}$$

Table 2: Fission-spectrum weights.

U-235	104.0	Pu-241	319.0
U-236	0.513	Pu-242	1.54
U-238	0.268	Am-241	1.32
Np-237	1.51	Am-242	2080.0
Pu-238	6.67	Am-242m	500.0
Pu-239	286.0	Am-243	1.15
Pu-240	1.79		

Table 2 shows the w_i values of thirteen fissionable isotopes—other fissionable isotopes get the w_i of U-235. They have been obtained from a PHOENIX calculation of an infinite square UO_2 lattice, at 20,000 MWd/t, defined as:

Fuel	-	920 Kelvin, radius 0.46 cm, 10 g/cm^3, 3 wt% enriched UO_2;
Clad	-	570 Kelvin, radius 0.54 cm, 6 g/cm^3, Zircaloy;
Coolant	-	560 Kelvin, pitch 1.43 cm, 0.74 g/cm^3, H_2O.

II. 5.3 Cross-section adjustments for (*n*,2*n*) and (*n*,3*n*)

II. 5.3.1 Reactivity effects

To avoid the use of (*n*,2*n*) and (*n*,3*n*) cross-sections in the transport calculations, their effect is included in the definitions of the absorption cross-sections and scattering matrices in the library. This is done as follows:

$$\left.\begin{aligned}\sigma_a &\to \sigma_a' = \sigma_a - \sigma_{n2n} - 2\sigma_{n3n}\\ \sigma_s &\to \sigma_s' = \sigma_s + 2\sigma_{n2n} + 3\sigma_{n3n}\end{aligned}\right\} . \tag{40}$$

This approximation preserves the total cross section, $\sigma' = \sigma$, and the number of secondaries, $c' = c$. The latter is defined as the average number of neutrons emerging from a collision, or

$$c = (\nu\sigma_f + \sigma_s + 2\sigma_{n2n} + 3\sigma_{n3n})/\sigma \ . \tag{41}$$

To also preserve the spectrum of the emerging neutrons f_s, it is modified as follows:

$$f_s \to f_s' = \frac{\sigma_s f_s + \sigma_{n2n} f_{n2n} + \sigma_{n3n} f_{n3n}}{\sigma_s + 2\sigma_{n2n} + 3\sigma_{n3n}} \ , \tag{42}$$

where f_{n2n} and f_{n3n} are normalized to 2 and 3, respectively.

II. 5.3.2 Depletion readjustments

The above adjustments are done for all isotopes that have (*n*,2*n*) and/or (*n*,3*n*) cross-sections in ENDF/B-VI, and have been present in all the HELIOS libraries since 1994. However, in the depletion calculations of chapter VII the (*n*,2*n*) and (*n*,3*n*) reactions must be included in the loss rate of an isotope, while the gain (production) rate of the next isotope in the chain must be due to capture only. Hence, the absorption cross-sections used to obtain the loss and capture rates should be re-adjusted for the (*n*,2*n*) and (*n*,3*n*) reactions as follows:

$$\left.\begin{aligned}&\text{Loss :} && \sigma_a' + 2\sigma_{n2n} + 3\sigma_{n3n}\\ &\text{Capture :} && \sigma_a' + \sigma_{n2n} + 2\sigma_{n3n}\end{aligned}\right\} . \tag{43}$$

These re-adjustments are made for all isotopes in the depletion chains with (*n*,2*n*) and/or (*n*,3*n*) cross-sections in the library. Until version 1.6, only a few isotopes had (*n*,2*n*) and/or (*n*,3*n*) reactions in the library. Since version 1.6, all isotopes with non-zero (n,2n) and/or (n,3n) cross-sections in ENDF/B-VI have these cross-sections also in the library.

II. 5.4 Energy release per fission

The total fission energy at zero energy of the incident neutron ET is given by (Rose and Dunford, 1990):

$$ET = EFR + ENP + END + EGP + EGD + \mathrm{EB} + ENU\,,$$

where

EFR	-	kinetic energy of the fission fragments;
ENP	-	kinetic energy of prompt fission neutrons;
END	-	kinetic energy of delayed fission neutrons;
EGP	-	total energy release by prompt gammas;
EGD	-	total energy release by delayed gammas;
EB	-	total energy release by delayed betas;
ENU	-	total energy carried off by neutrinos.

The term "fragments" includes all the charged particles that are emitted promptly, because they have short ranges and are considered to lose their energy locally. The dependence on the incident neutron energy E [eV] is found by adding $\delta E_i(E)$ to the different types i of energy release. They are, in units of eV:

$$\begin{gathered}\delta ENU = -0.100E \\ \delta EGD = \delta EB = -0.075E \\ \delta EFR = \delta END = \delta EGP = 0 \\ \delta ENP = 1.307E - 8.07{\times}10^6[\,\nu(E) - \nu(0)]\,.\end{gathered}$$

To account for capture gammas, the following term is added (Tasaka *et al.*, 1992):

$$EC(E) = 5.99{\times}10^6[\,\nu(E) - 1]\,. \tag{44}$$

So, with $E_i(E) = E_i + \delta E_i(E)$, the energy release in the reactor at neutron energy E is

$$\kappa(E) = ET - ENU + 1.157E - 8.07{\times}10^6[\,\nu(E) - \nu(0)] + 5.99{\times}10^6[\,\nu(E) - 1]\,. \tag{45}$$

Because in the Helios library $\kappa(E)$ is in W-s, $\kappa(E)$ of Eq. (45) must still be multiplied by $1.6019{\times}10^{-19}$ to convert it from eV to W-s.

Prior to HELIOS-1.6, the nuclear-data library contained for each fissile isotope a single κ value, obtained by averaging $\kappa(E)$ with a "typical" PWR flux spectrum. Since HELIOS-1.7, however, the library contains also $\kappa(0) = ET - ENU$ and Eq. (45) is used in the subroutine LIBDAT to define energy-group dependent κ's.

II. 5.5 Gamma cross-sections (XSs) and gamma source

The macroscopic XSs that are used in the gamma transport calculations must be evaluated whenever a new reactivity point is to be calculated. In principle, there are two types of XSs related to gammas. First, there are the XSs that describe the interactions of gammas with matter. Because gammas interact with electrons and not with nuclei, these XSs are the same for all isotopes of an element. For instance, all the Pu isotopes have the same gamma XSs. Also, there is no temperature or resonance dependence.

Second, there is the (n,γ) production matrix, which describes the gamma production caused by neutron interaction with the nuclei of a material. Being a nuclear reaction, this matrix is different for isotopes of the same element. It may even have tabulations at more than one temperature, in which case the interpolation is given by Eq. (34). In HELIOS, these temperatures may be different from the ones at which the P_1 data or the other XS data are tabulated.

The gamma-interaction XSs used by the transport calculations are Σ_g, Σ_{ag}, Σ_{rg} and $\Sigma_{tr,g'\leftarrow g}$; see also Table 1. They are constructed from microscopic XSs and number densities according to Eq. (33).

There is no pair-production XS. In the library, it has been incorporated in the other XSs such that the total (transport-corrected) XS and the number of secondaries per collision (σ_s / σ_t) are preserved[1]. The adjusted absorption and transport-corrected scattering XSs are:

$$\left.\begin{aligned} \sigma_{ag} &= \sigma_g^{pe} - \sigma_g^{pp} \\ \sigma_{tr,g'\leftarrow g} &= \sigma_{tr,g'\leftarrow g}^{c} + 2\delta_{g_p g'}\sigma_g^{pp} \end{aligned}\right\}. \tag{46}$$

In Eq. (46) the superscripts *pe*, *pp* and *c* denote the photoelectric effect (true absorption), pair production in which a high-energy gamma generates an electron-positron pair of which the positron annihilates with an electron producing two gammas of about 0.51 MeV each, and Compton (true) scattering, respectively. Further, g, g' and g_p are energy-group indexes for gammas, with g_p being the group around 0.51 MeV where the gamma pair appears.

The elements of the (n,γ) matrix in the library are $\sigma_{g\leftarrow n}$ or $\sigma_{g\leftarrow n}/\sigma_{an}$, with g and n being gamma and neutron-group indexes. The regular $\sigma_{g\leftarrow n}$ is given for almost all the components; the latter only for resonance isotopes in resonance groups and for BA tabulations in all groups. For resonance isotopes in the resonance groups, and BAs in all the groups, the (n,γ) elements have been divided by their unshielded absorption XSs. That is because their effective XSs depend on their concentrations. In actual situations their effective values are obtained by multiplying $\sigma_{g\leftarrow n}/\sigma_{an}$ by their effective absorption XSs, i.e.,

$$\hat{\sigma}_{g\leftarrow n} = (\sigma_{g\leftarrow n} / \sigma_{an})\hat{\sigma}_{an}\,. \tag{47}$$

[1] The library also contains microscopic XSs for pair production, but they are not used.

Observe that $\sigma_{g\leftarrow n}$ includes all types of neutron interaction, like capture, fission, (n,α) and inelastic scattering. In the case of fission, $\sigma_{g\leftarrow n}$ includes not only prompt fission gammas, but also delayed gammas from fission-product decay.

In HELIOS, the gamma-interaction XSs of BAs are adjusted by hardwired coding. BA of type 991** represents only the strongly-absorbing isotopes of Gd_2O_3, Gd-155 and Gd-157. But in gamma calculations all Gd isotopes have the same XSs, so the number density of all the Gd should be used. Assuming that neutron absorption by Gd just leads to another Gd isotope, the number density for the gamma calculations is $N_{BA}(0)/A_{Gd}$, with A_{Gd} the sum of the natural abundances of Gd-155 and Gd-157, i.e., A_{Gd}= 0.305.

BA 992** currently represents the neutron-absorption potential of the following Hf chain (natural abundances are shown in %)

$$\begin{array}{ccccccccc} \text{Hf-176} & \rightarrow & \text{Hf-177} & \rightarrow & \text{Hf-178} & \rightarrow & \text{Hf-179} & \rightarrow & \text{Hf-180} \\ 5\% & & 19\% & & 27\% & & 14\% & & 35\% \end{array}$$

At any time, this absorption potential is defined by

$$5N_{176} + 4N_{177} + 3N_{178} + 2N_{179} + N_{180} \,.$$

However, in gamma calculations the true Hf number density should be used. Assuming that neutron absorption just leads to another Hf isotope, the number density for the gamma calculations is $N_{BA}(0)/A_{Hf}$. Now A_{Hf} is given by

$$5*0.05 + 4*0.19 + 3*0.27 + 2*0.14 + 0.35 = 2.45.$$

Finally, the gamma source is found from number densities, (n,γ) data and neutron fluxes. Omitting the region index, the source in a group g is

$$Q_g = \sum_n \phi_n \sum_i N_i \sigma_{g\leftarrow n} + \sum_{n'} \phi_{n'} \sum_{i'} N_{i'} \hat{\sigma}_{g\leftarrow n'} \,, \tag{48}$$

with n' and i' the groups and isotopes for which Eq. (47) must be used.

II. 5.6 Albedos

While macroscopic XSs, fission spectra and (n,γ) matrices define the properties of the regions of the system, albedos define the properties of its boundary segments—save those with boundary coupling. The size of an albedo matrix is $G \times G$, where G is the number of neutron or gamma groups. Its element $a_{g'\leftarrow g}$ defines what fraction of the out-current in group g is reflected and contributes to the in-current in group g'. So at a given boundary segment, the in-current in group g due to the out-currents in all groups is given by

$$j_g^- = \sum_{g'} a_{g\leftarrow g'} j_{g'}^+ \,. \tag{49}$$

In Eq. (49) total currents are used, i.e., summed over the sectors in which the angular dependence of the currents has been discretized. So the albedo matrix only relates currents between energy groups, not between angular sectors. The application of the albedo boundary condition to sectors is discussed in more detail in chapter IV.

Black, or vacuum, boundary conditions are described by $a_{g'\leftarrow g} \equiv 0$. The often used boundary condition of total reflection is described by a diagonal matrix whose diagonal elements are 1 and whose off-diagonal elements are 0, i.e., $a_{g'\leftarrow g} = \delta_{g'g}$. Again, this says nothing about the angular discretization of this reflection.

No limitations have been put on the $a_{g'\leftarrow g}$. The sum of the elements of a column may be greater than 1, which implies a multiplicative boundary segment. Within limits, this is physically correct. However, the significance of negative elements is harder to envisage; yet they are allowed too. Of course, specifying unphysical albedos may jeopardize the convergence of the transport calculations.

II. 5.7 Notes on LINK08, LINK14 (and LINK18)

The material and albedo input are treated in LINK08. Apart from purely administrative parts, the main six subroutines are:

SOLVMT:	It calls DENSMT to evaluate the initial heavy-metal densities of burnable materials. Then it calls DEPLMT to complete the burnup isotopes of burnable materials and to analyze the isotopes of the linearized burnup chains (by calling OCURMT and BREAKS). It also saves information needed to decide if a material can become a multiple material.
COUPMT:	It analyzes the locations of materials in regions with different temperatures, taking into consideration different overlay combinations.
LOADMT:	It sets up a material map, the array REGMAT(), for each overlay combination, and stores it in a depot of material maps.
RSTART:	It retrieves, for restart, materials with dumps at more than one burnup level their number densities at the burnups needed for branch-off calculations. These number densities are added to the depot of material compositions. It calls the subroutine RSDEPO.
RSDEPO:	It is called from RSTART and replaces in the number-density array of all the materials, ISODNS(), negative values (-1.0) of restart isotopes by their values at a given burnup level. RSTART will then store a copy of this ISODNS() in the depot of material compositions.
ALBDOS:	It analyzes the albedo input.

The evaluation of the macroscopic XSs, fission spectra, (n,γ) matrices and gamma sources is done in LINK14. The main three subroutines are:

INTPOL:	It sets up data for the XS, P_1, and (n,γ) temperature interpolations, and for the BA fraction interpolations.

NEUSIG: It evaluates the neutron XSs, fission spectra and P_1 matrices by calling SIGMA1 and P1MATR. Both subroutines call PACKER to compact the scattering matrices, $\Sigma_{tr,g'\leftarrow g}$ and $\Sigma_{1,g'\leftarrow g}$, by eliminating per row leading and trailing zeros.

GAMSIG: It evaluates gamma XSs, (n,γ) matrices and gamma sources by calling SIGMA2, NGMATR and GAMSOR, respectively. Now the subroutine PACKER is called from SIGMA2 to compact $\Sigma_{tr,g'\leftarrow g}$.

The re-adjustment of the microscopic loss and gain rates for built-in (n,2n) and/or (n,3n) adjustments in the library XSs, Eq. (43), is done in LINK18 in the subroutine:

PREBUR: See chapter VII.

REFERENCES

ROSE P F and DUNFORD C L (1990) *ENDF-102, Data Formats and Procedures for the Evaluated Nuclear Data File ENDF-6*, BNL-NCS-44945 (Informal Report, Limited Distribution), page 1.15. [The factor 8.07 in δENP has been corrected to 8.07×10^{6}.]

TASAKA K *et al*. (1992) Review of Decay Heat Study and Standards, *Proc. Specialists Meeting on Fission Product Nuclear Data*, 25-27 May 1992, Tokai, Japan.

III. HELIOS: RESONANCE TREATMENT

III. 1 INTRODUCTION AND SUMMARY

The energy dependence of the cross-sections (XSs) in the library has been discretized by dividing the energy range of interest, from 20 MeV to 10^{-4} eV, into a number of broad groups. These XSs have been obtained by flux-averaging point XSs—sometimes more than 100,000 points—with typical reactor spectra. Groups and spectra should have been chosen such that the library XSs are insensitive to the variations of the flux spectra in their anticipated range of application. It is unfortunate that this procedure is impractical for the resonance isotopes in the range from about 100 keV to about 1 eV. In that range, their XSs exhibit many resonances and hence behave so wildly that thousands of groups would be required for a satisfactory discretization. The use of XS libraries with that many groups would make reactor design calculations prohibitively slow and expensive.

To keep the number of library (and computational) groups manageable, the resonance behavior of the XSs is not considered when determining the library's group structure. The number of resonance groups is thus limited to, say, 5 to 30. This comes at a price, though. It saddles the computer programs for reactor design with undoubtedly their most intricate task, the evaluation of effective microscopic resonance-group XSs from the formula

$$\sigma_x = \frac{\int_{\Delta u} \sigma_x(u)\phi(u)\mathrm{d}u}{\int_{\Delta u} \phi(u)\mathrm{d}u}. \tag{1}$$

In Eq. (1) the energy variable E has been replaced by the lethargy $u = \ln(E_0/E)$, with $E_0 = 10$ MeV; $\sigma_x(u)$ is the microscopic XS of type x (absorption, fission, ...); $\phi(u)$ is the flux per unit lethargy; and Δu is the width of the lethargy group. Observe that while the neutrons lose energy during slowing down, they gain lethargy. For example, the energy ranges 20 MeV - 10^{-4} eV and 100 keV - 1 eV correspond to the lethargy ranges -0.6931 - 25.33 and 4.61 - 16.12.

The resonance behavior of $\sigma_a(u)$ complicates the evaluation of group XSs. This is aggravated by the effect of the resonances on the flux $\phi(u)$, which will have dips at and near their lethargies. Not only do these dips depend on the resonances, but also on the concentration of the resonance isotopes and on their location, inside a fuel pin or near its border with a non-resonance region. Even XSs without resonances can be affected because of the flux averaging. For example, the out-scattering XS behaves exponentially with u and may change appreciably inside a group.

The objective of the resonance treatment is to evaluate effective XSs for the resonance isotopes, in all the regions where they occur and in all the resonance groups. The library data available for this purpose are tables of group **resonance integrals**

(RIs)[1], the numerator of Eq. (1). These tables are either for homogeneous mixtures of the resonance isotopes with hydrogen, or for a pin in light water with mixtures of the resonance isotopes and UO_2, Homogeneous tables are present in old libraries that are used with HELIOS versions 1.3 and earlier. Since version 1.4, released in mid-1996, heterogeneous tables are used.

In sections 2-4, a system with one resonance absorber is considered. In section 2 the infinite homogeneous system is discussed. By introducing the intermediate resonance factor, λ, an expression for the flux is derived. It turns out that both the RI and the group flux—numerator and denominator of Eq. (1)—are a function of the same dilution parameter, the background XS, σ_b. Isotopes other than hydrogen are represented by including their λ's in σ_b. Using this σ_b, the RI can be obtained, and ultimately the group XS.

Section 3 is devoted to the subgroup method. In this method, the RI and group flux are converted into quadratures in the (resonance) absorption XS. The quadrature weights and discrete absorption XSs, w_n and σ_n, are obtained from the RI tables in the library. Earlier, this calculation was based on homogeneous RIs and was done in HELIOS. Since version 1.4, the subgroup data are present in the library; they have been obtained from heterogeneous RIs and included in the library by the library service program HEBE.

In section 4, the heterogeneous system is discussed. The equivalence of its flux with that in a homogeneous system is enforced by adding an equivalence XS, Σ_e, to the background XS. The lethargy variation of Σ_e is assumed to be through the absorption XS only. It is found throughout the heterogeneous system from a few transport calculations. Thus, Σ_e can be evaluated at all the σ_n values of the quadrature, and so can the flux (equivalence) and the group XS.

Sections 5 and 6 describe the interaction of resonance isotopes and the averaging of the removal XS. Now the subgroup method is invalid, because lethargy dependence is not through the absorption XS of one isotope only. In section 5, each isotope is treated separately, representing the absorption of the others by their average XSs. In section 6, the flux dips are assumed to be caused by either one resonance placed at the lethargy centre of the group, or by a uniform resonance distribution inside the group.

In section 7, finally, the computational flow is outlined. The transport calculations to find Σ_e can be done for all the resonance groups combined or for each group separately. Besides, they can be done per category of resonance isotopes, with only the isotopes of the actual category "switched on".

III. 2 THE HOMOGENEOUS SYSTEM

The main objective of this section is to derive an expression for the flux in a homogeneous—infinite—system containing a mixture of isotopes, indexed i, of which one is a resonance absorber. First, some assumptions:

[1] It is rather RIs divided by group widths, Δu, that are tabulated in the library.

1) Non-resonance isotopes have negligible absorption and a constant potential (scattering) XS, σ_{pi}. This assumption is valid throughout the resonance range from a few eV up to 100 keV.
2) Resonance isotopes have, apart from σ_{pi}, a resonance absorption and scattering XS, $\sigma_{ai}(u)$ and $\sigma_{rs,i}(u)$, which vary with u at and near the resonances, and are negligible away from them. This assumption is not strictly valid in view of the $1/v$ absorption. However, the RI tables do include $1/v$ absorption, while its effect on the flux cancels to first order in Eq. (1).
3) The resonances are so well-separated that the flux $\phi(u)$ between them has its constant asymptotic value, henceforth set to 1. This flux, which is used in the definition of the RI, accounts only for the dips due to the local shielding effects of the resonances. The global (gradual) flux depletion caused by absorption is proportional to the resonance escape probability $p(u)$ [Eq. (10.19*b*) of Weinberg and Wigner, 1958; Rothenstein, 1980]. Thus, while the true flux is $p(u)\phi(u)$, it is $\phi(u)$ that is considered here. However, it will be shown below that in Eq. (1) the effects of $p(u)$ can be made to cancel to second order in 1-p, where p is the resonance escape probability of the group Δu.

The slowing-down equation at resonance lethargies, and away from fission sources, is [Eq. (10.8*a*) of Weinberg and Wigner, 1958]

$$\Sigma(u)\phi(u) = Q(u)\,, \tag{2}$$

where the source $Q(u)$ is exclusively due to slowing-down,

$$Q(u) = \sum_i \int_{u-\Delta_i}^{u} \Sigma_{si}(u')\phi(u')\frac{\exp(u'-u)}{1-\alpha_i}\mathrm{d}u'\,. \tag{3}$$

The notation that has been adopted in Eqs. (2-3) is:

$$\left.\begin{aligned}\Sigma_{xi} &= N_i\sigma_{xi}\\ \Sigma_{si}(u) &= \Sigma_{pi} + \Sigma_{rs,i}(u)\\ \Sigma_i(u) &= \Sigma_{si}(u) + \Sigma_{ai}(u)\\ \Sigma(u) &= \sum_i \Sigma_i(u)\\ \alpha_i(u) &= (A-1)^2/(A+1)^2\\ \Delta_i &= -\ln(\alpha_i)\end{aligned}\right\}. \tag{4}$$

In Eq. (4): N_i is the number density of isotope i; $\Sigma_{si}(u)$ is the macroscopic scattering XS of isotopes i; $\Sigma_i(u)$ is the macroscopic total XS of isotope i; $\Sigma(u)$ is the macroscopic total XS of the mixture; $1-\alpha_i$ is the maximum fractional energy loss per collision with isotope i of mass A_i; and Δ_i is the maximum lethargy gain per collision with isotope i of mass A_i.

III. 2.1 The intermediate resonance (IR) approximation

To cast the source term in a tractable form, the intermediate resonance (IR) approximation is introduced (Goldstein and Cohen, 1962). For isotope i, the fraction λ_i of its scattering is assumed to be so effective that the maximum lethargy gain per collision is significantly greater than the practical resonance width[2], the narrow resonance scattering. The resonances act in very small intervals of the integration range from $u\text{-}\Delta_i$ to u, and their contributions to $Q(u)$ are negligible. Since outside the resonances $\Sigma_{si}(u) = \Sigma_{pi}$ and $\phi(u) = 1$, the contribution of isotope i to $Q(u)$ is $\lambda_i\Sigma_{pi}$. Thus, this fraction of the scattering generates an unperturbed slowing-down source, while it acts as out-scattering (removal) if it occurs within the practical resonance widths.

The remaining fraction, 1-λ, is assumed to be so ineffective that neutrons gain a negligible amount of lethargy compared with Δu, the wide resonance scattering. The resonances are so wide that $\Sigma_{si}(u')\phi(u')$ can be replaced by its average value, yielding a source contribution of $(1\text{-}\lambda_i)\Sigma_{si}(u)\phi(u)$. Thus, this fraction of the scattering does not provide source neutrons from outside the resonance widths; its source contribution should rather be considered as due to self-scattering, which neither adds nor removes neutrons.

With the source contribution simplified by the IR approximation, the flux can be solved from Eq. (2) as

$$\phi(u) = \frac{\lambda\Sigma_p}{\Sigma_a(u) + \lambda\Sigma_s(u)}, \tag{5}$$

where $\lambda\Sigma_p$ and $\lambda\Sigma_s(u)$ of the mixture are defined from the general formula

$$\lambda\Sigma_{\text{x}}(u) = \sum_i \lambda_i \Sigma_{xi}(u). \tag{6}$$

Observe the physical significance of Eq. (5). The flux satisfies the balance that source rate equals removal rate, where both absorption and out-scattering remove neutrons from the widths of the resonances. Furthermore, if there is only wide resonance scattering, $\lambda = 0$; then there is no source and, hence, no flux. Physically, this corresponds to the situation where the neutrons cannot cross the resonances and none are left for further slowing down.

III. 2.2 The resonance integral (RI)

In this section it is described how the RI of a resonance isotope in a resonance group is obtained by interpolation in tables of RIs.

[2] The practical width is that part of the resonance where the total scattering XS is at least twice the potential XS.

III. 2.2.1 The background cross-section

Let N be the number density of the only resonance isotope in the mixture. Further, define the background XS of this mixture as

$$\Sigma_b = \lambda\Sigma_p \;\Leftrightarrow\; \sigma_b = \frac{1}{N}\sum_i N_i\lambda_i\sigma_{pi}\,. \tag{7}$$

Then, dividing both numerator and denominator of Eq. (5) by N, results in

$$\phi(u) = \frac{\sigma_b}{\sigma_a(u) + \lambda\sigma_{rs}(u) + \sigma_b}\,. \tag{8}$$

The lethargy dependence of the flux is only through the resonance XSs; its importance is determined by σ_b. At high concentrations of the resonance isotope, σ_b approaches its smallest value, $\lambda\sigma_p$ of the resonance isotope. Then the flux dips due to the resonances are deepest. At low concentrations, that is high dilutions, σ_b becomes very large and the flux approaches its asymptotic value 1, regardless of the resonances. Thus, both the RI and the group flux in Eq. (1) are solely a function of the microscopic background XS, which plays the role of a dilution parameter.

III. 2.2.2 The hydrogen-equivalence factor

The library RIs are based on accurate solutions of the slowing-down problem, Eqs. (2-3), in mixtures of hydrogen and the resonance isotopes at various dilutions. They are tabulated as a function of σ_b, where for hydrogen the IR factor λ has been defined as 1. For other isotopes, λ has been obtained by comparing solutions in U-238/H mixtures with mixtures where the hydrogen was partly replaced by the other isotopes (Aldous, 1969). This transforms λ into a hydrogen-equivalence factor, whereby the RI tables can be used for all the isotopes, provided σ_p is replaced by $\lambda\sigma_p$.

Other resonance absorbers than U-238 have different λ values. In fact, the library contains λ tables with respect to all the resonance absorbers. Presently, since U-238 is by far the most predominant resonance absorber in most reactors, only λ values with respect to U-238 are used.

It should be recalled that λ depends on the resonance width, which varies from one resonance to another; some can be considered narrow, others wide. So $\lambda\sigma_p$ varies with lethargy, although σ_p is constant. That is why the library has group-dependent λ's for the isotopes.

III. 2.2.3 Interpolation in resonance-integral tables

For each resonance isotope and in each resonance group, the library contains tables of absorption RIs. If an isotope is fissionable, its ν•fission tables are also given. The two table entries are temperature, T, and the dilution parameter, σ_b. RIs at any T and σ_b are obtained from interpolation in these tables.

Because the temperature dependence of the overall RI is linear in $\sqrt{T}$, the interpolation in the tables will also be linear in $\sqrt{T}$. This dependence is due to the Doppler broadening of the resonances. The $\sqrt{T}$ behavior is only used to interpolate in exact RI values per resonance group, so it can also account for group-to-group variations of the Doppler effect.

For the interpolation in σ_b, the method of Segev (1981) is used. It interpolates in tabulated R values, which are the RIs divided by the group width Δu, according to

$$f(\sigma) = \frac{R(\sigma)}{R_\infty} = \left(\frac{\sigma}{\sigma + \eta}\right)^p . \tag{9}$$

In Eq. (9), $R_\infty = R(\infty)$, while the constants p and η have to be determined from the two table entries $R_1 = R(\sigma_1)$ and $R_2 = R(\sigma_2)$, where $\sigma_1 < \sigma_2$ so that $R_1 < R_2 < R_\infty$. If any of the latter inequalities is not true by better than 1%, p is set to 1 and η is solved from Eq. (9) using only R_1.

To obtain p and η, Eq. (9) is recast in the following form

$$g^\omega - 1 = \frac{\eta}{\sigma} \quad \text{with} \quad g = 1/f \quad \text{and} \quad \omega = 1/p . \tag{10}$$

Next, σ_1 and σ_2, and their corresponding $g_1 > g_2 > 1$ are substituted in Eq. (10). Eliminating η by dividing the two equations, leads to an equation for ω,

$$G(\omega) = g_2^\omega - \rho g_1^\omega - (1-\rho) = 0 \quad \text{with} \quad \rho = \frac{\sigma_1}{\sigma_2} < 1 . \tag{11}$$

The two possible variants of $G(\omega)$ are shown in Fig. 1. Eq. (11) has the two roots 0 and ω_0, the latter of which is the sought root. It is found by repeatedly halving the ω interval between ω_1 and ω_2, where $G(\omega)$ has its maximum at ω_1, and ω_2 is a value such that the interval (ω_1, ω_2) certainly contains ω_0. [To avoid numerical problems if $|\omega_2| \ln g_2 > \ln(10^6)$, $\sqrt{\sigma}$ interpolation is used if $\sigma < \sigma_2$, while Eq. (9) with $p = 1$ is used to interpolate between σ_2 and R_∞ if $\sigma > \sigma_2$.]

The value $\omega = \omega_1$, at which $G(\omega)$ has its extreme value, is found from

$$\left.\frac{\mathrm{d}G(\omega)}{\mathrm{d}\omega}\right|_{\omega_1} = 0 \Rightarrow \omega_1 = \ln\left(\rho \frac{\ln g_1}{\ln g_2}\right) \Big/ \ln\left(\frac{g_2}{g_1}\right) . \tag{12}$$

That this is a maximum follows from the fact that the second derivative is negative,

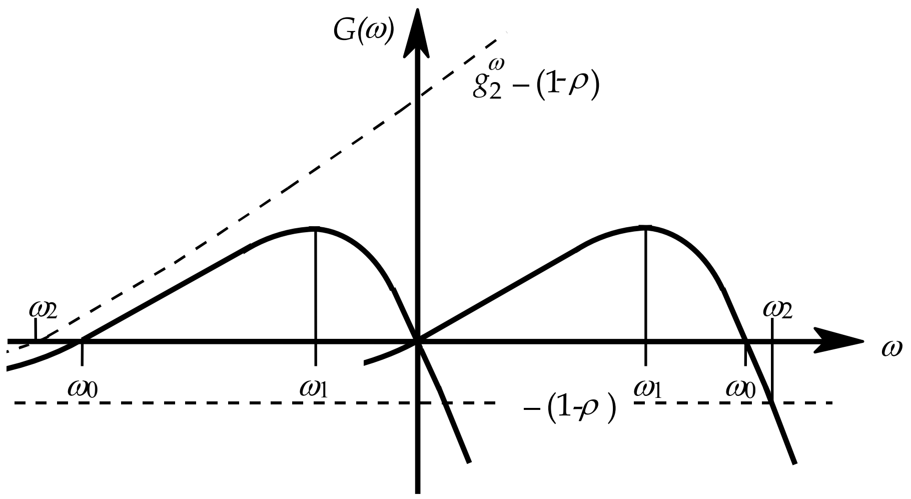


Fig. 1: Segev interpolation - searching the roots of G(ω) = 0, with ω =1/p .

$$\left.\frac{d^2G(\omega)}{d\omega^2}\right|_{\omega_1} = g_2^{\omega} \ln g_2 \ln\left(\frac{g_2}{g_1}\right) < 0 . \tag{13}$$

In the first variant of Fig. 1, which is not discussed by Segev, $\ln g_2 < \rho \ln g_1$ and $\omega_1 < 0$; in the second variant, $\omega_1 > 0$. The corresponding ω_2 values are found from

$$\left.\begin{array}{lll} G(\omega_2) + \rho g_1^{\omega_2} = 0 & \Rightarrow & \omega_2 = \dfrac{\ln(1-\rho)}{\ln g_2} \\ G(\omega_2) + (1-\rho) = 0 & \Rightarrow & \omega_2 = \dfrac{\ln \rho}{\ln(g_2/g_1)} \end{array}\right\} . \tag{14}$$

III. 2.3 Averaged resonance cross-sections

The flux of Eq. (8), here multiplied by the resonance escape probability $p(u)$, is used to obtain the group-averaged absorption XS from Eq. (1),

$$\sigma_a = \frac{\int_{\Delta u} \sigma_a(u) p(u) \phi(u) du}{\int_{\Delta u} p(u) \phi(u) du} . \tag{15}$$

Knowing that $p(u)$ varies only slightly in the group, a mean-value theorem is used to remove $p(u)$ from the integrands, leading to

$$\sigma_a = \frac{\overline{p} R}{\widetilde{p} - \overline{p}(R + \lambda R_s)/\sigma_b} . \tag{16}$$

In Eq. (16), R and R_S are the absorption and scattering RIs of the group divided by its width Δu (it is R values that are tabulated in the library), while the RI-averaged and lethargy-averaged resonance escape probabilities, $\overline{p}$ and $\tilde{p}$, are given by

$$\overline{p} = \frac{\int p(u)\mathrm{d}R}{R} \quad \text{and} \quad \tilde{p} = \frac{\int p(u)\mathrm{d}u}{\Delta u}. \tag{17}$$

It will be assumed that the two differently averaged resonance escape probabilities are equal, so that they cancel from Eq. (16). This is true if there are many resonances distributed uniformly in Δu. If there are few resonances an error is incurred whose order of magnitude is assessed here for the cases of one resonance located at the beginning, centre and end of Δu. In those cases, the lethargy-averaged resonance escape probability is p, $(p+1)/2$ and 1, respectively, where p is the resonance escape probability of the group.

To derive an expression for the RI-averaged resonance escape probability, a well-known relation between p and R is invoked, Eq. (10.19*a*) of Weinberg and Wigner (1958), which can be written as

$$p = \exp(-cR) \quad \text{with} \quad c = \frac{N\Delta u}{\xi\Sigma_s}. \tag{18}$$

In Eq. (18), $\xi\Sigma_S$ is the slowing-down power of the mixture which is irrelevant in the present treatment. From Eq. (18) follows that $\mathrm{d}p(u) = -cp(u)\mathrm{d}R$, which when introduced in Eq. (17) gives $\overline{p} = (p-1)/\ln(p)$. Expansion of $1/\overline{p}$ in terms of $\varepsilon = 1-p$ yields for the three locations of the resonance:

$$\begin{cases} 1 \mp \tfrac{1}{2}\varepsilon + \mathrm{O}(\varepsilon^2) & \text{beginning, end} \\ 1 + \tfrac{1}{12}\varepsilon^2 + \mathrm{O}(\varepsilon^3) & \text{centre.} \end{cases} \tag{19}$$

Thus, with judiciously chosen resonance groups, $\overline{p}$ and $\tilde{p}$ cancel to a high degree of accuracy from Eq. 16). Another simplification is the elimination of λR_S from its denominator. The reason for doing so is that, apart from simplifying Eq. (16), it is not customary to deal with scattering RIs; they often do not exist in the library. The justification for doing so is that the library's absorption RIs have been adjusted to yield the true σ_a if the λR_S term is neglected. Therefore, resonance scattering will henceforth not be considered any more. The final equation for σ_a becomes

$$\sigma_a = \frac{R(\sigma_b)}{1 - R(\sigma_b)/\sigma_b}. \tag{20}$$

The RI tables in the library have been obtained from Eq. (20) using σ_a's evaluated at given σ_b values (i.e., λ's) from exact solutions of the slowing-down problem. At those σ_b values, Eq. (20) reproduces the exact σ_a values. As was shown above, the fact that σ_a has been averaged with $\phi(u)$, and not with $p(u)\phi(u)$, leads to negligible errors. For other σ_b values, Eq. (20) is an interpolation formula whose errors cancel to a high degree.

Comparing Eq. (20) with Eq. (15) shows that the denominator is the group flux, ϕ, while the term R/σ_b represents its average dip. Because the library contains also RIs for ν·fission, the group-averaged ν·fission XS is found from R_ν/ϕ, which is divided by ν to get the fission XS

$$\nu\sigma_f = \frac{R_\nu(\sigma_b)}{1 - R(\sigma_b)/\sigma_b} \quad \text{and} \quad \sigma_f = \frac{\nu\sigma_f}{\nu}. \tag{21}$$

III. 3 THE SUBGROUP METHOD

Eq. (20) can be readily used in homogeneous systems. In heterogeneous systems, this is not straightforward. Often, the heterogeneous system is assumed to be equivalent to one or more homogeneous systems, represented by one or more σ_b's (Stamm'ler and Abbate, 1983). In HELIOS, σ_b is assumed to be energy-dependent, while, in the spirit of the subgroup method (Khairallah and Recolin, 1972; Roth, 1974; Notari and Garraffo, 1987), the numerator and denominator of Eq. (1) are approximated by quadratures in the absorption XS.

III. 3.1 From lethargy integral to cross-section quadrature

To arrive at Eq. (20), resonance scattering was eliminated, though not neglected. Consistent with this elimination, the flux of Eq. (8) is written as

$$\phi(u) = \frac{\sigma_b}{\sigma_a(u) + \sigma_b}. \tag{22}$$

The lethargy dependence of the flux of Eq. (22) is uniquely through $\sigma_a(u)$. Therefore, the lethargy dependence in the integrands of Eq. (1) is also uniquely through $\sigma_a(u)$. This is essential for the subgroup method. It allows the integration variable u to be replaced by σ_a and the integrals to be approximated by quadratures in σ_a according to the general formula

$$\frac{1}{\Delta u}\int_{\Delta u} f(u)\mathrm{d}u = \frac{1}{\Delta u}\int f(\sigma)\frac{\mathrm{d}u}{\mathrm{d}\sigma}\mathrm{d}\sigma \cong \sum_n w_n f_n. \tag{23}$$

In Eq. (23) $f_n = f(\sigma_n)$ are the integrands at the discrete values of σ_a. Each σ_n represents one of the subgroups into which the σ range has been subdivided. Figure 2 illustrates this; there $f(\sigma) = \sigma$ and the σ_n have been arbitrarily taken at the subgroup centers. The weights w_n are the sums of the interval fractions of Δu where the respective subgroups contribute to the integral. In Fig. 2, these intervals are marked

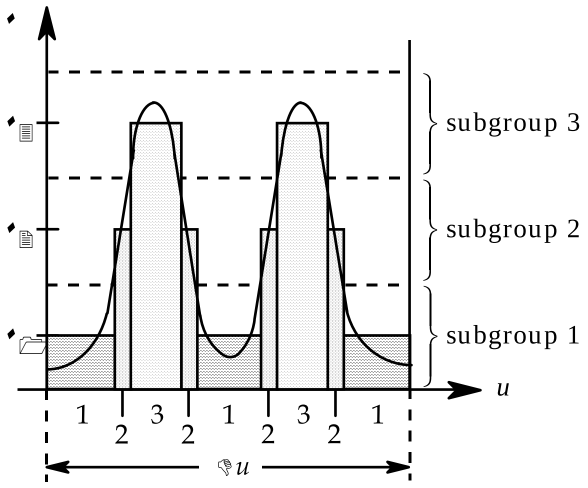


Fig. 2: Changing the integration variable from u to σ.

with the numbers of their contributing subgroups. This interpretation of the weights leads to two obvious conditions, to wit:

$$\sum_n w_n = 1 \quad \text{and} \quad 0 \le w_n \le 1 . \tag{24}$$

As a consequence of the above, both the lethargy integrals in Eq. (1) can be approximated by quadratures based on the same set of σ_n and w_n, so that

$$\sigma_a = \frac{\sum_n w_n \sigma_n \phi_n}{\sum_n w_n \phi_n} . \tag{25}$$

Once a set of σ_n and w_n is known, it is sufficient to evaluate the fluxes ϕ_n corresponding to cases in which the resonance isotope in the system has the absorption XS σ_n. Inserting these fluxes in Eq. (25) then yields the group-averaged absorption XS. This method offers no advantage in homogeneous systems where σ_a can be directly obtained from Eq. (20). In heterogeneous systems, however, it allows an evaluation of the group-averaged absorption XSs in all regions by calculating the ϕ_n. Thus, in contrast to traditional resonance methods, the subgroup method is not limited to resonance absorption in pins in more-or-less regular lattices. The concept of an average pin can be abandoned and resonance absorption can be evaluated in all regions of the system.

Unfortunately, the situation is not all that simple. Due to resonance scattering and because there can be more than one resonance isotope, each with its own (σ_n,w_n) set, lethargy dependence is not only through $\sigma_a(u)$. Also, it does not say how to deal with resonance fission. Resonance interaction is discussed in section 5, the other points in section 3.3. But first it will be shown how (σ_n,w_n) sets are obtained from RIs.

III. 3.2 Subgroup cross-sections and weights

In principle, the choice of the σ_n is arbitrary, provided their number N is sufficient for an accurate quadrature. In fact, by a judicious choice of the σ_n, N can be limited to values as small as 2 to 4 (Khairallah and Recolin, 1972). It is assumed that a good set of σ_n values is known—see below. The corresponding w_n are then obtained from a least-squares fit to a large number $K>N$ of absorption cross sections σ_{ak}, obtained from RIs $R_k \equiv \mathrm{RI}(\sigma_{bk})/\Delta u$ according to Eq. (20). The R_k are evaluated from exact library RIs by Segev's interpolation. Further, the side condition that the w_n yield the true R_∞ is imposed. This condition is treated by the method of Lagrange multipliers.

In the present case, using Eqs. (22) and (23), the desired fit is written as

$$\sigma_{ak} = \frac{\sum\limits_n w_{an}\sigma_{an}\phi_{nk}}{1-\sum\limits_n w_{an}\sigma_{an}\phi_{nk}/\sigma_{bk}} = \frac{\sum\limits_n w_{an}\sigma_{an}\phi_{nk}}{1-\sum\limits_n w_{an}\left(1-\phi_{nk}\right)}, \tag{26}$$

which leads to

$$\sum_n w_{an}\sigma_{an}\left[\frac{1}{\sigma_{an}}+\phi_{nk}\left(\frac{1}{\sigma_{ak}}-\frac{1}{\sigma_{an}}\right)\right]=1\,. \tag{27}$$

Similarly, one finds from $\nu\sigma_f$, here written as σ_ν,

$$\sum_n w_{\nu n}\sigma_{\nu n}\frac{\phi_{nk}/\sigma_{\nu k}}{1-\sum\limits_n w_{an}\left(1-\phi_{nk}\right)}=1\,. \tag{28}$$

In general, Eqs. (27) and (28) can be written as

$$\sum_n a_{kn}w_n = 1, \text{ or } \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{K1} & \cdots & a_{KN} \end{pmatrix}\begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow \mathrm{A}\vec{w} = \vec{1}\,. \tag{29}$$

According to matrix theory (Dongarra *et al.*, 1979; Press *et al.*, 1990), the $K\times N$ matrix $\mathbf{A}$ of Eq. (iv) can be written as the product of three matrices: a $K\times N$ column-orthogonal matrix $\mathbf{U}$, an $N\times N$ diagonal matrix $\mathbf{D}$ and the transpose of an $N\times N$ orthogonal matrix $\mathbf{V}$. This is called singular value decomposition. The elements of $\mathbf{D}$ are positive or zero and do not increase in size, i.e., $d_1 \geq d_2 \ldots \geq d_N \geq 0$; they are the singular values of $\mathbf{A}$. Thus,

$$\mathrm{A}(K\times N) = \mathrm{U}(K\times N)\cdot\mathrm{diag}(d_1,d_2,\cdots,d_N)\cdot \mathrm{V}^T(N\times N)\,. \tag{30}$$

Because $\mathbf{U}$ is column-orthogonal and $\mathbf{V}$ is orthogonal, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ [$=\mathrm{diag}(1,1,...1)$] and $\mathbf{V}\mathbf{V}^T = \mathbf{I}$. This can be used to solve $\mathbf{A}\vec{w} = \vec{1}$ by left-multiplying it successively by $\mathbf{U}^T$, $\mathbf{D}^{-1}$ and $\mathbf{V}$, leading to

$$\vec{w} = \left[\mathrm{V} \cdot \mathrm{diag}(1/d_1, 1/d_2, \cdots, 1/d_N) \cdot \mathrm{U}^T \right] \vec{1} \,. \tag{31}$$

The equal sign in Eq. (31) is misleading because it is very likely that Eq. (29) is overdetermined (more equations than unknowns) and thus has no unique solution. However, in that case it can be shown that $\vec{w}$ of Eq. (31) is the closest "solution" in the least-squares sense. On the other hand, if Eq. (29) happens to be underdetermined or almost underdetermined (ill-conditioned), then one or more of the last singular values will be zero or almost zero. In that case, by setting the offending singular values and their inverses (!) to zero, $\vec{w}$ of Eq. (31) is again the solution in the least-squares sense (Press *et al.*, 1990).

So once a set of σ_n's is known, the weights are obtained as described above. What remains is the determination of the subgroup cross-sections. An initial set of N subgroup cross sections σ_n is assigned, having been determined by rough trial-and-error testing, to each resonance material. The following iterative process, which is done per isotope, is then performed in the service program HEBE to optimize the σ_n selection.

Starting from the initial σ_n values, the weights w_n are calculated as described above. With this (σ_n, w_n) set, RIs are evaluated for a large number $K>N$ of σ_b's. Their values are compared to RIs obtained by Segev's method, yielding a root-mean-square error *rmse*. Then σ_1 is modified upwards by a given fraction *fr* and new weights and a new *rmse* are found. If this is smaller than the old *rmse*, the new σ_1 value is kept. If not, the old σ_1 is modified downwards by *fr* and the same test is made. The same is done for the other subgroup cross sections, $\sigma_2, \ldots, \sigma_N$. This loop through the N σ_n values is repeated until $rmse < 0.05$ and its relative change is less than 0.001. If that is not the case after 50 iterations $fr \rightarrow fr/2$ and the process continues until convergence or until $fr < 0.02$.

III. 3.3 Resonance integral and flux

As mentioned earlier, the basic condition for the subgroup method—i.e., lethargy dependence is uniquely through the resonance behavior of $\sigma_a(u)$—is not always valid. In fact, the library RIs account for $1/v$ absorption and resonance scattering, which do not depend on $\sigma_a(u)$, nor do the ν·fission RIs. Therefore, the change of integration variable in the first member of Eq. (23) may not be correct. However, its second member, the quadrature, is a very accurate fit and it will now be shown that the same set of σ_n and w_n can be used to obtain σ_a from Eq. (1), i.e., both the RI in the numerator and the flux in the denominator can be obtained from the same set of σ_n and w_n.

Assume that a set of σ_n and w_n has been found for the RI. Then, using the flux of Eq. (22), the numerator of Eq. (1) divided by Δu can be written as

$$\frac{1}{\Delta u}\int_{\Delta u}\sigma_a(u)\phi(u)\mathrm{d}u \cong \sum_n w_n\sigma_n\frac{\sigma_b}{\sigma_n+\sigma_b} \ . \tag{32}$$

The flux (denominator), on the other hand, becomes

$$\frac{1}{\Delta u}\int_{\Delta u}\phi(u)\mathrm{d}u \cong \sum_n w_n\phi_n = \sum_n w_n\frac{\sigma_b}{\sigma_n+\sigma_b} = 1-\sum_n w_n\frac{\sigma_n}{\sigma_n+\sigma_b} \ , \tag{33}$$

where the last summation is R/σ_n, which can be evaluated with the same quadrature as Eq. (32). But the last member is only valid if

$$\sum_n w_n = 1 \ . \tag{34}$$

Eq. (34) is a condition that is not satisfied by the original set of σ_n and w_n. It can, however, be enforced by adding an (N+1)th member to this set according to

$$\sigma_{N+1} = 0 \quad \text{and} \quad w_{N+1} = 1-\sum_{n=1}^{N} w_n \ . \tag{35}$$

Because $\sigma_{N+1}=0$, it does not contribute to the RI, nor does it affect the accuracy of its quadrature, while it assures the same accuracy for the flux quadrature. Thus, both RI and group flux can be evaluated from the same, expanded quadrature set, even if the subgroup method is not valid.

III. 4 THE HETEROGENEOUS SYSTEM

In classical resonance theory, the RI in the region of a heterogeneous system is found from that in a homogeneous system by invoking an equivalence theorem. It states that the flux—and thus the RI—in a heterogeneous region is equal to that in a homogeneous system of the same material, though with its background XS augmented by a constant equivalence XS, Σ_e[3],

$$\Sigma_b = \lambda\Sigma_p \xrightarrow{\text{equivalence}} \Sigma_b = \lambda\Sigma_p + \Sigma_e \ . \tag{36}$$

Sometimes the heterogeneous system is equivalent to two homogeneous systems combined, each with its own Σ_e (Stamm'ler and Abbate, 1983). In any case, to interpolate in the RI tables, Σ_e must be constant. Then, from the interpolated RIs and Eqs. (20) and (21), the group XSs are obtained.

[3] In classical resonance theory, this is known as the *escape XS*. In the present method, however, the concepts of escape probability and rational approximation are not needed. Instead, the term *equivalence XS* is the preferred one.

III. 4.1 An enforced equivalence

In HELIOS, the equivalence of Eq. (36) is also used. However, now the Σ_e part of Σ_b is allowed to vary with lethargy through $\sigma_a(u)$. Thus, the equivalence of the heterogeneous system to a homogeneous one is enforced, and Eq. (22) for the flux becomes the definition of $\sigma_b(u)$,

$$\phi(u) = \frac{\sigma_b(u)}{\sigma_a(u)+\sigma_b(u)} \Rightarrow \sigma_b(u) = \frac{\sigma_a(u)\phi(u)}{1-\phi(u)} \ . \tag{37}$$

The σ_a dependence of σ_b is obtained from the second equation of (37), where the fluxes come from transport calculations (see section 4.2). Interpolation in these σ_b yields the σ_{bn}, which, when used in the first equation of (37), give the fluxes ϕ_n. Substituting these ϕ_n into Eq. (25) gives the effective absorption XS,

$$\sigma_a = \frac{\sum_n w_n \sigma_n \frac{\sigma_{bn}}{\sigma_n + \sigma_{bn}}}{\sum_n w_n \frac{\sigma_{bn}}{\sigma_n + \sigma_{bn}}} \ . \tag{38}$$

The ν-fission XS is also evaluated from the Eq. (38), now with the factor $w_{\nu n}\sigma_{\nu n}$ in the numerator, instead of $w_n\sigma_n$. The $w_{\nu n}$ are obtained as described in section 3.2, replacing R by R_ν and σ_n by $\sigma_{\nu n}$. For $\sigma_{\nu n}$, the following expression is used,

$$\sigma_{\nu n} = \sigma_n \frac{R_{\nu\infty}}{R_\infty} \ . \tag{39}$$

Instead of directly using the fluxes ϕ_n of the heterogeneous transport calculations, the detour via σ_{bn} was chosen. First, owing to the weak dependence of σ_b on σ_a (traditionally, Σ_b is constant) the number of σ_b's to describe this dependence is smaller than the number of subgroups, requiring fewer heterogeneous calculations. Second, if there are more resonance isotopes, each has its own σ_n set, unless a superset with many σ_n's is chosen. In both cases the number of heterogeneous calculations becomes large. Third, as will be seen in section 5, σ_{bn} rather than ϕ_n is needed to deal with interacting resonance isotopes.

III. 4.2 Evaluation of the equivalence cross-section

To determine the variation of Σ_e with σ_a, M transport calculations are done to obtain fluxes at M values of $\sigma_a = \sigma_m$.[4] These are fixed-source calculations in which,

[4] These σ_m are selected such that they can be used also to determine the flux needed to correct the removal XS; see section 6. Currently, M has been set to 4.

according to the IR approximation and assumption 3 of section 2, the source in each region is $\lambda\Sigma_p$. Recalling that so far only one resonance isotope with number density N is considered (for interaction see section 5), the following input must be given per region:

$$\left.\begin{array}{lll} Q = \lambda\Sigma_p & - & \text{source} \\ \Sigma_a = N\sigma_m & - & \text{absorption XS} \\ \Sigma_r = \Sigma_a + \lambda\Sigma_p & - & \text{removal XS} \\ \Sigma = \Sigma_a + \Sigma_p & - & \text{total XS} \end{array}\right\} . \tag{40}$$

All the above data are based on number densities and microscopic library XSs, the so-called "smooth" XSs. The exception is σ_m, which does not come from the library and which is only non-zero for resonance absorbers. Further, if albedo boundary conditions are used, all albedo matrices are assumed to be identity matrices, i.e., diagonal with all elements equal to 1.

Because $\lambda\sigma_p$ is lethargy-dependent—see section 2.2.2—the heterogeneous calculations should be done per resonance group. Alternatively, these calculations can be done for the entire resonance range as a whole. This requires group-averaged $\lambda\Sigma_p$ and Σ_p values. These averages are obtained by weighting them with infinitely diluted RIs. Here, in anticipation of section 5.1, the weighting formula is given for the resonance category k,

$$\Sigma_x = \frac{\sum_{i\in k}\sum_g \Sigma_{xg} I_{ig\infty}\Delta u_g}{\sum_{i\in k}\sum_g I_{ig\infty}\Delta u_g} \quad \text{with} \quad I_i = N_i R_i . \tag{41}$$

Eq. (41) is based on the consideration that the groups with the largest RIs should contribute most. This formula is also used in regions that do not contain isotopes of category k. In those regions, the sums over the isotopes disappear, and the macroscopic I_{ig}'s are replaced by the microscopic R_{rg} of the category's representative isotope introduced in section 5.1.

The result of the transport calculations is, for each region, a set of M fluxes, ϕ_m, each associated with a σ_m. Substitution of ϕ_m into Eq. (37), and subsequent use of Eq. (36), then yields the associated Σ_{em}. This establishes the σ_a dependence of Σ_e, in the form of tables of $\Sigma_e(\sigma_m)$. These tables are used, inter alia, to obtain by linear interpolation in $\ln(\sigma_m)$ the σ_{bn} of Eq. (38).

To avoid numerical problems, Eq. (37) is not used if the flux is greater than 0.999. This happens for small σ_n values, in which case the corresponding Σ_{en} is obtained by linear extrapolation in $\ln(\sigma_m)$. For $\sigma_{N+1}=0$, the average of the linearly-extrapolated value at 0 and σ_1 is used. At the other end, extrapolation for large σ_n's may produce negative Σ_{en}'s. This is avoided by enforcing a minimum value of 10^{-4} cm^{-1}.

III. 5 THE INTERACTION OF RESONANCE ISOTOPES

The object of the heterogeneous transport calculations is to obtain Σ_e at M values of the microscopic resonance absorption XS, σ_m. If there is just one resonance isotope, the input to the heterogeneous calculations is readily set up according to Eqs. (40-41). However, evaluating Σ_a becomes problematic if there are various resonance isotopes. The problem lies in the fact that Σ_a depends on both σ_m and the number densities of the resonance isotopes.

III. 5.1 Resonance sets and resonance categories

In HELIOS, this problem is dealt with in its two extreme limits: no interaction and full interaction. In the former, the resonances of the different isotopes do not overlap at all. In the latter, the isotopes have overlapping, but not necessarily equally strong, resonances; such isotopes form a resonance <u>category</u>. A combination of categories is called a resonance <u>set</u>. Of course, each set contains all the resonance isotopes exactly once. Table 1 lists the currently available nine resonance sets and their categories.

Table 1. Sets of resonance categories in HELIOS.

Set	Cat	Description
1	1	All isotopes except natl Zr
	2	Natural Zr
	3	Zr isotopes
2	1	Heavy-metal, fiss products
	2	Natural Zr
	3	Zr isotopes
	4	Other isotopes
3	1	HM isotopes, FPs
	2	Natural Zr
	3	Zr isotopes
	4	In-115, Hf, rest of Ag
	5	Ag-109, other isotopes
4	1	U-238
	2	Other HM isotopes, FPs
	3	Natural Zr
	4	Zr isotopes
	5	Other isotopes
5	1	U-238
	2	Other HM isotopes, FPs
	3	Natural Zr
	4	Zr isotopes
	5	In-115, Hf, rest of Ag
	6	Ag-109, other isotopes
6	1	Th-232
	2	U-238
	3	Other HM isotopes, FPs
	4	Natural Zr
	5	Zr isotopes
	6	Other isotopes
7	1	Th-232
	2	U-238
	3	Other HM isotopes, FPs
	4	Natural Zr
	5	Zr isotopes
	5	In-115, Hf, rest of Ag
	6	Ag-109, other isotopes
8	1	Th-232
	2	U-233
	3	U-235
	4	U-238
	5	Other HM isotopes, FPs
	6	Natural Zr
	7	Zr isotopes
	8	Hf-177, Gd, Er
	9	In-115, other Hf, other Ag
	10	Ag-109, other isotopes
9	1-*n*	All isotopes individually, but all FPs together, and Pu-238, Pu-242 and Am-241 together

The mention of an isotope in Table 1 only means that it will be treated as a resonance isotope if it has resonance data in the nuclear-data library.

Further, the entire resonance lethargy range is subdivided into nine intervals, and a different set can be assigned to each interval. These sets per lethargy interval are converted into sets per library group, where each group gets the set that corresponds to the interval that contributes most to the group. The interval boundaries are shown below in eV, not lethargy.

```
<----1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9---->
        1000       399       150      49.8      27.9      15.9      9.96      4.01
```

The *M* transport calculations are done for all the relevant categories, i.e., the categories of each resonance-group set or the categories of the set for the entire resonance range. In these calculations, absorption by the isotopes of the other categories is neglected, which corresponds to the limit of no interaction between isotopes from different categories. On the other hand, the isotopes *i* of category *k* contribute to Σ_a in proportion to their infinitely diluted macroscopic RIs, $I_{i\infty} = N_i R_{i\infty}$.

Each category has been assigned a representative isotope, *r*, which should be the most dominant resonance absorber in the category. It is the σ_m values of *r* that are used to obtain Σ_a in Eq. (40), i.e., Σ_a is found from

$$\Sigma_a(\sigma_m) = \left(\frac{I_\infty}{R_{r\infty}}\right)\sigma_m \quad \text{or} \quad \Sigma_a(\sigma_m) = \left(\frac{\sum_g I_{g\infty}\Delta u_g}{\sum_g R_{rg\infty}\Delta u_g}\right)\sigma_m , \tag{42}$$

where

$$I_\infty = \sum_i N_i R_{i\infty} \quad \text{or} \quad I_{g\infty} = \sum_i N_i R_{ig\infty} . \tag{43}$$

If the categories are per resonance group, then the first expression of Eq. (42) applies. The second expression applies if the categories are for the entire resonance range.

Since HELIOS-1.10, a refinement suggested by Ferri (2005) has been introduced for the case of a temperature variation inside a coherent set of regions with the same resonance material. An example is the temperature profile in a fuel pin, where the temperature increases towards the centre. Because of Doppler broadening, the cross-section of the material is not constant throughout the pin. This effect is accounted for in an approximate manner by adjusting the contributions of the category's isotopes to $\Sigma_e(\sigma_m)$. Let $\overline{T}$ be the average temperature of the coherent regions and *T* the temperature of the actual region. Then a given isotope's contribution to $\Sigma_e(\sigma_m)$ in Eq. (42) is varied by the factor $f(T)$, given by

$$f(T) = \frac{\hat{\sigma}_a(T,\lambda\sigma_p)}{\hat{\sigma}_a(\overline{T},\lambda\sigma_p)} = \frac{R(T,\lambda\sigma_p)\,\big(\lambda\sigma_p - R(\overline{T},\lambda\sigma_p)\big)}{R(\overline{T},\lambda\sigma_p)\,\big(\lambda\sigma_p - R(T,\lambda\sigma_p)\big)} . \tag{42a}$$

This adjustment multiplies $R_{i\infty}$ and $R_{ig\infty}$ by $f_i(T)$ and $\bar{f}_i(T)$, which transforms the two members of Eq. (43) as follows:

$$I_\infty \to \sum_i N_i R_{i\infty}(\bar{T}) f_i(T) \quad \text{or} \quad I_{g_\infty} \to \frac{\sum_i N_i \sum_g R_{ig}(\bar{T}) f_i(T)}{\sum_i N_i \sum_g R_{ig}(\bar{T})} \sum_i N_i R_{ig\infty}(\bar{T}) \,. \tag{43a}$$

Now, the calculated tables of $\Sigma_e(\sigma_m)$ of section 4.2 can be used to obtain, by interpolation in σ_m, the background XS, σ_{bni}, of resonance isotope i corresponding to its subgroup absorption XS, σ_{ni}. This requires the interpolation argument σ^*,

$$\sigma^* = \left(\frac{R_{r\infty}}{R_{i\infty}} \right) \sigma_{ni} \quad \text{or} \quad \sigma^* = \left(\frac{\sum_g R_{rg\infty} \Delta u_g}{\sum_g R_{ig\infty} \Delta u_g} \right) \sigma_{ni} \;, \tag{44}$$

which then is used to obtain σ_{bni} from

$$\sigma_{bni} = \frac{\lambda \Sigma_p + \Sigma_e(\sigma^*)}{N_i} \;. \tag{45}$$

In the common situation of isotopes with non-overlapping resonances, of which one is predominant (say, U-238) and the others occur in small amounts, one single category seems unsatisfactory. That is because the σ^*'s of the other isotopes interpolate in tables of $\Sigma_e(\sigma_m)$ that were generated with far too large Σ_a's. This is mitigated because the RIs vary little at the high dilutions of the other isotopes, nor are their contributions that important. If the calculations are per resonance group, a possible choice is to put U-238 in a category of its own in the (lowest) group where its most important resonance lies, otherwise using only one category. If the entire resonance range is treated at once, and if the too strong shielding of the other isotopes by U-238 is unacceptable for Σ_e, then U-238 should be put in a category of its own.

If two strong non-overlapping isotopes don't share the same regions, they should be put in different categories. This yields more reliable Σ_e's for a checkerboard lattice of pins that contain one isotope or the other. Yet, the errors incurred by not separating them in categories may be mitigated by the weak variation of Σ_e with σ_m.

If it is decided to have a separate group for U-238, the other heavy isotopes may share a category with isotopes with which they will never occur in the same regions, like Hf, Ag and In. Because of their low concentrations, the heavy isotopes affect the others only marginally. The others isotopes, on the other hand, occur in control materials, and are either separated by a gap from the fuel pins or they affect only a few neighbouring fuel pins.

III. 5.2 Group-averaged absorption cross-sections

If there are more resonance isotopes, indexed *i* and *j*, Eqs. (25) and (22) can be applied to the mixture of these isotopes. Introducing macroscopic XSs, an expression for Σ_a of the mixture can be derived. Breaking this expression up into its isotopic components and returning to microscopic XSs leads to the equivalent of Eq. (25) for an arbitrary isotope *i*:

$$\sigma_{ai} = \frac{\displaystyle\int \frac{\sigma_{ai}(u)\sigma_{bi}(u)}{\sigma_{xi} + \sigma_{ai}(u) + \sigma_{bi}(u)} \mathrm{d}u}{\displaystyle\int \frac{\sigma_{bi}(u)}{\sigma_{xi} + \sigma_{ai}(u) + \sigma_{bi}(u)} \mathrm{d}u} , \tag{46}$$

where

$$\sigma_{bi}(u) = \frac{1}{N_i} \Sigma_{bk(i)}(u) \quad \text{and} \quad \sigma_{xi}(u) = \frac{1}{N_i} \sum_{j \neq i} N_j \sigma_{aj}(u) . \tag{47}$$

In Eqs. (46-47), $\Sigma_{bk(i)}(u)$ is the background XS of the category *k* that contains isotope *i*. Further, $\sigma_{xi}(u)$ represents the absorption contributions of the other isotopes. Assuming the resonances to be randomly distributed in Δu, $\sigma_{xi}(u)$ can be replaced by its group-averaged value, $\sigma_{xi}(u) \rightarrow \sigma_{xi}$ (Askew, Fayers and Kemshell, 1966). Introducing this approximation into Eq. (46) and replacing the integrals by quadratures, the equivalent of Eq. (38) is obtained for isotope *i* in a mixture of resonance (and other) isotopes,

$$\sigma_{ai} = \frac{\displaystyle\sum_n w_{ni} \frac{\sigma_{ni}\sigma_{bni}}{\sigma_{ni} + \sigma_{xi} + \sigma_{bni}}}{\displaystyle\sum_n w_{ni} \frac{\sigma_{bni}}{\sigma_{ni} + \sigma_{xi} + \sigma_{bni}}} . \tag{48}$$

Apart from the assumption that $\sigma_{xi}(u) \rightarrow \sigma_{xi}$, the passage from Eq. (46) to Eq. (48) contains another approximation. In section 3.3, it was concluded that in the case of only one resonance isotope the same quadrature can be used for RI and group flux. But with more resonance isotopes, Eq. (22) for the flux has an extra term $\sigma_x(u)$ in the denominator. This passage is still rigorous for constant $\sigma_b(u)$ because, taking the factor $\sigma_b/(\sigma_x + \sigma_b)$ out of the integrals restores fluxes of the type of Eq. (22) in the integrands. It also is rigorous if the subgroup condition holds that the lethargy dependence of $\sigma_b(u)$ is uniquely through $\sigma_a(u)$.

The violation of the above conditions is not very serious. First, $\sigma_b(u)$ may be assumed to vary slowly with lethargy; it is constant in the classical theory. Second, the subgroup condition is fulfilled for the representative isotopes of each category; they are usually the predominant isotopes. Third, the approximations cause only errors in the interaction correction to σ_{ai}; in the zeroth order approximation, σ_{xi} is zero.

Because the σ_{xj} depend on σ_{ai}, Eq. (48) implies an iteration process. With σ_{ni} and σ_{bni} known, the iterations start with $\sigma_{xj} = 0$ and proceed as follows:

$$\sigma_{xi}^{(0)} \xrightarrow{\text{Eq.(48)}} \sigma_{ai}^{(0)} \xrightarrow{\text{Eq.(47)}} \sigma_{xi}^{(1)} \xrightarrow{\text{Eq.(48)}} \sigma_{ai}^{(1)} \xrightarrow{\text{etc.}} \cdots \, . \tag{49}$$

The convergence criterion of this process requires that the resonance-shielded absorption XSs of all the resonance isotopes in the mixture shall not differ by more than a fraction of 0.001 from their values in the previous iteration. A relaxation factor of 0.5 is used, while at most 30 iterations are made.

III. 5.3 Group-averaged scattering cross-sections

Until HELIOS-1.7, the scattering cross sections of a resonance isotope and its down-scattering elements of the P_0 (and P_1) matrix were taken directly from the library. In the library, these data, like the absorption and fission cross sections, are shielded at a fixed dilution of the isotope in question. Because resonance scattering is not very important, it is not a bad approximation to use a fixed "typical" shielding. However, the nuclear data-base contains tables of resonance shielded scattering and down-scattering cross sections at different dilutions. So it was decided to add them to the library and try to use them to get at least shielded scattering data that vary with dilution. Since HELIOS-1.8, the following simple model for resonance scattering and down-scattering has been implemented.

Once the final σ_x of a resonance isotope has been found from the process given by Eq. (49), its corresponding $R(\sigma_b) = \mathrm{RI}(\sigma_b)/\Delta u$ is also known — the numerator of Eq. (48) divided by Δu. This in turn yields σ_b, which is found from the library tables of R. First, its nearest larger and smaller R values are found, $R_j(\sigma_{bj}) \leq R(\sigma_b) \leq R_{j+1}(\sigma_{bj+1})$; thereafter σ_b is obtained by repeated halving in the interval $<\sigma_{bj}, \sigma_{bj+1}>$. Effective scattering and down-scattering cross sections are then obtained by interpolating with this σ_b in library tables of resonance-shielded scattering and down-scattering cross sections (not RIs!). It is not these effective cross sections that are used, but rather their ratios to the above-mentioned library cross section with fixed "typical" shielding.

III. 6 THE REMOVAL CROSS-SECTION

The effect of resonance absorption on reactivity is a matter of competition between absorption and slowing down or removal. In this section the effect on the removal XS is considered. This XS is the sum of the out-scattering XSs[5], which in the library have been weighted, according to Eq. (1), with the asymptotic flux, $\phi(u) = 1$. Resonance scattering is unimportant in this context, since it is rather the lighter, non-resonance isotopes that contribute to removal. Although these isotopes have a virtually constant

[5] In deviation from standard HELIOS usage, where removal is out-scattering plus absorption.

scattering XS, their slowing down behaves exponentially with lethargy; see Eq. (3). Moreover, the down-scattering of an isotope *i* acts only in a lethargy band Δ_i, which may be smaller than the lethargy-group width Δu. Thus, the upper ranges of the groups contribute most to removal. There, $\phi(u) \neq 1$ because it is reduced by the resonance escape probability *p* of the group, nor is $\phi(u)$ equal to the group-averaged flux, which is smaller than *p* due to the resonance dip(s).

In HELIOS, correction factors f_r are evaluated to account for the variation of $\phi(u)$ on the removal XS. Because the scattering matrices in the library are based on a constant epithermal flux, f_r is the ratio of the removal XS averaged with the true flux to the removal XS averaged with $\phi(u)$=1. Clearly, f_r depends only on the out-scattering group. For a mixture of isotopes *i* one has

$$f_r = \frac{\sum_i N_i \bar{\sigma}_{ri}}{\sum_i N_i \sigma_{ri}} \ . \tag{50}$$

Using the slowing-down kernel of Eq. (3) and dropping the isotope index *i*, the average removal XS of an arbitrary isotope according to Eq. (1) becomes

$$\left.\begin{aligned} \bar{\sigma}_r &= \frac{\sigma_s \int_{u_0}^{\Delta u} p(u)\phi(u)\left[\int_{\Delta u}^{u+\Delta} \exp(u-u')\mathrm{d}u'\right]\mathrm{d}u}{(1-\alpha)\int_0^{\Delta u} p(u)\phi(u)\mathrm{d}u} = \\ &= \frac{\sigma_s \int_{u_0}^{\Delta u} p(u)\phi(u)\left[\exp(u-\Delta u)-\alpha\right]\mathrm{d}u}{(1-\alpha)\int_0^{\Delta u} p(u)\phi(u)\mathrm{d}u} \end{aligned}\right\} \ . \tag{51}$$

In Eq. (51), which for $\phi(u)=1$ gives σ_r, the lower group limit was arbitrarily set to zero, so the upper limit is Δu. In the numerator, the integration limits represent the fact that the maximum lethargy gain per collision is Δ. Hence, out-scattering at *u* can only be into the range from Δu to u+Δ. On the other hand, if $\Delta < \Delta u$, only the upper range from Δu to Δ contributes, so that the out-scattering interval in general extends from u_0 to Δu, where

$$u_0 = \max(0, \Delta u - \Delta) \ . \tag{52}$$

In Eq. (51) the resonance escape probability $p(u)$ is included because it does not cancel as it did when evaluating the average absorption XS in section 2.3. To derive an expression for $p(u)$, two different assumptions are made regarding the location of the resonances in the groups. Below a given cutoff, u_c, the resonances are assumed to be uniformly distributed; above u_c, each group has only one centrally located narrow resonance (Fayers, Kemshell and Terry, 1967). The group containing u_c gets a weighted correction factor according to

$$f_r = \frac{\Delta_c}{\Delta u} f_r^- + \left(1 - \frac{\Delta_c}{\Delta u}\right) f_r^+ \ , \tag{53}$$

where Δ_c is the lethargy interval below u_c, f_r^- and f_r^+ are the correction factors below and above u_c, and (currently) $u_c = 100$ eV.

Before deriving expressions for $p(u)$ based on these two assumptions, the group resonance escape probability p of Eq. (18) will be expressed in terms of the group-flux dip, $1-\phi$. First, it should be noticed that passing from a homogeneous to a heterogeneous system increases the background XS by the factor $(\lambda\Sigma_p+\Sigma_e)/\lambda\Sigma_p$, so the $\xi\Sigma_s$ will be multiplied by the same factor. Then, from Eq. (18) and the interpretation of the denominator of Eq. (20) as the group flux ϕ (≤ 1), it follows for p (the evaluation of p is discussed in section 6.1) that

$$p = \exp(-a\Delta u) \quad \text{with} \quad a = \frac{\lambda\Sigma_p}{\xi\Sigma_s}(1-\phi) \ . \tag{54}$$

Remembering that the lethargy group is taken from 0 to Δu, the resonance escape probability $p(u)$ and its lethargy-averaged value according to the two assumptions about the locations of the resonances are found to be:

$$\left\{\begin{array}{lll} \text{uniform}: & p(u) = \exp(-au) & \Rightarrow \quad \tilde{p} = \dfrac{1-p}{a\Delta u} \\ \text{central}: & \left\{\begin{array}{l} p(u) = 1; \quad u < \frac{1}{2}\Delta u \\ p(u) = p; \quad u > \frac{1}{2}\Delta u \end{array}\right\} & \Rightarrow \quad \tilde{p} = \dfrac{1+p}{2} \end{array}\right\} . \tag{55}$$

With $p(u)$ known, the integrals in the numerator and denominator of Eq. (51) can be evaluated. In both assumptions on the resonance distribution, the flux in the denominator is replaced by its average value, $\phi(u) \to \phi$. Thus, the denominator becomes $\tilde{p}\,\phi\Delta u$. The integral in the numerator, however, differs for the two assumptions. For a <u>uniform resonance distribution</u>, the replacement $\phi(u) \to \phi$ may be made, whereupon the integration can be performed to yield

$$\frac{1-\exp[-(1-a)(\Delta u - u_0)]}{1-a} + \alpha \frac{1-\exp[a(\Delta u - u_0)]}{a} \ . \tag{56}$$

For a <u>centrally located narrow resonance</u>, the flux of Eq. (22) is written as

$$\left[\phi(u) = 1 - \frac{\Sigma_a(u)}{\Sigma_b(u)+\Sigma_a(u)}\right] . \tag{57}$$

Substituting this flux into the integrand of the numerator of Eq. (51) yields two integrals. The integration range of the first integral is split into two intervals: before and after the resonance at $\Delta u/2$. If $u_0 > \Delta u/2$, the first interval is zero, which is formulated by introducing u_1 such that the intervals are from u_0 to u_1 and from u_1 to Δu, i.e.

$$u_1 = \max(\tfrac{1}{2}\Delta u, u_0) \ . \tag{58}$$

The first integral is readily evaluated in the two intervals. To integrate the second integral, it is observed that its integrand acts in a narrow band centred at $\Delta u/2$, at least if the integration range includes the resonance. Hence, $p(u)$ can be replaced by its average value $\tilde{p}$, and $[\exp(u\text{-}\Delta u)-\alpha]$ by $[\exp(u_1\text{-}\Delta u)-\alpha]$. Further, the remaining part of the integrand is just $[1-\phi(u)]$, which also can be replaced by its average value $1-\phi$. The resulting expression for the flux-averaged removal XS becomes

$$\bar{\sigma}_r = \sigma_s \left(\frac{\xi^A - (1-p)\xi^B - \tfrac{1}{2}(1+p)(1-\phi)\xi^C}{\tfrac{1}{2}(1+p)\phi\Delta u} \right) , \tag{59}$$

where

$$\left.\begin{aligned} \xi^A &= \frac{1-\alpha(\Delta u - u_0) - \exp[-(\Delta u - u_0)}{1-\alpha} \\ \xi^B &= \frac{1-\alpha(\Delta u - u_1) - \exp[-(\Delta u - u_1)}{1-\alpha} \\ \xi^C &= \Delta u \frac{\exp[-(\Delta u - u_1)] - \alpha}{1-\alpha} \end{aligned}\right\} . \tag{60}$$

In the unperturbed case of <u>no resonances</u>, $p(u)=1$ and $\phi(u)=1$. Further, if the resonance is excluded from the integration, then $u_0=u_1$, so that $\xi^A=\xi^B$ and $\xi^C=0$. In that case, σ_r of the denominator of Eq. (50) is obtained from Eq. (51)—or from Eq. (59)—as

$$\sigma_r = \sigma_s \frac{\xi^A}{\Delta u} \ . \tag{61}$$

Substitution of Eq. (61) and Eq. (56) or Eq. (59) into Eq. (50) yields, finally, f_r. Removal correction factors, f_r, are evaluated for all the scattering matrices in all the resonance groups. According to section II.5.1, the same material can occur in different regions and different materials can share the same scattering matrix. So, in Eq. (50), f_r is found as the ratio of flux-volume-integrated corrected to uncorrected removal XSs.

Since HELIOS-1.8, resonance isotopes are excluded from the above removal-correction factor, because the resonance-scattering treatment discussed in section 5.3 takes care of it. The correction is only calculated for the mixture of non-resonance isotopes and applied to these isotopes.

While the out-scattering XSs of the non-resonance isotopes are corrected with f_r, their total scattering XSs do not change. This implies another correction factor for the self-scattering XSs. Since self-scattering XSs are not used in the collision probability calculations, this other correction factor is never evaluated directly; rather, for the MoC calculations, the self-scattering XSs are computed by subtracting the corrected out-scattering XSs from the total scattering XSs.

III. 6.1 The resonance escape probability

The only remaining problem is the evaluation of the group resonance escape probability p to be used in Eqs. (56) and (59). This p, which must be evaluated for each region i, is a global quantity, i.e., it depends on contributions from the entire system. On the one hand, the difference between sources and slowing down in region i, $\lambda\Sigma_{pi}(1-\phi_i)$ according to Eq. (40), represents absorption losses anywhere in the system. On the other hand, the neutrons that slow down in region i, $\lambda\Sigma_{pi}\phi_i$, can have originated anywhere in the system. The following recipe is used to obtain the resonance escape probability of region i, p_i.

With Eq. (54) in mind, a system-averaged flux and resonance escape probability, ϕ_{av} and p_{av}, are defined as follows (the sums are over all the regions of the system):

$$\left.\begin{aligned} \phi_{av} &= \frac{\sum_i \xi\Sigma_{si}\phi_i V_i}{\sum_i \xi\Sigma_{si} V_i} \\ p_{av} &= \exp\left(-\frac{\sum_i \lambda\Sigma_{pi}(1-\phi_i)V_i}{\sum_i \xi\Sigma_{si} V} \Delta u \right) \end{aligned}\right\} . \tag{62}$$

In a region with resonance absorption, ϕ_i is the converged denominator of Eq. (48). In regions without resonance isotopes, ϕ_i is evaluated from the sum of the flux dips belonging to the different categories k. The dips are evaluated from the fluxes ϕ_{mk} by subgroup quadratures. That is why the transport calculations in each category k are done at σ_{mk}'s that correspond to a quadrature set of its representative isotope. In view of the simple assumptions, and to keep the number of heterogeneous calculations down, that is also why low-order quadratures are used ($M=4$). To avoid negative fluxes, an exponential expression is used to evaluate ϕ in each region from the sum of the flux dips,

$$\phi = \exp\left[-\sum_k \sum_m w_{mk}(1-\phi_{mk}) \right] . \tag{63}$$

Utilizing ϕ_{av} and p_{av} of Eq. (62), the two following relations ensure that if $\phi_i \le \phi_{av}$, then $p_i \le p_{av}$, and if $\phi_i > \phi_{av}$, then $p_i > p_{av}$, at the same time keeping p_i within its limits, $0 \le p_i \le 1$:

$$\left.\begin{array}{ll} p_i = \dfrac{p_{av}}{\phi_{av}}\phi_i & \text{if } \phi_i \le \phi_{av} \\ 1-p_i = \dfrac{1-p_{av}}{1-\phi_{av}}(1-\phi_i) & \text{if } \phi_i > \phi_{av} \end{array}\right\} . \qquad (64)$$

The assumptions underlying the above removal correction may seem to be crude. It should be recognized that slowing down mainly occurs in the coolant regions, where the flux is already near its asymptotic value and the correction is small.

III. 7 SOME REMARKS ON THE PROGRAM

The calculation of effective microscopic resonance XSs and removal correction factors is done in LINK13. This module is called twice per calculational point: first to prepare input for the transport calculations, and next to evaluate resonance XSs and correction factors.

Before HELIOS-1.4 (mid-1996) the subgroup data were evaluated in LINK01 from library resonance integrals. Since then, these data are read from the library (they are evaluated by the library-service program HEBE). Since HELIOS-1.10, the subroutines that deal with the evaluation of subgroup constants—LIBRES, RESINT, SUBSIG and RESFIT—have been eliminated.

III. 7.1 Notes on LINK13

The preparation of XSs and sources, according to Eq. (40), is done during the first call of LINK13. The following eight subroutines are involved:

AUXIAR:	It determines which of the resonance categories are active. The number of transport calculations to be executed is M times the number of active categories, times the number of resonance groups—or 1 if the entire resonance range is treated at once (default).
CALTAV:	It obtains the average temperature of coherent regions with the same material.
TINCOF:	It presets indexes and coefficients for temperature interpolation in the library tables to obtain the weights w_n at the actual material temperatures. This is done both for the actual region temperatures and for the average temperatures of coherent regions with the same material. It also evaluates by temperature interpolation the w_m values at the average temperature of all the resonance materials. They are used when obtaining fluxes in regions without resonance isotopes; see section 6.1.
RESADR:	It sets albedo data—resonance albedo matrices are identity matrices.
SMOOTH:	It evaluates the "smooth" macroscopic XSs needed in the transport calculations and in the calculations of the effective XSs.

RESSIG:	It evaluates XSs and sources for the transport calculations, calling SIGMA3 or SIGMA4, as appropriate.
SIGMA3:	It evaluates XSs and sources if the entire resonance range is to be treated at once.
SIGMA4:	It evaluates XSs and sources for each resonance group, i.e., the resonance range is treated group by group.

After the transport calculations are done, LINK13 is called a second time. From the calculated fluxes, it evaluates the effective XSs and the removal-correction factors. The following subroutines are involved:

TINCOF:	See above
POSFLU:	It evaluates the final resonance data by calling the remaining seven subroutines below.
EQUIXS:	Using Eqs. (37) and (36), it establishes the σ_a dependence of Σ_e for all regions, i.e. tables of $\Sigma_e(\sigma_m)$, where $m = 1, \ldots M$.
WINTER:	It evaluates the temperature-interpolated weights, w_n, required in RESOXA. This is done for all the resonance isotopes in a given region.
BACKXS:	It evaluates σ_{bni} of all the resonance isotopes i in a given region, using Eq. (47) and interpolation in $\ln(\sigma_m)$.
RESOXA:	It evaluates effective resonance absorption XSs for all the resonance isotopes in a given region according to the iteration process outlined in section 5.2. It also finds, per isotope and per region, mutual shielding factors.
RESOXS:	It evaluates effective scattering and down-scattering XSs for all the resonance isotopes in a given region, as described in section 5.3.
RINPOL:	It is called from RESOXS and interpolates in the library to get resonance integrals (absorption or ν·fission) or cross sections (scattering or down-scattering) at one of their tabulated background cross sections.
RESCAP:	It evaluates p_i in all the regions i, as described in section 6.1.
REMFAC:	It evaluates the removal-correction factors for all the materials, as described in section 6.

REFERENCES

ALDOUS A C (1969) Numerical studies of the hydrogen equivalent of some structural materials in their effect on U-238 resonance capture, *UKAEA Report AEEW-M 860*, Winfrith, England.

ASKEW J R, FAYERS F J and KEMSHELL P B (1966) A general description of the lattice code WIMS, *J. British nucl. Energy Soc.* **5**, 564-585.

DONGARRA J J, MOLER C B, BUNCH J R and STEWART G W (1979) *LINPACK, Users' Guide*, SIAM (Soc. Ind. Appl. Math.), Philadelphia.

FAYERS F J, KEMSHELL P B and TERRY M J (1967) An evaluation of some uncertainties in the comparison between theory and experiment for regular light water lattices, *J. British nucl. Energy Soc.* **6**, 161-181.

FERRI A A (2005) HELIOS: Resonance Calculations in Non-uniform Temperature Distributions, *Private Note*, Necochea, Argentina.

GOLDSTEIN R and COHEN E R (1962) Theory of resonance absorption of neutrons, *Nucl. Sci. Engng* **13**, 132-140.

KHAIRALLAH A and RECOLIN J (1972) Calcul de l'autoprotection résonnante dans les cellules complexes par la méthode des sous-groupes, *Proc. Seminar IAEA-SM-154 on Numerical Reactor Calculations*, pp 305-317, I.A.E.A., Vienna.

NOTARI C and GARRAFFO Z (1987) Spatial self-shielding for heterogeneous cells, *Ann. nucl. Energy* **14**, 615-618.

PRESS W H, FLANNERY B P, TEUKOLSKY S A and VETTERLING W T (1990) *Numerical Recipes (FORTRAN)*, Cambridge University Press, Cambridge.

ROTH M J (1974) Resonance absorption in complicated geometries, *UKAEA Report AEEW-R 921*, Winfrith, England.

ROTHENSTEIN W (1980) Resonance absorption calculations in thermal reactors, *Progr. nucl. Energy* **5**, 95-144.

SEGEV M (1981) Interpolation of resonance integrals, *Nucl. Sci. Engng* **17**, 113-118.

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

WEINBERG A M and WIGNER E P (1958) *The Physical Theory of Neutron Chain Reactors*, The University of Chicago Press.

IV. HELIOS: THE TRANSPORT CALCULATION

IV. 1 INTRODUCTION AND DISCRETIZATION

In HELIOS, two or three particle-transport calculations are required per reactivity point. The first is done as part of the resonance treatment to obtain the fluxes from which the lethargy dependence of the equivalence cross-sections (XSs) is established; see chapter III. The second yields the neutron fluxes from which the power distribution and the reaction rates for burnup are obtained. The third is optional - it gives the gamma fluxes from which the gamma-smearing effect on the power distribution and gamma-detector responses can be evaluated.

To solve the transport equation its variables must be discretized. The time variable is not considered—in the burnup calculations of chapter VII, it is only the number densities that change with time, not the flux distribution. The energy variable is discretized by dividing the energy range of interest into groups. The group division used is that of the nuclear-data library. Group fluxes and currents are always energy (or lethargy) integrated over the group g, i.e., $\phi_g = \int \phi(E)\,\mathrm{d}E$ and $j_g = \int j(E)\,\mathrm{d}E$.

What remains are the spatial and angular discretizations; they characterize the transport method. Because of the group discretization, the transport calculations are done per group. The one-group solutions are coupled by group sources and possibly boundary albedos. This implies, at least for neutrons, a process of group-to-group iterations which is described in section 4.

The transport method of HELIOS is called the CCCP method, because it is based on current coupling and collision probabilities.[1] One simplification is the restriction to two-dimensional systems. Otherwise, the system to be calculated consists of space elements (see chapter II.3) that are coupled with each other and with the boundaries by interface currents, while the properties of each space element—its responses to sources and in-currents—are obtained from CPs. The CCCP method is discussed in sections 2 and 3.

The spatial discretization of the system is the subdivision of its space elements into flat-flux regions and the subdivision of their periphery into flat-current straight-line and arc segments (see chapter II.2). Inside these regions the XSs are assumed to be constant.

There are two angular discretizations. First, scattering is assumed to be isotropic, with first-order anisotropic scattering approximated by the transport correction (see chapter II.5.2). Second, the angular dependence of the interface or coupling currents can be discretized in various ways. This is done by partitioning the directional half-sphere into a number of polar θ levels, and each θ level into a number of azimuthal φ intervals (see Fig. II.13). The sectors are defined such that for the current caused by an isotropic flux (cosine current), the same number of particles passes through each

[1] It should rather be *first-flight probabilities* because transmission and escape probabilities are included, but for reasons of tradition *collision probabilities (CPs)* will be used too.

sector. Moreover, inside each sector cosine currents are assumed. More details on the sectors are given in chapter V.

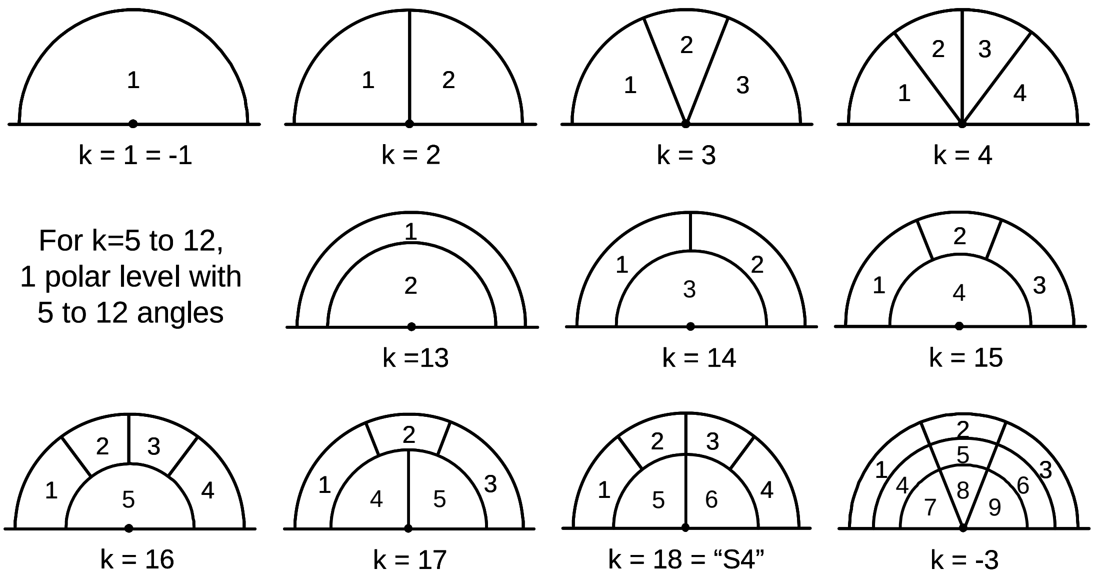


Fig. 1: Angular sectors and sector numbering for current coupling.

A number of discretizations, also called coupling orders, denoted by k, are built-in. Figure 1 shows the coupling orders currently available, except $k = 0$, $k = -4$ and $k = -5$. Zero coupling ($k = 0$) is the special case of "no coupling"; it is used to combine input structures into a space element, as mentioned in chapter II.3. A negative k represents $|k|$ θ levels, each with $|k|$ φ intervals; thus $k = 1$ and $k = -1$ are identical. Furthermore, each interface segment between two space elements or between a space element and a reflector element (see section 3.2) may have its own k.

Before proceeding with a more detailed description of the CCCP method, the convention for the indexes used in this chapter is given:

g, h	= generic group indexes
G	= total number of groups
e	= generic space-element index
E	= total number of space elements
i, j	= generic region indexes
I	= total number of regions (in a space-element)
s, t	= generic (interface) sector indexes
S	= total number of sectors (in a space element).

IV. 2 FLUXES

In this section and the next, the CCCP method is described in a given energy group g. In the present section, the evaluation of the flux in an arbitrary space element e is considered. The next section describes the coupling of the elements by interface currents. For brevity of notation, the group and element indexes are omitted whenever possible. First some definitions:

ϕ_i	=	the particle flux in region i integrated over its volume V_i.
Q_i	=	the volume-integrated particle source in region i.
j_s^-	=	the surface and sector-integrated in-current in the sector s, i.e., the fraction of the total in-current through the peripheral segment containing s that enters in this sector.
X_{ij}	=	the source-response flux, i.e., the volume-integrated flux in region i due to one particle born uniformly and isotropically in region j.
Y_{is}	=	the current-response flux, i.e., the volume-integrated flux in region i due to one particle entering the space element in sector s as an in-current caused by an isotropic flux.

Owing to the linearity of the transport equation—no particle-particle interactions—the flux in region i can be written as a linear combination of source and in-current contributions (Villarino and Stamm'ler, 1984),

$$\phi_i = \sum_{j=1}^{I} X_{ij} Q_j + \sum_{s=1}^{S} Y_{is} j_s^- \ . \tag{1}$$

To evaluate the flux from Eq. (1), the sources, currents, and response fluxes must be known. The next two sections, 2.1 and 2.2, address the evaluation of the sources and the response fluxes, while the problem of how to obtain the in-currents is treated in section 3.

IV. 2.1 Sources

Each of the three types of transport calculation has a different source. For the resonance calculations, this is a fixed source which does not depend on the other groups. Recalling Eq. (III.40), the source is

$$Q_i = (\lambda \Sigma_p)_i V_i \ . \tag{2}$$

For the neutron calculations, the source depends on the other groups. It consists of an in-scattering and a fission contribution, and is given by

$$Q_{ig} = \sum_{h \neq g}^{G} \Sigma_{i,g \leftarrow h} \phi_{ih} + \frac{\chi_{ig}}{K} \sum_{h=1}^{G} (\nu \Sigma_f)_{ih} \phi_{ih} \ . \tag{3}$$

In Eq. (3), K is the multiplication factor (the eigenvalue of the transport problem; see section 4.2) and $\Sigma_{i,g \leftarrow h}$ are the elements of the transport-corrected P_0-scattering matrix of chapter II.5.2. Because the transport-corrected diagonal elements are not involved, these elements are the same as those of the pure P_0-scattering matrix. The other quantities in Eq. (3), the macroscopic XSs and the fission spectrum, have been defined in chapter II.5.2.

For the gamma calculations, the source depends on the other groups. This time there is no up-scattering and, instead of a fission source, there is a fixed (n,γ) source. The expression for the source is

$$Q_{ig} = \sum_{h<g}^{G} \Sigma_{i,g\leftarrow h}\phi_{ih} + \sum_{n=1}^{G_n} \Sigma_{i,g\leftarrow n}\phi_{in} , \tag{4}$$

where the index n denotes neutron groups. The (n,γ) matrix is made up of shielded and unshielded contributions, depending on whether the material contains BA isotopes of type 991**, or resonance isotopes if n is a resonance group. This has been discussed in chapter II.5.5.

IV. 2.2 Response fluxes

The response fluxes of a space element are obtained from first-flight CPs by considering the total number of collisions in a given region i (Stamm'ler and Abbate, 1983). But first two more definitions, both pertaining to a single space element:

p_{ij}	=	the probability of particles born uniformly and isotropically in region j to suffer a first collision in region i.
p_{is}	=	the probability of particles entering the space element in sector s as an in-current caused by an isotropic flux, to suffer a first collision in region i.

Introduce these probabilities and apply the flat-flux approximation. Then the collision rate in region i, caused by either a unit source in region j or a unit in-current in sector s, can be written as

$$\left.\begin{aligned} \Sigma_i X_{ij} &= p_{ij} + \sum_{j'=1}^{I} p_{ij'} c_{j'} \Sigma_{j'} X_{j'j} \\ \Sigma_i Y_{is} &= p_{is} + \sum_{j=1}^{I} p_{ij} c_j \Sigma_j Y_{js} \end{aligned}\right\} . \tag{5}$$

In Eq. (5), Σ_i is the total (transport-corrected) XS in region i and c_i is the number of secondaries per collision inside group g, i.e., with Σ_{ir} the removal XS of Table II.1, one has

$$c_i = (\Sigma_i - \Sigma_{ri})/\Sigma_i \ . \tag{6}$$

After moving the summation terms to the left-hand side, it is seen that to solve the $I \times I$ source-response fluxes and the $S \times I$ current-response fluxes, it is sufficient to invert just one $I \times I$ matrix, **M**, whose coefficients are given by

$$M_{ij} = \Sigma_i \delta_{ij} - c_j \Sigma_j p_{ij} \ , \tag{7}$$

where δ_{ij} is the Kronecker delta function. The sought response fluxes are found by applying $\mathbf{M}^{-1}$ to the I column vectors of CPs of particles born in the regions $j = 1, \dots I$, and to the S column vectors of CPs of particles entering in the sectors $s = 1, \dots S$, col$\{p_{1j}, p_{2j}, \dots p_{Ij}\}$ and col$\{p_{1s}, p_{2s}, \dots p_{Is}\}$. Currently, single-precision Gauss elimination combined with back-substitution is used.

Thus, the problem of finding the response fluxes now has been shifted to the problem of finding the basic first-flight CPs, p_{ij} and p_{is}. Their evaluation is described in chapter V.

IV. 3 CURRENT COUPLING

The present section deals with the evaluation of the in-currents. Again, for simplicity of notation, group and space-element indexes are suppressed wherever possible. First, another two definitions:

P_{si}	=	the probability of particles born uniformly and isotropically in region i to leave the space element through sector s after any number of collisions (the total escape probability, not just first-flight).
P_{st}	=	the probability of particles entering the space element in sector t as an in-current due to an isotropic flux, to leave the space element through sector s after any number of collisions (the total transmission probability).

The out-current through a sector s of a given space element can be written as a linear combination of source and in-current contributions (Lindahl and Weiss, 1975; Weiss and Lindahl, 1981),

$$j_s^+ = \sum_{i=1}^{I} P_{si} Q_i + \sum_{t=1}^{S} P_{st} j_t^- \ . \tag{8}$$

To evaluate the out-current from Eq. (8), both the sources and the total escape and transmission probabilities must be known. For regular space elements, the sources are known from section 2.1. The evaluation of P_{si} and P_{st} from first-flight probabilities is discussed in section 3.1. The introduction of reflector elements, the interpretation of Eq. (8) for these reflector elements and the evaluation of the involved quantities, are discussed in section 3.2.

Each out-current is nothing but an in-current at the other side of the same segment. The sector-coupling matrix **H** that expresses this relation as follows:

$$\vec{j}^{\,+} = \mathrm{H} \vec{j}^{\,-} \ . \tag{9}$$

In Eq. (9), $\vec{j}^{\,+}$ and $\vec{j}^{\,-}$ are column vectors of the out- and in-currents in all the sectors of the system. Then **H** is a square matrix with as many rows as the number of sectors in the system; it is discussed in section 3.3. Finally, the iteration process for the currents that is implied in Eqs. (8-9) is discussed in section 3.4.

IV. 3.1 Escape and transmission probabilities

Like the response fluxes, the escape and transmission probabilities of a space element are obtained from first-flight probabilities. Apart from the earlier introduced p_{is}, two new probabilities are needed; their definitions are:

p_{si}	=	the probability of particles born uniformly and isotropically in region i to escape the space element uncollided through sector s.
p_{st}	=	the probability of particles entering the space element in sector t as an in-current caused by an isotropic flux, to leave the element uncollided through sector s.

The total escape and transmission probabilities, P_{si} and P_{st}, have two contributions: one from uncollided particles; the other from particles that have collided and remained in the same group. Upon introducing the above-defined p_{si} and p_{st}, and applying the flat-flux approximation, the following expressions for P_{si} and P_{st} are found:

$$\left.\begin{aligned} P_{si} &= p_{si} + \sum_{j=1}^{I} p_{sj} c_j \Sigma_j \;\; X_{ji} \\ P_{st} &= p_{st} + \sum_{j=1}^{I} p_{sj} c_j \Sigma_j Y_{jt} \end{aligned}\right\} . \tag{10}$$

IV. 3.2 Boundary conditions (BCs)

A reflector element is assigned to each boundary segment of the system. This is an element with only one segment, the boundary segment itself, and one region of zero volume. It represents the properties of the outside of the system. Of course, segments with boundary coupling (see chapter II.4) have on the other side, the "outside", another regular space element, i.e., they do not border on reflector elements.

IV. 3.2.1 Albedo conditions

The first (heterogeneous) term of Eq. (8) is the out-current due to sources in the regions of the space element. The reflector element has only one region, i.e., one source. That source is due to albedo in-scattering, which is assumed isotropic. Using index b for the reflector element, that source is the total out-current—back into the system—which is equally distributed over the S_b sectors. So for group g, the first term of Eq. (8) becomes:

$$P_{sb}Q_b = \frac{1}{S_b}\sum_{h\neq g}^{G} a_{b,g\leftarrow h}\, j_{bh}^{-} \quad . \tag{11}$$

In Eq. (11), the albedo matrix element $a_{g\leftarrow h}$ is defined as the probability of particles leaving the system through a boundary segment in group *h*, to return through that same segment in group *g*. Observe that there is only one term on the left side because the reflector element consists of only one region. In the resonance transport calculations, which are a set of uncoupled one-group calculations, this entire term does not exist.

The second term on the right-hand side of Eq. (8) is the out-current—back into the system—from reflection in the group *g* itself. This reflection probability is described by the diagonal elements of the albedo matrix, $a_{g\leftarrow g}$. It is only for this term that the angular discretization *k* is used. For each sector of entry, *t*, there is just one exit sector, *s*, for which $P_{st} = a_{g\leftarrow g}$; the other $P_{s'\neq s,t} = 0$. Referring to the sector numbering for $k = 18$ in Fig. 1, its $\mathbf{P}(s,t)$ matrix for the only segment of a reflector element is

$$\begin{pmatrix} 0 & 0 & 0 & a_{g\leftarrow g} & 0 & 0 \\ 0 & 0 & a_{g\leftarrow g} & 0 & 0 & 0 \\ 0 & a_{g\leftarrow g} & 0 & 0 & 0 & 0 \\ a_{g\leftarrow g} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{g\leftarrow g} \\ 0 & 0 & 0 & 0 & a_{g\leftarrow g} & 0 \end{pmatrix} . \tag{12}$$

IV. 3.2.2 Specular boundary

The specular boundary condition is included in the first-flight probabilities; each time a particle reaches a specular boundary, it is reflected and followed further (see chapter V). As there is no current coupling at specular boundary segments, currents would not be needed if the only purpose of the transport calculation were to obtain fluxes. However, boundary currents are needed if the output processor ZENITH has to evaluate discontinuity (heterogeneity) factors.

Therefore, each time a particle is specularly reflected, this reflection is registered and added to the first-flight probabilities, p_{si} and p_{st}. Further, reflector elements with the identity matrix as albedo matrix are assigned to all the specular segments. In that way, both out and in-currents at specular segments can be evaluated automatically—i.e., without IF statements for the boundaries.

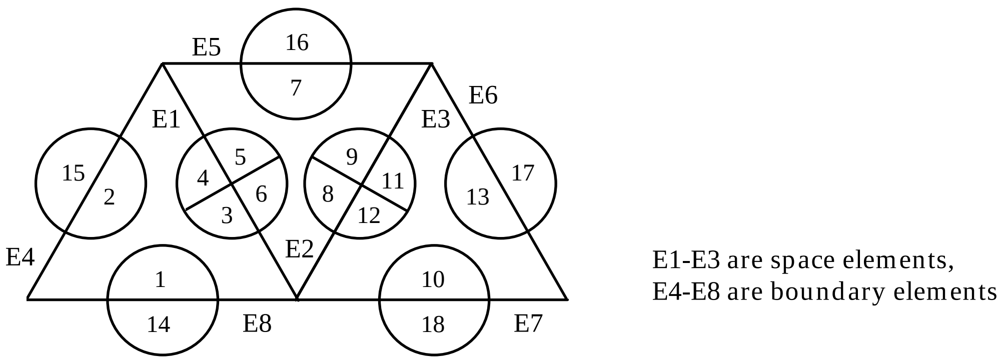
These "false" in-currents from the boundary segments must not be added to the particles whose reflection already is included in the first-flight probabilities. Therefore, the first-flight CPs and transmissions of particles that enter in a sector *s* of a specular segment are set to zero. By virtue of Eqs. (5) and (10), the corresponding current-response fluxes and total transmissions also are zero. In other words:

$$(p_{is} = 0, p_{ts} = 0) \;\Leftrightarrow\; (Y_{is} = 0, P_{ts} = 0) \; . \tag{13}$$

This ensures that "false" in-currents cannot contribute to the flux of Eq. (1), nor to the out-current of Eq. (8). Because only the total in- and out-current are needed, there is no need to have more than one sector at specular segments. Otherwise, the albedo matrix is diagonal, so $P_{sb}Q_b = 0$ and $a_{g \leftarrow g} = 1$.

IV. 3.3 The sector-coupling matrix H

The **H** matrix assigns to each sector its counterpart on the other side of the same interface segment. This very large, square matrix is also very sparse, with only one non-zero entry per row. Because the outsides of the boundary segments are included in the system as one-dimensional space elements with their own sectors, this non-zero entry is everywhere a 1, even for the boundary sectors.



```
            1             5                  10                 15         18
JPLAUX() = (14, 15, 5, 6, 3, 4, 16, 11, 12, 18, 8, 9, 17, 1, 2, 7, 13, 10)
```

Fig. 2: Illustration of the sector-coupling array JPLAUX().

In the program, this information is not stored as a matrix, but as the one-dimensional array JPLAUX(). The one-to-one relation between the in-current in sector S of one element and the out-current through sector T of its neighbour (it is the same current) can be expressed as

$$\text{JPLAUX(S) = T} \quad \Leftrightarrow \quad \text{JPLAUX(T) = S} \,. \tag{14}$$

Figure 2 illustrates the principle of the sector-coupling array JPLAUX() for a small system made up of the three triangular space elements E1–E3. The five boundary segments generate equally many reflector elements (with zero volume), E4–E8. The numbers inside the circles at the midpoints of the segments denote the sectors—the currents are not concentrated at these midpoints, but are flat along the segments. The coupling between the space elements is $k = 2$, while the coupling with the outside is everywhere $k = 1$.

IV. 3.4 Inner iterations

The currents are solved from Eqs. (8-9) by the inner iterations. This is done for one group at a time (the group-to-group iterations are discussed in section 4). HELIOS, uses Gauss-Seidel or Liebmann iterations with relaxation and normalization. When starting the iterations in a given group, it is assumed that the sources and the heterogeneous first terms of Eq. (8) have already been calculated. Further, it is assumed that start values of the in-currents are available. These start values come either from the previous solution in the same group or from an initial guess, as described in section 4.1.

The iterations go from one space element to the next, reflector elements included. The sequence in which the elements are treated is according to the total out-current due to their sources. For element *e*, this total out-current is

$$L_e = \sum_{s=1}^{S(e)} \sum_{i=1}^{I(e)} P_{si} Q_i \quad . \tag{15}$$

where $S(e)$ and $I(e)$ are the total number of sectors and regions of element *e*. The first element treated is that with the greatest L_e, and so on.

In each space element, Eq. (8) is used to evaluate, per sector, its out-current. From JPLAUX(), its position as in-current in the sector *s* of a neighbour element is found. The two in-currents are compared, and the one just found is relaxed with the old one to yield the new in-current, according to

$$j_s^{(t)} = j_s^{(t-1)} + \omega\left(j_s - j_s^{(t-1)}\right) \quad . \tag{16}$$

In Eq. (16), ω is the relaxation factor, t is the iteration counter ($j^{(0)}$ is the start value), and j_s is the newly calculated current before relaxation. During the first iteration, $\omega = 1$; thereafter, the default value is used. The default value is also 1, but the User may change this through the input. It appears that cases that do not converge need under-relaxation ($\omega < 1$), not over-relaxation.

The currents of the t^{th} iteration replace (overwrite) the currents of the $(t\!-\!1)^{th}$ iteration as soon as they have been evaluated. So in Eq. (8), currents of both the actual and the previous iteration will occur. This is typical of the Gauss-Seidel or Liebmann iteration process.

The inner iterations in group *g* are considered converged after *T* iterations, and the iteration process is stopped, if

$$\left| j_s^{(T)} - j_s^{(T-1)} \right| < \varepsilon j_{\max}^{(T)} \quad \text{for all } s \text{ of the system} \quad . \tag{17}$$

This criterion avoids unnecessary iterations on very small and unimportant currents. The default value of ε is 0.000 25, a value that the User may change through the input. The fulfilment of condition (17) does not imply that the currents are correct within the same accuracy. In general, for slowly convergent cases, the error in the currents may be greater than ε by more than a factor of 10, while for fast converging cases the error in the currents may be less than ε. By fast convergence, it is understood that less than four iterations are needed to reduce the left-hand side of Eq. (17) by a factor of 10.

So if convergence is slow, a smaller ε value might be needed. The User has still another option: it is possible to prescribe that after convergence of the inner iterations, a given number of extra inner iterations be done.

Before the first inner iteration in a group, and after each inner iteration, the currents are <u>normalized</u> to satisfy the balance that total losses equal total sources. Equation (8) is summed over all the elements and reflector elements of the system. The sources (the first term on the right-hand side) are collected, and so are the losses. The latter are proportional to the currents, i.e., the start values or the latest iterate. The normalization factor by which the currents are multiplied is the ratio of the sources to the losses. This factor is

$$f_{norm} = \frac{\sum_{e=1}^{E} L_e}{\sum_{e=1}^{E}\sum_{s=1}^{S(e)} \Gamma_{es} j_{es}^{-}} \,, \tag{18}$$

where L_e is given by Eq. (15), and Γ_{es} is the blackness of sector s of element e (the probability of particles entering e through s to be absorbed or scattered out of the group), which is given by

$$\Gamma_{es} = 1 - \sum_{t=1}^{S(e)} P_{e,ts} \;. \tag{19}$$

For sectors on a specular segment of a regular space element, Γ_{es} is zero.

IV. 4 MULTIGROUP CALCULATIONS

The transport solution in a given group depends, through the source, on the solutions in the other groups. However, for the <u>resonance</u> calculations, this dependence does not exist. There, a "group" is one of the M σ_m values of one of the resonance categories (see chapter III), and its source is fixed—Eq. (2). Thus, per calculational point, each "group" is solved only once.

The <u>gamma</u> source of Eq. (4) has two contributions. The fixed (n,γ) source determines the level of its flux-and-current solution, while the down-scattering source couples the groups (together, they determine the level of the solution in an individual group). Because there is only down-scattering, this coupling does not generate group-to-group iterations. Going from high to low energies, the groups have to be solved only once per calculational point.

The <u>neutron</u> source of Eq. (3) has two contributions: one due to fission, the other due to up and down-scattering. They generate two types of group-to-group iterations: the eigenvalue or outer iterations, and the thermal or up-scattering iterations. Although this section deals mainly with group-to-group iterations for neutrons, differences with the other transport calculations are pointed out when necessary.

IV. 4.1 The initial guess

In a HELIOS case, the geometry remains unchanged, while the properties of the regions may change due to different materials, temperatures, and/or density-correction factors. Thus, except for the first time, there is always an earlier solution in all the regions and groups to provide start values for the group-to-group iterations. If no earlier solution is available, an initial guess is set up from scratch. In that case, a unit isotropic flat flux is assumed throughout the system, in all the groups. Recalling that for an isotropic flux $j = \phi/4$, and that in HELIOS fluxes and currents are volume-integrated and surface-and-sector-integrated, respectively, the initial guess is

$$\phi_i = V_i \text{ and } j_s^- = l_n/(4S_n) \ , \tag{20}$$

where l_n and S_n are the length of segment n and its number of sectors, respectively. This is also done for the resonance and gamma calculations.

If the initial guess is set up from scratch, its level may be off by orders of magnitude. This does not matter in the epithermal groups, where the source has no up-scattering component. That is because: (i) the calculations proceed from the first group (with the highest energy) downward; and (ii) the fission component of the source is kept fixed during a pass through all the groups. So the source does not depend on the lower groups, which have not yet been calculated with the actual fission source. The normalization at the beginning of the inner iterations then adjusts the currents to the correct level, that of the group source. Notice that this is always the case with the resonance and gamma transport calculations.

However, in the thermal groups, the source depends also on the in-scattering from groups that have not yet been calculated. Their up-scattering contributions to the source may be incorrect by orders of magnitude. The normalization of the inner iterations then would adjust the currents to the incorrect level of the group source. That is why the initial-guess fluxes and currents are normalized to one fission neutron born in the system. This corresponds to assuming that the eigenvalue $K = 1$ (see section 4.2). The normalization factor is

$$f = \frac{1}{\sum_i \sum_g (\nu\Sigma_f)_{i,g} \phi_{i,g}^{(-1)}} \ , \tag{21}$$

where $\phi^{(-1)}$ denotes the initial-guess flux before the normalization.

Once the neutron calculations have converged, a criticality (B_1) spectrum is enforced on their solution (see chapter VI), which is also scaled to power (see chapter VII). In view of the importance of the level of the start values, this solution cannot be used to provide the start values at a next calculational point. It is therefore in each group first multiplied by the inverse of the product of the scaling factor and the B_1 factor of that group.

IV. 4.2 The fission source

Each eigenvalue iteration begins with the evaluation of a fission source normalized to one neutron born in the system. For the t^{th} eigenvalue iteration, the fission rate in region i is

$$F_i^{(t)} = \frac{1}{K^{(t-1)}} \sum_g (\nu\Sigma_f)_{ig} \phi_{ig}^{(t-1)} \ . \tag{22}$$

The total fission rate is always normalized to 1. Thus, the flux-and-current solution resulting from such a fission rate always corresponds to a volume-and-energy integrated loss rate of 1. Therefore, the eigenvalue of the $(t-1)^{th}$ iterate in Eq. (22) is the total fission rate,

$$K^{(t-1)} = \sum_i \sum_g (\nu\Sigma_f)_{ig} \phi_{ig}^{(t-1)} \ . \tag{23}$$

Finally, the fission source in region i and group g is the fission rate multiplied by the fission spectrum χ_{ig}. Of course, χ_{ig} summed over all the groups is 1. The fission rates are also relaxed; see Eq. (16). The default relaxation factor is $\omega = 1.3$, which can be modified by the User. During the first two eigenvalue iterations, no relaxation is applied, i.e., $\omega = 1$. This prevents contamination of the fission rates of the actual calculational point with the solution of the previous reactivity point.

IV. 4.3 Fundamental-mode rebalancing

At the end of each iteration sweep, when the last group has been calculated, what is called fundamental-mode rebalancing[2] is applied to all the groups that receive up-scattering, the thermal groups. This is nothing but a group-dependent normalization, which is described below. First, some quantities in group g are defined, where g is a thermal group:

Q_g	=	the volume-integrated source rate and surface-integrated in-leakage rate due to fission and slowing down from the epithermal groups. This plays the role of a fixed source into the thermal groups and is given by

$$Q_g = \sum_b \sum_{h \le g_{epi}} a_{b,g\leftarrow h} j_{bh}^{-} + \sum_i \sum_{h \le g_{epi}} \Sigma_{i,g\leftarrow h} \phi_{ih} + \frac{1}{K} \sum_i \chi_{ig} \sum_{\text{all } h} (\nu\Sigma_f)_{ih} \phi_{ih} \ , \tag{24}$$

[2] The method was suggested by Bo Fredin, ASEA-Atom, in 1969.

where g_{epi} is the last epithermal group and b denotes the boundary sectors.

A_g = the volume-integrated absorption (not removal!) rate and surface-integrated net leakage rate of the system, in group g, i.e.,

$$A_g = \sum_i \Sigma_{a,ig}\phi_{ig} + \sum_b \Big(1 - \sum_{\text{all } h} a_{b,h\leftarrow g}\Big) j_{bg}^- \ . \tag{25}$$

S_{gh} = the volume-integrated scattering rate and surface-integrated albedo "scattering" rate from thermal group h to thermal group g, i.e.,

$$S_{gh} = \sum_i \Sigma_{i,g\leftarrow h}\phi_{ih} + \sum_b a_{b,g\leftarrow h}\, j_{bh}^- \ . \tag{26}$$

The group-dependent normalization factors f_g can then be solved from the balance equations for the thermal groups, $g = g_{epi}+1, \ldots G$, i.e., from

$$\Big(A_g + \sum_{h>g_{epi}} S_{hg}\Big) f_g = \sum_{h>g_{epi}} S_{gh} f_h + Q_g \ . \tag{27}$$

Once the f_g are found—by Gauss elimination with back-substitution—the thermal fluxes and currents are multiplied by f_g. In the highest thermal groups, which receive little up-scattering, these factors will be close to 1. Of course, all the f_g are 1 once the thermal iterations (see section 4.4) have converged.

In HELIOS, the S_{gh} are evaluated before the flux-and-current solutions have been calculated in the groups $h > g$. This is corrected before solving Eq. (27), by multiplying the offending S_{gh} by the ratio of the total flux of the present eigenvalue iteration to that of the previous one,

$$f_{\phi_h} = \frac{\sum_i \phi_{ih}^{(t)}}{\sum_i \phi_{ih}^{(t-1)}} \ . \tag{28}$$

IV. 4.4 Group-to-group iterations

The multigroup iteration scheme is shown in Fig. 3. Each eigenvalue iteration starts with the calculation of the fission rate and the estimate of its corresponding eigenvalue K, according to Eqs. (22) and (23), respectively. Then, for each group, the source is evaluated by adding the in-scattering contributions to the fixed source; this includes albedo in-scattering contributions. The resonance calculations have no in-scattering source, only a fixed source. For the neutron calculations, the fixed source is χ_{ig} times the fission rate at the beginning of an eigenvalue iteration.

Once the source is known, the currents are obtained from the inner iterations, as described in section 3.4. Then, finally, the fluxes are evaluated from the currents and the sources, according to Eq. (1). This completes the calculation in one group.

Having calculated the last group, $g = G$, a fundamental-mode rebalancing is done for all the groups that receive up-scattering. Then, before returning to the first group and starting the next eigenvalue iteration sweep, one or more sweeps through the thermal groups are made. These thermal iterations are made because group coupling by scattering is much stronger than by fission, at least in thermal reactors. In fact, below about 0.18 eV, up-scattering is so strong that for every thermal iteration sweep one or more extra deep-thermal iteration sweeps are made through the groups below about 0.18 eV.

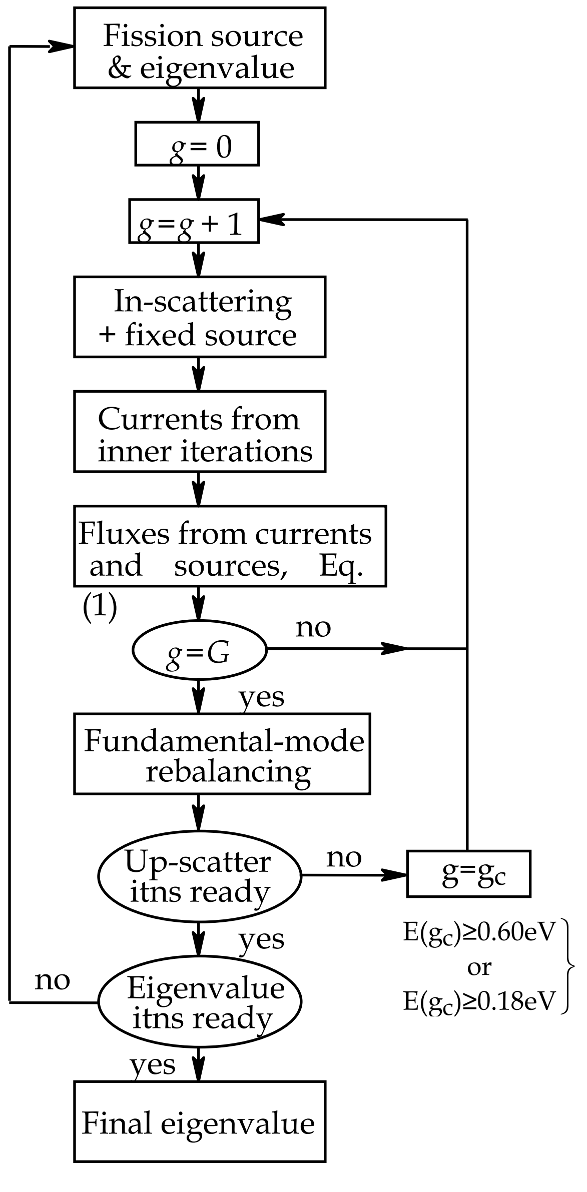

Fission source & eigenvalue
g = 0
g = g + 1
In-scattering + fixed source
Currents from inner iterations
Fluxes from currents and sources, Eq. (1)
g = G
no
yes
Fundamental-mode rebalancing
Up-scatter itns ready
no
g=g_c
yes
E(g_c)≥0.60eV
or
E(g_c)≥0.18eV
no
Eigenvalue itns ready
yes
Final eigenvalue

Fig. 3: The multigroup iteration process.

It is not necessary to pursue the inner iterations or the thermal iterations to full convergence during the first few eigenvalue iterations, where a "converged" solution in a group can still change considerably in the next eigenvalue iteration because of the modified fission and scattering sources. That is why, during the first few eigenvalue iterations, a reduced accuracy is used for ε of the inner iterations (see Eq. (17)), which is sharpened successively to its final desired value. Of course, testing on eigenvalue convergence should then not begin before the final ε is used in the inner iterations. The following strategy is used:

- If the iteration process starts from scratch, then the first eigenvalue iteration uses $\varepsilon = 0.05$ in the inner iterations. This is the 0th iteration, after which the same procedure is followed as when start values from a previous calculational point are available; see the next point.
- If the iteration process begins with start values from a previous calculational point, then the first eigenvalue iteration uses $\varepsilon = 10 * \varepsilon$ in the inner iterations. The second and later iterations all use the desired ε.

Not all the groups with up-scattering participate in the thermal iterations. Groups above about 0.6 eV are excluded because they receive very weak up-scattering, i.e., the source is strongly dominated by down-scattering. These groups are treated like the epithermal groups, except that they also participate in the fundamental-mode rebalancing.

At most four thermal iterations are made per eigenvalue iteration, while at most two deep-thermal sweeps are made per thermal iteration. Of course, this scheme, which has been adapted to thermal reactors, may be modified by the User through input.

Relaxation is not applied to the solutions of groups that do not participate in the thermal iterations. These solutions are fully, or almost fully (in case of weak up-scattering), determined by the fission source of the present eigenvalue iteration. If relaxation were used, they would become contaminated with the solutions of the preceding eigenvalue iteration, thereby reducing the rate of convergence.

For similar reasons, relaxation of the thermal iterations is avoided during the first thermal iteration sweep of each eigenvalue iteration. Then the previous solution still is part of the previous eigenvalue iteration. However, in the second and later thermal iterations, relaxation is applied. The default value of this relaxation is $\omega = 1.4$, which may be changed by the User.

The eigenvalue iterations have converged when the following two conditions are satisfied (the User may request additional eigenvalue iterations):

- Each group needs just one inner iteration to converge.
- The eigenvalue K has converged within a preset δ, whose default value of 10^{-5} can be modified by the User. This condition is

$$|K - K^p| < \delta, \tag{29}$$

IV. 5 TRANSFORMATION LAWS

The evaluation of the response fluxes X_{ij} and Y_{is} is time consuming on two scores - both the evaluation of the underlying first-flight probabilities, and the subsequent matrix inversion, Eqs. (5-7), are costly in computer time. Therefore, a perturbation method has been developed that strongly reduces the time to calculate X_{ij}, Y_{is}, P_{si} and P_{st}.

Often, the many space elements that make up the system belong to a few element types only, i.e., they have the same geometry. At zero burnup, such elements can contain the same material or a material whose XSs in some energy groups, in particular the fast groups, are nearly the same. Even during burnup, there may be many elements of the same type whose XSs remain the same, or nearly the same, e.g., pin cells in symmetric positions.

Thus, response fluxes frequently are needed in elements with XSs (nearly) equal to those of another element of the same type, or of the element itself at an earlier burnup. This fact is utilized to save computer time. In the trivial case of very small differences, the previous response fluxes are used directly. Otherwise, so long as the differences are less than a preset value, transformation laws are used. Section 5.1 gives a derivation of the transformation laws and section 5.2 describes how they are applied in HELIOS.

IV. 5.1 A derivation of the transformation laws

Takahashi (1966) first derived a transformation law for first-flight CPs and applied it to one and two-region space elements. His incentive was to obtain approximate CPs from exact values at different XSs, at the same time preserving their conservation and reciprocity relations. In a review article, Lefvert (1979) analysed this transformation law in some simple cases and suggested its use in multigroup calculations. Krishnani (1985) expanded this transformation law to the various probabilities used in the current-coupling methods and implemented these laws in a fuel-burnup code. Villarino, et al. (1993) expanded these laws to angular-dependent first-flight probabilities and implemented them in an earlier HELIOS version.

While these applications avoid the expensive evaluation of the first-flight probabilities, they do not affect the sometimes time-consuming matrix inversion. The present method, which is based on an idea of Wio (1984), applies similar laws directly to the response fluxes, thereby also avoiding the inversion of the CPs.

Consider, for a given group g, two states, the actual and the previous state. Let the actual XSs in a space element have changed as follows (the previous state is indexed by 0; the actual state is not indexed):

$$\Sigma = \Sigma^0 + \delta\Sigma; \quad \Sigma_s = \Sigma_s^0 + \delta\Sigma_s; \quad \Sigma_r = \Sigma_r^0 + \delta\Sigma_r \ , \tag{30}$$

where Σ, Σ_s, and Σ_r are the total, self-scattering, and removal XSs in group g. In the case of isotropic sources, self-scattering included, the transport equation of the actual state becomes:

$$\left.\begin{aligned}\vec{\Omega}\cdot\vec{\nabla}\Phi(\vec{r},\vec{\Omega})+\Sigma^{0}(\vec{r})\Phi(\vec{r},\vec{\Omega}) &= \frac{Q(\vec{r})+\Sigma_{s}(\vec{r})\phi(\vec{r})}{4\pi}-\delta\Sigma\Phi(\vec{r},\vec{\Omega})\\ &\cong \frac{Q(\vec{r})+[\Sigma_{s}^{0}(\vec{r})-\delta\Sigma_{r}(\vec{r})]\phi(\vec{r})}{4\pi}\end{aligned}\right\} . \qquad (31)$$

When bringing $\delta\Sigma\Phi$ to the source side of Eq. (31), it is assumed that $\delta\Sigma$ is so small with respect to the source that the anisotropy of the angular flux Φ may be neglected. From Eq. (31), it follows that the flux in the actual state can be evaluated by using the XSs of the previous state, provided the source is reduced by $\delta\Sigma_r\phi$.

With this in mind, Eqs. (1) and (8) for the fluxes and currents of the actual state can be rewritten in matrix notation as:

$$\left.\begin{aligned}\vec{\phi} &= \mathrm{X}\vec{Q}+\mathrm{Y}\vec{j}^{-} &&= \mathrm{X}^{0}(\vec{Q}-\mathrm{D}_{r}\vec{\phi})+\mathrm{Y}^{0}\vec{j}^{-}\\ \vec{j}^{+} &= \mathrm{E}\vec{Q}+\mathrm{T}\vec{j}^{-} &&= \mathrm{E}^{0}(\vec{Q}-\mathrm{D}_{r}\vec{\phi})+\mathrm{T}^{0}\vec{j}^{-}\end{aligned}\right\} . \qquad (32)$$

In Eq. (32), $\vec{\phi}$ and $\vec{Q}$ are the fluxes and sources in all regions, and $\vec{j}^{-}$ and $\vec{j}^{+}$ are the in- and out-currents in all sectors of the space element, represented by column vectors with I and S entries. $\mathbf{X}^0$ is the IxI matrix whose columns j are the response fluxes in all regions i due to a unit source in region j; $\mathbf{Y}^0$ is the IxS matrix whose columns s are the response fluxes in all regions i due to a unit in-current in sector s; $\mathbf{E}^0$ is the SxI matrix whose columns j are the escape currents through all the sectors s due to a unit source in region j; and $\mathbf{T}^0$ is the SxS matrix whose columns s are the transmission currents leaving the space element through all the sectors t due to a unit in-current in sector s. Further, Δ_r is the diagonal matrix with $\delta\Sigma_{ri}$ as diagonal elements.

From Eq. (32), the following expressions for **X**, **Y**, **E** and **T** can be derived:

$$\left.\begin{aligned}\mathrm{X} &= (\mathrm{I}+\mathrm{X}^{0}\mathrm{D}_{r})^{-1}\mathrm{X}^{0}\\ \mathrm{Y} &= (\mathrm{I}+\mathrm{X}^{0}\mathrm{D}_{r})^{-1}\mathrm{Y}^{0}\\ \mathrm{E} &= \mathrm{E}^{0}-\mathrm{E}^{0}\mathrm{D}_{r}\mathrm{X}\\ \mathrm{T} &= \mathrm{T}^{0}-\mathrm{E}^{0}\mathrm{D}_{r}\mathrm{Y}\end{aligned}\right\} . \qquad (33)$$

The first two equations of Eq. (33) show that, to solve the IxI source-response fluxes and the SxI current-response fluxes of the new state, it is sufficient to invert just one IxI matrix, $\mathbf{M}=\mathbf{I}+\mathbf{X}^0\Delta_r$, whose coefficients are given by

$$M_{ij}=\delta_{ij}+\delta\Sigma_{rj}X_{ij}^{0} \ , \qquad (34)$$

with δ_{ij} the Kronecker delta function. The response fluxes are then found by applying $\mathbf{M}^{-1}$ to the I column vectors of the previous state's response fluxes due to particles born in the regions $j=1, \ldots I$, and to the S column vectors of the previous state's response fluxes due to particles entering in the sectors $s=1, \ldots S$. Currently, single-precision Gauss elimination with back-substitution is used. Note the similarity with Eq. (7).

The third equation of Eq. (33) yields the following expression for the total escape probabilities:

$$P_{si} = P^0_{si} - \sum_j P^0_{sj} \delta\Sigma_{rj} X_{ji} \ , \tag{35}$$

where the elements of the escape matrix are $E_{si} = P_{si}$. Note the similarity with the first equation of Eq. (10).

The fourth equation of Eq. (33) yields the following expression for the total transmission probabilities:

$$P_{st} = P^0_{st} - \sum_j P^0_{sj} \delta\Sigma_{rj} Y_{jt} \ , \tag{36}$$

where the elements of the transmission and escape matrices are $T_{st} = P_{st}$, and $E_{sj} = P_{sj}$, respectively. Note the similarity with the second equation of Eq. (10).

IV. 5.2 Application of the transformation laws

Consider a space element *e,* of type *k,* in group *g*. There are three ways to obtain its response fluxes X_{ij} and Y_{is}, and its total escape and transmission probabilities P_{si} and P_{st}: (i) direct copying; (ii) using the transformation laws of section 5.1; and (iii) exact calculations. In the case of exact calculations, first-flight probabilities are evaluated as described in chapter V and are then used to obtain X_{ij}, Y_{is}, P_{si} and P_{st} as described in sections 2.2 and 3.1.

The test to determine which of these three options to use requires, apart from the actual XSs, two more sets of XSs: the total XSs of the last exact calculation and the removal XSs of the previous calculation (which is not necessarily the same as the last exact calculation). Direct copying is done if the maximum relative difference of both total and removal XSs in all regions is smaller than $0.4\varepsilon_{inner}$, with ε_{inner} the convergence criterion of the inner iterations; see Eq. (17). In formula, the criterion for direct copying is

$$\left.\begin{aligned} d &= \max_{i=1...I}\{|\delta\Sigma_i / \Sigma_i|\} < 0.4\varepsilon_{inner} \\ d_r &= \max_{i=1...I}\{|\delta\Sigma_{ri} / \Sigma_{ri}|\} < 0.4\varepsilon_{inner} \end{aligned}\right\} \text{with } \delta\Sigma/\Sigma = 1 - \Sigma^x/\Sigma^a \ , \tag{37}$$

where the superscript *a* denotes the actual calculation, and *x* the last exact calculation for Σ_i or the previous calculation for Σ_{ri}.

For the actual space element *e*, these tests start with a comparison of *e* with itself. If the direct-copying criterion is not fulfilled, the same comparison is done for space element *e* and all the space elements $f < e$ of type *k* that have already been treated in the actual calculational point. This comparison is interrupted, and direct copying is done, as soon as the criterion of Eq. (37) is met.

If criterion (37) is not met, the minimum of the maximum total-XS differences d is found for all elements $f \le e$ that are of the same type as e (having the same regions and sectors). Let this minimum occur for element f' and let it be smaller than the criterion for applying transformation laws ε, i.e.,

$$d_{\min} = \min_{f \le e}\{d(f)\} = d(f') < \varepsilon; \quad e, f \in k \ . \tag{38}$$

If criterion (38) is met, the transformation laws are applied, using X_{ij}, Y_{is} P_{si} and P_{st} of element f' as previous values. If criterion (38) is not met, exact calculations are done. However, if the element has geometric symmetry, the exact calculations can still be reduced as described below.

HELIOS detects and utilizes geometric symmetries of space elements to reduce the computational time of the first-flight probabilities (see chapter V). If the symmetry is perturbed by small XS differences, it still may be utilized.

First, maximum and minimum values of Σ_i and Σ_{ri} are obtained for the regions i of all the symmetry sections of the element. If the maxima of their relative differences meet either criterion (37) or (38), an exact calculation is done for a symmetry section with Σ_i and Σ_{ri} that are averages of their maximum and minimum values. Then, depending on which criterion is met, the results are expanded to the different symmetry sections of the element by direct copying or by using transformation laws. If neither criterion is met, an exact calculation is done for the entire space element.

IV. 6 SOME REMARKS ON THE PROGRAM

The response fluxes and the total escape and transmission probabilities are evaluated in LINK15 for each calculational point. In this same module the first-flight probabilities are evaluated, which is discussed in chapter V. The group-to-group calculations are done in LINK16.

IV. 6.1 Notes on LINK15

LINK15 uses the following subroutines to evaluate the response fluxes, and the total escape and transmission probabilities needed in Eq. (1) and Eq. (8):

COLRMD:	It calls per group and per element the subroutines that evaluate (first and) multiple-flight quantities. It calls CPTYPE to determine what method to use: copying, transformation laws, or exact calculations. It calls MOVETI for direct copying, COLPRO for first-flight probabilities and RMDATA for multiple-flight quantities from exact calculations or transformation laws. Finally, for reflector elements, it sets up the total transmission matrix $\mathbf{P}(s,t)$ and the related blacknesses Γ_s; see Eqs. (12) and (19).

CPTYPE: It determines how to evaluate the multiple-flight quantities as described in section 5.2. Also the tests for symmetric elements are done here.

MOVETI: It is a general subroutine; in this module it copies the multiple-flight quantities.

RMDATA: It evaluates multiple-flight quantities. It exploits the similarity of Eqs. (34-36) with Eqs. (7) and (10), to calculate these quantities in almost exactly the same way, whether from first-flight probabilities and the flat-flux approximation or from transformation laws.

IV. 6.2 Notes on LINK16

The flux-and-current solution at each calculational point is obtained from the group-to-group calculations. LINK16 performs the group calculations with inner iterations on the currents, and the eigenvalue iterations. The following subroutines are used by this module:

INITIA: It initializes the start values for the actual solution as described in section 4.1. However, the normalization of the initial guess to $K = 1$ according to Eq. (21) is done in the subroutine FISSOR.

EIGITR: It controls the multigroup calculations described in section 4. It calls, directly or indirectly, all the routines mentioned below to perform the multigroup calculations.

FISSOR It is called—only for neutron calculations—by EIGITR to set up the fission rates at the start of an eigenvalue iteration, to normalize them to 1 fission neutron born in the system, and to evaluate the new eigenvalue K, using Eqs. (22) and (23). It also normalizes the initial guess (from scratch) to $K = 1$.

SOURCE: It is called per group by EIGITR to calculate the in-scattering. It calls QUELLE to evaluate sources; in-currents from reflector elements due to albedo scattering; and, for fundamental-mode rebalancing, the denominator of f_{ϕ_h}, Q_g and S_{gh}—see Eqs. (28), (26) and (24). It also evaluates the first term on the right side of Eq. (8), and the total out-currents due to sources from Eq. (15). Finally, it calls DECREA to find the sequence in which to treat the space elements in the inner iterations.

QUELLE: It is called from SOURCE. Depending on the arguments, it evaluates either sources for the regions or in-currents from reflector elements due to albedo scattering. It also builds up some of the fundamental-mode rebalancing quantities; see SOURCE above.

DECREA: This general subroutine, which is here called by SOURCE, returns the sequence of the elements according to decreasing total out-currents due to sources or due to albedo scattering.

ITERAT: It is called per group by EIGITR to perform the inner iterations as described in section 3.4.

FLUXES: It is called per group by EIGITR to obtain the fluxes from Eq. (1). It also builds up A_g from Eq. (25) for the fundamental-mode rebalancing.

FUNMOD: It is called—only for neutron calculations—by EIGITR to perform, at the end of each iteration sweep (after group G has been calculated), the fundamental-mode rebalancing of section 4.3.

REFERENCES

KRISHNANI P D (1985) "Transformation law for probabilities in the interface current method and its application to cluster geometry," *Annals of Nucl Energy* **12**, 675-680.

LEFVERT T (1979) "New applications of the collision probability method in neutron transport theory," *Prog. Nucl. Energy* **4**, 97-118.

LINDAHL S-Ö and WEISS Z J (1975) "High-order response matrix equations in two-dimensional geometry," *Nucl. Sci. Eng.* **58**, 166-181.

MARCHUK G I (1961) "A Survey of Nuclear-reactor Design Methods," *Atomnaya Energiya* **11** (4), 356.

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

TAKAHASHI H (1966) "Approximation for the Calculation of the Generalized First-Flight Collision Probability," *Nucl. Sci. Eng.* **26**, 254-261.

VILLARINO E A, MÅRTENSSON E, and STAMM'LER R J J (1993) "HELIOS: Usage of Transformation Laws for Angularly-Dependent Collision Probabilities," *Joint Intl Conf on Mathematical Methods and Supercomputing in Nuclear Applications*, Karlsruhe, Germany, 19-23 April, Vol. **2**, p. 433-444.

VILLARINO E A and STAMM'LER R J J (1984) The heterogeneous response method in slab geometry, *Annals of Nucl Energy* **11**, 429-440.

WEISS Z J and LINDAHL S-Ö (1981) "The response matrix method," *Advances in Nuclear Science and Technology* **13**, 73-154; Plenum Press, New York.

WIO H S (1984) "Transformation law for response fluxes within the collision probability method," *Annals of Nucl. Energy* **11**, 425-428.

V. HELIOS: COLLISION PROBABILITIES[1]

V. 1 INTRODUCTION AND NOTATION

One of the most important aspects of HELIOS is its geometric flexibility. The concept of a regular lattice does not exist any more: the pins can be positioned arbitrarily and geometric irregularities, like a water cross or water holes of any shape, can be treated without restriction. In fact, the calculations can be done on any number of assemblies or parts of them. The geometric build-up of the system to be calculated is described in chapter II (see also Casal, *et al.*, 1991).

The freedom to specify almost any two-dimensional (2D) system requires an accurate and versatile transport method throughout the system. HELIOS uses the CCCP method of chapter IV. Its basis is the geometric independence of collision-probabilities[2] (CPs) evaluated with the method of Carlvik (1965) (see also Stamm'ler and Abbate, 1983). To avoid prohibitive run times, the CP method has been combined with the method of current coupling. At the User's discretion, a system of many regions can be partitioned into a number of smaller systems, the space elements, which are coupled by currents. This reduces run time, because the computational effort of evaluating CPs is proportional to the square of the number of regions.

Of course, the reduced run time spent on the CPs is partially offset by the additional time spent on the current coupling. However, unless a high-order angular discretization is used for the currents—the k of Fig. IV.1—this additional time is smaller than the run-time reduction of the CPs. Moreover, the utilization of transformation laws (see chapter IV.5) and geometric symmetries (see section 7), which avoid many of the expensive CP calculations, are usually only effective in small space elements.

In the CCCP method, the angular dependence of the interface currents between the space elements is discretized by partitioning the directional half-sphere into sectors. At each interface segment, the polar angle is divided into a number of θ levels, and each θ level into a number of equal azimuthal φ intervals. Fig. II.13 illustrates a case where all θ levels are subdivided into the same φ intervals, and Fig. IV.1 shows the currently available discretizations. In this chapter, the generic indexes k and l are used for the sectors. Within each angular sector k, the angular flux dependence is still assumed isotropic, i.e., $\Phi(\varphi,\theta) = \Phi_k$, and the incoming current is (see also Fig. 1)

$$j(\varphi,\theta) = \vec{\Omega}\Phi(\varphi,\theta) \bullet \vec{n} = \Phi_k \sin\varphi\cos\theta, \quad (\varphi,\theta) \in k \ . \tag{1}$$

Figure 1 shows the coordinates of Eq. (1) and the position of the 2D space element of Fig. 2. If the angular dependence of the in-current is described by a single sector, this corresponds to the well-known cosine-current approximation (although there is also a $\sin\varphi$ behaviour).

[1] The present chapter is a modified version of a publication by Villarino et al. (1992).

[2] It should rather be *first-flight probabilities* because transmission and escape probabilities are included, but for the sake of tradition the term *collision probabilities (CPs)* will be used too.

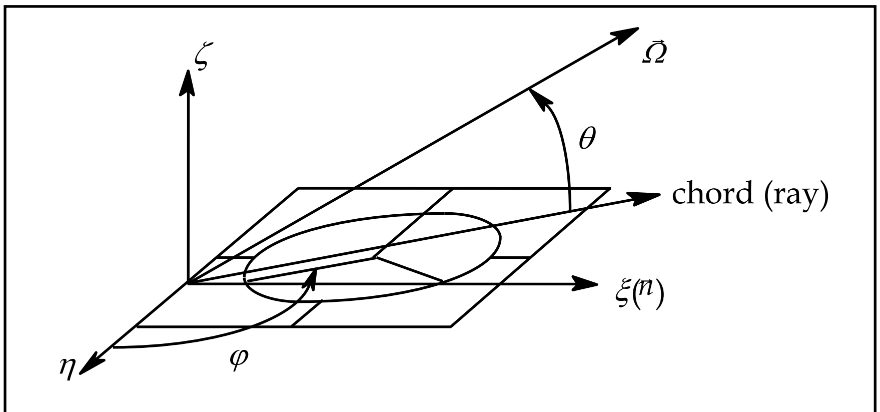


Fig. 1: Angular dependence of in-current in direction $\vec{\Omega}$.

This chapter is devoted to the evaluation of the CPs in a space element, the CCCP method is described in chapter IV. The 2D space element that is used to illustrate the derivation of the CPs is shown three times in Fig. 2. It is divided into homogeneous regions—the space meshes of the problem—inside which the flat-flux approximation is used. These regions are denoted by the generic indexes i and j. The surface is divided into straight-line segments on which the currents are assumed to be spatially flat. Their generic indexes are s and t. The discretization in angular sectors, indexed k or l, was already mentioned above.

The nomenclature used in this chapter for the space element is as follows:

<u>Regions:</u>	i, j	=	generic region indexes;
	I	=	total number of regions;
	V_i	=	volume of region i per 1 cm height;
	Σ_i	=	transport-corrected total macroscopic cross-section.
<u>Segments:</u>	s, t	=	generic segment indexes[3];
	S	=	total number of segments;
	A_s	=	surface of segment s per 1 cm height.
<u>Sectors:</u>	k, l	=	generic sector indexes; the azimuthal and polar intervals are: $(\varphi_m, \varphi_{m+1})$, and $(-\theta_{n+1}, -\theta_n)$ and (θ_n, θ_{n+1});
	$K(s)$	=	total number of sectors on segment s.

In section 2, formulae for the following CPs are derived:

- p_{ji} and $p_{sk,i}$, the probabilities of a particle that is born uniformly and isotropically in region i to suffer its first collision in region j, or to escape uncollided through sector k on side s;

[3] In this chapter s and t are segment indexes, while in chapter IV they were sector indexes (here k and l are sector indexes). That is because here the sectors are considered per peripheral segment of the space element, while in chapter IV they were considered per space element.

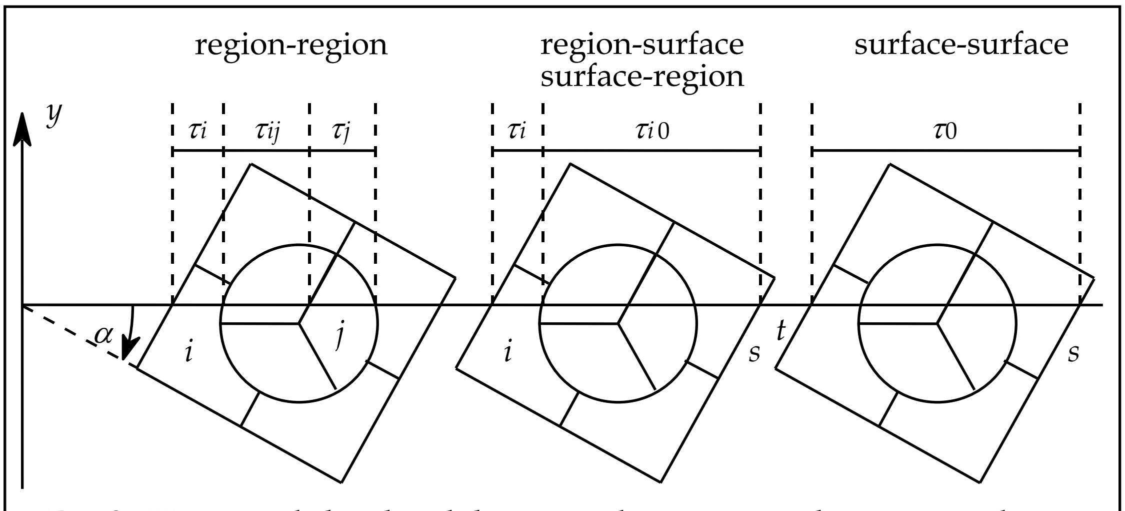


Fig. 2: Horizontal chord and the space element rotated over an angle α.

- $p_{i,sk}$ and $p_{tl,sk}$, the probabilities of a particle entering through sector k of side s to suffer its first collision in region i, or to escape uncollided through sector l on side t.

The reciprocity relation between $p_{sk,i}$ and $p_{i,sk}$ is also obtained.

Sections 3 and 4 present an integration scheme and a normalization method that give stable probabilities. The selection criterion for the angular sectors is described in section 5. The partition of the polar angle into θ levels has required the introduction of partial Bickley functions. Section 6 describes a computer-efficient evaluation of these functions. Section 7, finally, shows how the symmetries of a space element are obtained.

V. 2 FIRST-FLIGHT PROBABILITIES

The angular dependence of the interface currents causes angular dependence of the CPs. In this section, their formulae are derived. First, two expressions are derived for the probability of particles that are emitted from two different line sources to travel, in horizontal projection—in the plane of the space element—t cm or $\tau=\Sigma t$ mean free paths (mfp). One line source is isotropic, the other has the $\sin\varphi\cos\theta$ dependence of Eq. (1). The first expression is used to obtain CPs of particles born isotropically inside a region; the second is used to obtain CPs of in-currents entering through the angular sector of a segment.

V. 2.1 Isotropic line source (per slice dφ)

For an isotropic line source perpendicular to the plane of the 2D space element, the angular emission density normalized to one particle is $1/4\pi$. The probability of particles to be emitted in the differential solid angle $\cos\theta d\theta d\varphi$ and to travel uncollided

τ mfp in the plane of the space element is then (Carlvik, 1965; Stamm'ler and Abbate, 1983)

$$p(\tau,\theta,\varphi)=\frac{\cos\theta\mathrm{d}\theta\mathrm{d}\varphi}{4\pi}\exp(-\tau/\cos\theta)\ . \tag{2}$$

Observe that $p(\tau,\theta,\varphi)=p(\tau,-\theta,\varphi)$. Thus, the same probability of particles emitted in the double θ interval $(-\theta_{n+1},-\theta_n)$ and (θ_n,θ_{n+1}) of the azimuthal slice $\mathrm{d}\varphi$, now normalized to one particle emitted in that slice, is

$$p(\tau,\theta_n,\theta_{n+1},\varphi)=ki_2(\tau,\theta_{n+1})-ki_2(\tau,\theta_n)\ . \tag{3}$$

In Eq. (3), the partial Bickley function $ki_2(\tau,\theta)$ is introduced; its integration is not from 0 to $\pi/2$ but from 0 to θ:

$$ki_n(\tau,\theta)=\int_0^{\theta}\exp(-\tau/\cos\theta)\cos^{\mathrm{n-1}}\theta\mathrm{d}\theta\ . \tag{4}$$

For $\theta=\pi/2$, the partial Bickley functions simplify to the standard Bickley functions (Bickley and Nayler, 1935), $ki_n(t,\pi/2)=Ki_n(\tau)$. Hence, setting $\theta_n=0$ and $\theta_{n+1}=\pi/2$ yields the familiar result (Carlvik, 1965; Stamm'ler and Abbate, 1983)

$$p(\tau,\varphi)=Ki_2(\tau)\ . \tag{5}$$

V. 2.2 Angularly-dependent line source (per slice dφ of sector k)

For a line source with angular dependence $\sin\varphi\cos\theta$, the in-current normalized to one particle entering the space element is $(1/\pi)\sin\varphi\cos\theta$. The probability of entering in the differential solid angle $\cos\theta\mathrm{d}\theta\mathrm{d}\varphi$ and travelling uncollided τ mfp is then

$$\gamma(\tau,\theta,\varphi)=\frac{\sin\varphi\cos^2\theta\mathrm{d}\theta\mathrm{d}\varphi}{\pi}\exp(-\tau/\cos\theta)\ . \tag{6}$$

Observe that $\gamma(\tau,\theta,\varphi)=\gamma(\tau,-\theta,\varphi)$. Thus, the same probability of particles entering in the double θ interval $(-\theta_{n+1},-\theta_n)$ and (θ_n,θ_{n+1}) of the azimuthal slice $\mathrm{d}\varphi$, but normalized to one particle emitted inside this double θ interval of the slice, is

$$\left.\begin{aligned}\gamma(\tau,\theta_n,\theta_{n+1},\varphi)&=\frac{2\sin\varphi\mathrm{d}\varphi\left[ki_3(\tau,\theta_{n+1})-ki_3(\tau,\theta_n)\right]}{\pi k_1(\varphi,\varphi+\mathrm{d}\varphi)k_2(\theta_n,\theta_{n+1})}\\&=\frac{2\left[ki_3(\tau,\theta_{n+1})-ki_3(\tau,\theta_n)\right]}{\pi k_2(\theta_n,\theta_{n+1})}\end{aligned}\right\}\ . \tag{7}$$

In Eq. (7), the two normalization factors for the φ and θ integrations, k_1 and k_2, are

$$k_1(\varphi_m,\varphi_{m+1}) = \cos\varphi_m - \cos\varphi_{m+1} \tag{8}$$

and

$$k_2(\theta_n,\theta_{n+1}) = \frac{2\theta_{n+1} - 2\theta_n + \sin 2\theta_{n+1} - \sin 2\theta_n}{2\pi} \ . \tag{9}$$

Observe that with $\varphi_m=\varphi$ and $\varphi_{m+1}=\varphi+\mathrm{d}\varphi$, the φ normalization of Eq. (8) reduces to $\sin\varphi\,\mathrm{d}\varphi$. Further, if $\theta_n=0$ and $\theta_{n+1}=\pi/2$, $k_2=1/2$, which leads to the familiar result (Carlvik, 1965)

$$\gamma(\tau,\varphi) = \frac{4}{\pi}Ki_3(\tau) \ . \tag{10}$$

V. 2.3 Region-to-region collision probabilities

Because sources and scattering are assumed to be isotropic, the region-to-region collision probabilities have no angular dependence. They are defined as

p_{ji}	=	probability of particles born uniformly and isotropically in region i to suffer a first collision in region j.

The derivation that follows is well-known (Carlvik, 1965; Stamm'ler and Abbate, 1983; Villarino, 1989); it is repeated here to aid in the understanding of that of the other CPs. According to Fig. 2 and Eq. (5), particles emitted at t in region i—with volume V_i and total cross-section Σ_i—in the direction α have the following probability of suffering a first collision in region j:

$$p_{ji}(\tau,y,\alpha) = Ki_2\big(\Sigma_i(t_i - t) + \tau_{ij}\big) - Ki_2\big(\Sigma_i(t_i - t) + \tau_{ij} + \tau_j\big) \ . \tag{11}$$

To obtain the same probability for particles emitted uniformly along the length of the chord inside region i, Eq. (11) has to be integrated along t_i. With the emission density $1/t_i$ normalized to one particle born uniformly along t_i, this yields

$$p_{ji}(y,\alpha) = \frac{Ki_3(\tau_{ij}) - Ki_3(\tau_{ij}+\tau_i) - Ki_3(\tau_{ij}+\tau_j) + Ki_3(\tau_{ij}+\tau_i+\tau_j)}{\tau_i} \ . \tag{12}$$

The τ's depend on y and α, so the total p_{ji} is the double integral of $p_{ji}(y,\alpha)$ over the relevant values of y and α. Then, after multiplication by $t_i\mathrm{d}y/V_i$ and $\mathrm{d}\alpha/2\pi$, to normalize it to one particle emitted inside V_i, the final expression for p_{ji} becomes

$$p_{ji} = \frac{\int_0^{2\pi}\int_{y_{\min}}^{y_{\max}}\left[Ki_3(\tau_{ij}) - Ki_3(\tau_{ij}+\tau_i) - Ki_3(\tau_{ij}+\tau_j) + Ki_3(\tau_{ij}+\tau_i+\tau_j)\right]\mathrm{d}y\,\mathrm{d}\alpha}{2\pi\Sigma_i V_i} \ . \tag{13}$$

Note that the y interval ($y_{min} \le y \le y_{max}$) depends on the integration angle α. It is determined by the y coordinates of the lowest and highest vertices of the element at that angle. Because the chords for all the y-values do not pass through the regions i and/or j, parts of the y interval do not contribute to p_{ji}.

The symmetric integrand in Eq. (13) implies the reciprocity relation

$$\Sigma_i V_i p_{ji} = \Sigma_j V_j p_{ij} \ . \tag{14}$$

The expression for the self-collision probability is readily found (Carlvik, 1965; Stamm'ler and Abbate, 1983):

$$p_{ii} = 1 - \frac{\int_0^{2\pi} \int_{y_{\min}}^{y_{\max}} \left[Ki_3(0) - Ki_3(\tau_i)\right] \mathrm{d}y \, \mathrm{d}\alpha}{2\pi \Sigma_i V_i} \ . \tag{15}$$

V. 2.4 Region-to-sector escape probabilities

The region-to-surface escape probabilities are defined as

$p_{sk,i}$	=	the probability of particles born uniformly and isotropically in region i to escape the space element uncollided through sector k of segment s.

According to Fig. 2 and Eq. (3), particles emitted at τ in region i in the direction of α and in the double θ interval ($-\theta_{n+1}, -\theta_n$) and (θ_n, θ_{n+1}), have the following probability to escape from the element through segment s:

$$p_{sk,i}(\tau, y, \alpha, \theta_n, \theta_{n+1}) = \Delta ki_2\left(\Sigma_i (t_i - t) + \tau_{i0}, \theta_n, \theta_{n+1}\right) \ , \tag{16}$$

where

$$\Delta ki_2\left(\tau, \theta_n, \theta_{n+1}\right) = ki_2\left(\tau, \theta_{n+1}\right) - ki_2\left(\tau, \theta_n\right) \ . \tag{17}$$

The final expression for $p_{sk,i}$ is found in the same way as Eq. (13) for p_{ji}. First, integrate along t_i, with the source density $1/t_i$. Next, perform the double integration over y and α, with the source density $\mathrm{d}y\mathrm{d}\alpha / 2\pi V_i$, and define Δki_3 analogous to Δki_2. This yields

$$p_{sk,i} = \frac{\int_{\alpha(\varphi_m)}^{\alpha(\varphi_{m+1})} \int_{y_{\min}}^{y_{\max}} \left[\Delta ki_3\left(\tau_{i0}, \theta_n, \theta_{n+1}\right) - \Delta ki_3\left(\tau_{i0} + \tau_i, \theta_n, \theta_{n+1}\right)\right] \mathrm{d}y \, \mathrm{d}\alpha}{2\pi \Sigma_i V_i} \ . \tag{18}$$

The α integration is from $\alpha(\varphi_m)$ to $\alpha(\varphi_{m+1})$, the azimuthal interval that corresponds to sector k at segment s. (The θ interval corresponds to the polar interval of k.) The y interval, $y_{max} - y_{min} = A_s \sin\varphi$, causes the $\sin\varphi$ behaviour of the in-currents. Again, not all the chords will pass through region i and/or the sector sk.

Because a particle born in region i must either collide in the element or escape from it, the following balance can be formulated:

$$\sum_{j=1}^{I} p_{ji} + \sum_{s=1}^{S} \sum_{k=1}^{K(s)} p_{sk,i} = 1 \ . \tag{19}$$

V. 2.5 Sector-to-region collision probabilities

The sector-to-region probabilities are the partial blacknesses, defined as

$p_{i,sk}$ = the probability of particles entering the space element in sector k of segment s, with angular dependence $\sin\varphi\cos\theta$ (isotropic flux), to suffer a first collision in region i.

According to Fig. 2 and Eq. (7), particles entering through segment s in a slice dφ of sector k have the following probability to suffer a first collision in region i:

$$p_{i,sk}(y,\varphi) = \frac{2\left[\Delta ki_3(\tau_{i0},\theta_n,\theta_{n+1}) - \Delta ki_3(\tau_{i0}+\tau_i,\theta_n,\theta_{n+1})\right]}{\pi k_2(\theta_n,\theta_{n+1})} \ . \tag{20}$$

The total $p_{i,sk}$ is obtained by integrating over y and φ, with the source density $\mathrm{d}y\mathrm{d}\varphi/A_s k_1(\varphi_m,\varphi_{m+1})$ normalized to one particle [see also Eq. (8)]. The limits of y are 0 and $A_s\sin j$, those of φ are φ_m and φ_{m+1}. The final expression for $p_{i,sk}$ is then

$$p_{i,sk} = \frac{2\int_{\varphi_m}^{\varphi_{m+1}}\int_0^{A_s\sin\varphi}\left[\Delta ki_3(\tau_{i0},\theta_n,\theta_{n+1}) - \Delta ki_3(\tau_{i0}+\tau_i,\theta_n,\theta_{n+1})\right]\mathrm{d}y\,\mathrm{d}\varphi}{\pi A_{sk}} \ . \tag{21}$$

Here too, parts of the y-interval may not contribute to $p_{i,sk}$. A_{sk} can be interpreted as the "sector surface"; it is the surface A_s weighted with its fraction of the total in-current entering through the segment, i.e.,

$$A_{sk} = A_s k_1(\varphi_m,\varphi_{m+1}) k_2(\theta_n,\theta_{n+1}) \ . \tag{22}$$

Comparison of Eq. (21) with Eq. (18) shows the following reciprocity relation between partial blackness and escape probability:

$$p_{i,sk} = \frac{4V_i\Sigma_i}{A_{sk}} p_{sk,i} \ . \tag{23}$$

Note that if segment s has only one sector ($\theta_1 = 0$, $\theta_2 = \pi/2$ and $\varphi_1 = 0$, $\varphi_2 = \pi$), $A_{sk} = A_s$ and Eq. (23) reduces to a well-known reciprocity relation (Stamm'ler and Abbate, 1983).

V. 2.6 Sector-to-sector transmission probabilities

The sector-to-sector transmission probabilities are defined as

$p_{sk,tl}$ = the probability of particles entering the space element in sector l of segment t, with angular dependence $\sin\varphi\cos\theta$ (isotropic flux), to leave the element uncollided through sector k of segment s.

According to Fig. 2 and Eq. (7), particles entering through segment t in a slice dφ of sector l have the following probability to escape through sector k of segment s:

$$p_{sk,tl}(y,\varphi) = \frac{2\left\lfloor ki_3(\tau_0,\theta^*_{n+1}) - ki_3(\tau_0,\theta^*_n)\right\rfloor}{\pi k_2(\theta_n,\theta_{n+1})} \ , \tag{24}$$

where the θ* limits of the exiting interval, at sk, are given by

$$\left.\begin{aligned} \theta^*_n &= \max(\theta_{n,tl},\theta_{n,sk}), & \theta_n &= \theta_{n,tl} \\ \theta^*_{n+1} &= \min(\theta_{n+1,tl},\theta_{n+1,sk}), & \theta_{n+1} &= \theta_{n+1,tl} \end{aligned}\right\} . \tag{25}$$

$p_{sk,tl}$ can be obtained by integrating Eq. (24) over y and φ, with the source density $\mathrm{d}y\mathrm{d}\varphi/A_s k_1(\varphi_m,\varphi_{m+1})$ normalized to one particle [see also Eq. (8)]. Observe that $\varphi_m = \varphi_{m,tl}$ and $\varphi_{m+1} = \varphi_{m+1,tl}$; i.e., they belong to the segment of entry, tl:

$$p_{sk,tl} = \frac{2\int_{\varphi^*_m}^{\varphi^*_{m+1}}\int_0^{A_t\sin\varphi}\left[ki_3(\tau_0,\theta^*_{n+1}) - ki_3(\tau_0,\theta^*_n)\right]\mathrm{d}y\,\mathrm{d}\varphi}{\pi A_{tl}} \ . \tag{26}$$

The φ interval is restricted in a similar way as the θ interval, Eq. (25). However, the angle between the inward normal on t and the outward normal on s must be subtracted from $\varphi_{m,sk}$ and $\varphi_{m+1,sk}$. In fact, the y interval is limited, too; there are angles φ for which segment s cannot be seen from some or all points on segment t.

The numerator of Eq. (26) is the same for transmission in the other direction, i.e., for $p_{tl,sk}$. Thus, there exists a reciprocity relation between the transmissions:

$$A_{tl}\,p_{sk,tl} = A_{sk}\,p_{tl,sk} \ . \tag{27}$$

Because particles entering the space element either collide inside the element or escape through one of its sectors, the following balance can be formulated:

$$\sum_{i=1}^{I} p_{i,sk} + \sum_{t=1}^{S}\sum_{l=1}^{K(t)} p_{tl,sk} = 1 \ . \tag{28}$$

V. 3 INTEGRATION BY RAY TRACING

The double integrals over y and α (or φ) in the CPs are evaluated numerically. The space element is considered under different angles of rotation, the integration angles α_p (α in Fig. 2). For each angle a set of parallel, horizontal rays (or chords) are drawn across the element, the integration chords $y = y_q$. With w_p and w_{qp} being the weights of the angles and chords the numerical approximation is

$$\int\int f(y,\alpha)\mathrm{d}y\mathrm{d}\alpha = \sum_p \sum_q w_p w_{qp} f(y_{qp},\alpha_p) \ . \tag{29}$$

In HELIOS two integration schemes are applied, volume and surface integration. The first deals with particles born in the regions; it evaluates the region-to-region probabilities. Per energy group, this is done once for all the regions together. The second deals with particles entering through the segments; it evaluates the sector-to-region/sector probabilities—the escape probabilities then are obtained from Eq. (23). Per group, this is done for each surface segment that has no specular reflection.

This separation into two schemes is because the surface integration needs a finer angular-integration mesh than the volume integration. Both schemes are summarized below; basically, both use Carlvik's method (1965). Their accuracy is determined by two parameters: the number of chords per cm and the number of angles per π radians.

V. 3.1 Angular discretization

The integration interval $(0-\alpha_{sym})$ is established. Its upper limit α_{sym} depends on whether volume or surface integration is done (see sections 3.4 and 3.5). This interval is divided into a set of subintervals, indexed by μ. Finally, each subinterval is divided into N_μ equidistant angles, α_p. With $\Delta\alpha_\mu$ the width of the μ'th interval, the weights of its integration angles are

$$w_p = \Delta\alpha_\mu / N_\mu, \quad p \in \mu \ . \tag{30}$$

V. 3.2 Discretization of y

For each α_p, the element is rotated, keeping the chords horizontal. This facilitates finding the chord intersections with the boundaries of the regions. The y interval is divided into macrobands, which are essential for the stability of the numerical integration scheme (Villarino *et al.*, 1992). The y coordinates of the macrobands are defined by the vertices of all the segments outside the circular systems, the midpoints of the circular systems, and the tangential points of all the radial zones of the circular systems. In these points, $f(y,\alpha)$ has a discontinuous first derivative.

The chords inside a macroband cross the same regions, except regions caused by azimuthal subdivisions of circular systems—if present, they do not present strong discontinuities. Figure 3 shows the macrobands of a space element and the subset of macrobands of its segment s.

Inside each macroband, indexed by ν, N_ν equidistant chords are drawn, y_q. Each chord represents a strip of the macroband. With $\Delta y_{p\nu}$ the width of the νth macroband at angle α_p, its chord weights are the widths of its strips,

$$w_{qp} = \Delta y_{p\nu} / N_\nu , \quad q \in \nu \ . \tag{31}$$

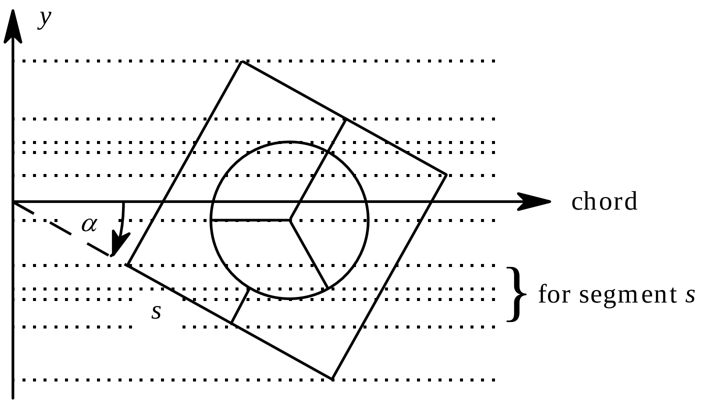


Fig. 3: Macrobands in chord integration.

Mobile chords are used, instead of fixed chords. This means that the relative positions of the chords are not fixed in the centres of their strips, but vary with the integration angle. If $0 \le r_p \le 1$ denotes the relative position in the strips for the angle φ_p and the total number of integration angles is P, then r_p is

$$\left.\begin{array}{ll} r_p = p/2P, & p = odd \\ r_p = 1 - r_{p-1}, & p = even \end{array}\right\} . \tag{32}$$

Carlvik (1965) used mobile chords in CLUCOP (see the source code). They improve the integration accuracy but have the disadvantage that the results for a space element depend slightly on its input orientation (Villarino *et al.*, 1992). Since even simple elements can have many narrow macrobands (see Fig. 3) with just one chord, Gauss integration is of little use.

Because the number of macrobands may become very large, their width has a lower limit. Currently that limit, which can be changed by the User, is 0.001 cm. Also, macrobands are eliminated if there are more than three times the number of chords (if the chords were equidistant). Macrobands caused by the vertices of a peripheral segment, or by the innermost radius of a circular system, are eliminated last.

Finally, the intersections of each chord with the region boundaries are obtained and the region thicknesses along the chord, t_i, are registered. This geometric process is independent of the energy groups and is done only once per case as part of the input processing.

V. 3.3 Boundary conditions

At each surface segment, several types of boundary condition (BC) are allowed (see chapter II.4 and IV.3.2). However, the CP integration schemes can handle only free (vacuum) and specular BCs. All the equations shown are for free BCs; the particles leave the space element once they reach a segment—after which they are taken care of by the current coupling. The specular BC, which can only be treated at CP level, are recommended in systems where some of the fuel pins are only represented by a symmetric fraction. The segments along which the pins have been cut off are almost certainly boundary segments of the system. In general, the other system boundaries will not have specular BCs.

The specular BC differs mainly in the calculation of the integration chords. The chords are reflected at the specular segments and continued until they reach a free segment and leave the element. Since the ray tracing is part of the input processing, it is not possible to guarantee that the particles will have flight distances of at most λ mfp (in HELIOS, $\lambda = 6$). Instead, the chords have a preset number of 250 reflections or a maximum length of 500 cm. If this leads to shorter flight paths than 6 mfp, a warning is given but the calculations continue. This will hardly happen in practice because, in general, not all the segments of a space element are specular.

V. 3.4 Volume integration

In the region integration, there is only one φ interval. If φ_{sym} is the symmetry angle of the space element (see section 7), then the element is rotated over a reference angle such that the integration interval becomes $(0 - \varphi_{sym})$. If there is no symmetry, $\varphi_{sym} = \pi$ (each chord represents particles travelling in positive and negative directions).

The numerically integrated region volumes are

$$V_i^* = \frac{1}{\varphi_{sym}} \sum_p \sum_q w_p w_{qp} t_{i(qp)} \quad . \tag{33}$$

The ratio between the true and the numerically integrated volume is a measure of the integration accuracy. If the maximum relative error of any volume exceeds a preset tolerance, one more ray tracing is done with four more angles per π radians and a 1.5 times reduced maximum chord spacing inside each macroband. Presently, the initial number of angles per π radians is 16 and the initial maximum chord spacing inside each macroband is 0.01 cm. Of course, the User may modify both these numbers.

Further stability of the CPs is obtained (Villarino *et al.*, 1992) by normalization of the region thicknesses, as is done in WIMS (Askew *et al.*, 1966) (see the source code) and CONDOR (Villarino, 1990). This is done as follows:

$$t_{i(qp)}^* = (V_i / V_i^*) t_{i(qp)}, \quad \forall i(qp) \quad . \tag{34}$$

V. 3.5 Surface integration

Surface integration is done for each segment that has no specular reflection. The integration is again from 0 to φ_{sym}, where φ_{sym} is π, or $\pi/2$ if the system is symmetric with respect to the normal through the midpoint of the segment (see section 7). This integration interval is divided into macroangles whose vertices coincide with those of the segment—the left and right vertices of segment *s* in Fig. 4.

As can be seen in Fig. 4, the macroangle sides pass through the vertices of all the segments outside the CCS-es (circular cylindrical systems) and through the midpoints and external tangential points of the CCS-es. Moreover, the φ_m of the sectors on segment *s* also define macroangles (not shown in Fig. 4).

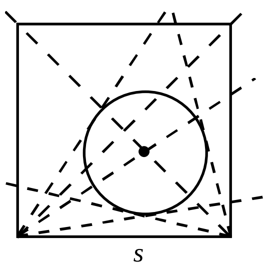


Fig. 4: Macroangles of segment s.

The macroangles have a lower limit, which depends on the type of macroangle. For macroangles due to internal segments and CCS points, that limit is the smallest of $\pi/36$ and π/N_{surf} radians. The latter is the maximum angular subinterval inside each macroangle, defined by N_{surf}. Currently, the default is N_{surf}= 24, which the User may change. For macroangles caused by peripheral segment vertices, that limit is the smallest of $\pi/180$ and π/N_{surf} radians. Macroangles caused by the azimuthal φ_m intervals on the directional half-sphere are always enforced. Finally, macroangles are eliminated if there are more than three times the number of integration angles (if they were equidistant, i.e., $3*N_{surf}$), except those due to the azimuthal φ_m intervals.

The y integration is performed from y_{min} to y_{max}, in much the same way as in the volume integration. However, the concept of mobile chords is *not* used; occasionally, it may produce very inaccurate transmission probabilities. Otherwise, the use of both macroangles and macrobands has greatly enhanced the stability and accuracy of the transmissions and partial blacknesses.

The numerically integrated azimuthal sector corresponding to $(\varphi_m, \varphi_{m+1})$ of the segment *s* under consideration is

$$k^*_{1s}(\varphi_m, \varphi_{m+1}) = \frac{1}{A_s}\sum_p \sum_q w_p w_{qp} \quad . \tag{35}$$

The ratio between the true and the integrated azimuthal sector is a measure of the integration accuracy. If the maximum relative error of any azimuthal sector exceeds a preset tolerance, one more ray tracing is done with four more angles per π radians

($N_{surf}=N_{surf}+4$) and 1.5 times smaller maximum chord spacing inside each macroband—this has seldom been observed. Currently, the initial value of N_{surf} is 24 and the initial maximum chord spacing inside each macroband is 0.01 cm. The User may modify both these numbers.

Further stability of the CPs is obtained by normalizing the angular weights,

$$w^*_{p(sm)} = \frac{k_{1s}(\varphi_m, \varphi_{m+1})}{k^*_{1s}(\varphi_m, \varphi_{m+1})} w_{p(sm)}, \quad \forall p(sm) \ . \tag{36}$$

V. 4 NORMALIZATION

In the resonance transport calculations, small σ_m values in Eq. (III.42) can lead to situations where the source is nearly equal to the macroscopic removal XS, ($Q \approx \Sigma_r$). To avoid numerical problems in Eq. (III.39), the flux must approximate 1 from below. The CCCP method does not produce this behaviour unless the reciprocities and balances are satisfied [Eqs (14), (19), (23), (27) and (28)]. This is achieved by normalizing the probabilities as described below.

First, the reciprocities are satisfied. Because Eq. (14) is used to reduce the angular integration interval ($0-2\pi$) to ($0-\pi$), it is implicit in the p_{ji}. Equation (23) is used to obtain escape probabilities from partial blacknesses, i.e., $p_{sk,i}$ from $p_{i,sk}$. Equation (27) is enforced by averaging the transmissions according to

$$A_{tl} p_{sk,tl} = \tfrac{1}{2}(A_{tl} p'_{sk,tl} + A_{sk} p'_{tl,sk}) = A_{sk} p_{tl,sk} \ , \tag{37}$$

where the primed transmissions are those obtained from the ray tracing. Notice that Eq. (27) is not used to obtain $p_{tl,sk}$ from $p_{sk,tl}$ because the surface integration, which is done for each segment individually, gives more accurate transmissions to the other segments than Eq. (27). This is one of the reasons why volume and surface integrations are separated.

Next, the balances must be enforced, for which several methods exist (Roy and Marleau, 1990; Gelbard, 1985). In HELIOS, a variant of the multiplication method of Roy and Marleau is used. It is based on symmetric correction factors. The symmetry preserves already existing reciprocities, and by using factors, quantities that are zero remain zero, e.g., the self-transmission $p_{sk,sk}$. If a tilde (~) denotes uncorrected values, the chosen correction is

$$q_{ab} = (w_a + w_b)\tilde{q}_{ab} \ . \tag{38}$$

In Eq. (38), the generic probabilities q_{ab} represent

$$\left.\begin{array}{llll} q_{ab} = p_{ij} & \text{for} & a = i \le I, & b = j \le I \\ q_{ab} = p_{i,sk} & \text{for} & a = i \le I, & b = I + sk \le A \\ q_{ab} = p_{sk,i} & \text{for} & a = I + sk \le A, & b = i \le I \\ q_{ab} = p_{sk,tl} & \text{for} & a = I + sk \le A, & b = I + tl \le A \end{array}\right\} , \tag{39}$$

where *sk* and *tl* are generic sector indexes that run through all the sectors on all the segments, i.e., from $I+1$ to A, with A given by

$$A = I + \sum_{s=1}^{S} K(s) \ . \tag{40}$$

The w components are solved iteratively from the balances, Eqs (19) and (28), i.e., from the equation system

$$\sum_{a=1}^{A} (w_a + w_b)\tilde{q}_{ab} = 1 \quad \Rightarrow \quad w_b = \frac{1 - \sum_{a \neq b} w_a \tilde{q}_{ab}}{\sum_a \tilde{q}_{ab} + \tilde{q}_{bb}} \ . \tag{41}$$

All w's get the start value of 0.5, the value they have if there were no errors in the probabilities. The iteration process uses a relaxation factor of 0.7. The iterations are stopped when all w's of two consecutive iterands differ by less than 10^{-5} or after a preset number of iterations, currently 14. The resulting correction factors are equal to those of the method of Roy and Marleau (1990), who use products $f_a f_b$ instead of sums $w_a + w_b$. Of course, the User may change the tolerance, the relaxation factor, and the maximum number of iterations.

To ensure that the reciprocities and balances are rigorously satisfied—maybe the w iterations did not converge, or the accuracy of 10^{-5} is insufficient—a final normalization is done. For each sector *sk,* the balance for one entering particle, Eq. (28), is enforced by multiplying the partial blacknesses by a factor, leaving the transmissions unaltered. Next, the escape probabilities are evaluated from their reciprocity with the partial blacknesses, Eq. (23). Finally, for each region *i,* the balance for one particle born, Eq. (19), is enforced by adjusting the self-collision probability, leaving the other probabilities unaltered.

Normally, the w iterations converge within 10 iterations. In that case, and also if they have almost converged after 10 iterations, this final normalization changes the affected probabilities only marginally. In all other cases, this final normalization has the weakness that the errors of the partial blacknesses are passed on via the escape probabilities to the self-collision probabilities, p_{ii}. Both normalizations—using $w_a + w_b$ or adjusting p_{ii}—have the same preset tolerance by which the correction factors may differ from 1. If that tolerance is exceeded (12%, which the User cannot change) a warning is printed and the User may opt to rerun the case with more integration chords and/or angles.

V. 5 CHOICE OF ANGULAR SECTORS

In the CCCP method, the directional half-sphere of incoming particles is divided into angular sectors. Thus, the polar angle θ is divided into N levels, each of which is divided into N_n azimuthal intervals. In HELIOS, all the angular sectors on a segment s have the same "sector surface" A_{sk}, i.e.,

$$A_{sk} = A_s / K(s) \quad \text{with} \quad K(s) = \sum_{n=1}^{N} N_n \ . \tag{42}$$

From the definition of A_{sk} [Eqs (22), (8) and (9)] follows that the azimuthal weights are proportional to $\cos\varphi$ and the polar weights to $\int\cos^2\theta\, d\theta$. This leads to the following recursive formulae for the interval limits, φ_{mn} and θ_n:

$$\left.\begin{aligned} \varphi_{mn} &= \cos^{-1}\left(\cos\varphi_{m-1,n} - 2/N_n\right) \quad \text{with} \quad \varphi_0 = 0, \varphi_{N_n} = \pi \\ 2\theta_n + \sin 2\theta_n &= 2\theta_{n-1} + \sin 2\theta_{n-1} + \frac{N_n}{K(s)}\pi \quad \text{with} \quad \theta_0 = 0, \theta_N = \pi/2 \end{aligned}\right\} . \tag{43}$$

It should be noted that this choice of sectors is not invariant with respect to a $\pi/2$ rotation of η towards ζ (see Fig. II.13). While the azimuthal φ intervals are limited by longitudes on the directional half-sphere, the θ levels are limited by latitudes. The former have equal lengths, the latter become shorter near the poles. In slab geometry, a case with four φ intervals and one θ level will give different results than a case with one φ interval and two θ levels.

V. 6 EVALUATION OF (PARTIAL) BICKLEY FUNCTIONS

The partial Bickley function $ki_3(\tau,\theta)$, Eq. (4), occurs in the surface integrations if the space element has at least one segment with sectors with more than one polar level. Otherwise—and for volume integrations—only the Bickley function $Ki_3(\tau)$ is involved. Their evaluation largely determines the time needed to integrate the CPs. This section presents a fast method for their evaluation.

Because the function $ki_3(\tau,\theta)$ is continuous in both τ and θ, it was decided to approximate it by Chebyshev expansions (Press *et al.*, 1990) rearranged into polynomials in τ and θ:

$$ki_3(\tau,\theta) = \sum_{i=0}\sum_{j=0} a_{ij}\tau^i\theta^j \ . \tag{44}$$

The demands put on these expansions are: (i) prescribed maximum absolute and relative errors; (ii) $0 \le \tau \le 12$; (iii) $0 \le \theta \le \pi/2$; (iv) few terms, so several intervals for both variables; and (v) continuity at the interval limits. The first four demands are trivial to satisfy.

Continuity has been enforced by recalling that the Chebyshev expansion of order N is exact at the $N+1$ zeros of the $(N+1)$th Chebyshev polynomial, T_{N+1} (Press *et al.*, 1990). So Eq. (44) is applied in intervals that coincide with the first and the last zero of T_{N+1}, although the coefficients were obtained in wider intervals.

The polynomials of Eq. (44) can be used when the θ-limits are not known in advance. HELIOS, however, has a set of fixed sector divisions built-in, i.e., it has a set of fixed θ_n values (Fig. IV.1). A small auxiliary program GENKI3 has been written to generate the expansion coefficients for $ki_3(\tau,\theta_n)$ in many small τ intervals at fixed θ_n values. These ki_3 expansions are not only continuous, even at $\tau=0$, but they also have everywhere a negative first derivative. For HELIOS, coefficient tables for two-term linear expansions, $ki_3(\tau,\theta_n)\cong a_1+a_2\tau$, were evaluated in the interval $0\le\tau\le6$, with maximum absolute and relative errors of $2\cdot10^{-5}$ and 10^{-3}, respectively.

Apart from fast evaluations of ki_3 and Ki_3, the cpu time spent on their evaluation has been reduced in two more ways. First, the number of Ki_3 evaluations used for the region-to-region CPs has been optimized (Villarino, 1990; Weiss, 1990). According to Eqs (15) and (13), p_{ii} and p_{ji} need one and four Ki_3 values, respectively. However, Fig. 5 and Eq. (45) show that the four CPs share Ki_3 values and need nine, not 16, Ki_3 evaluations:

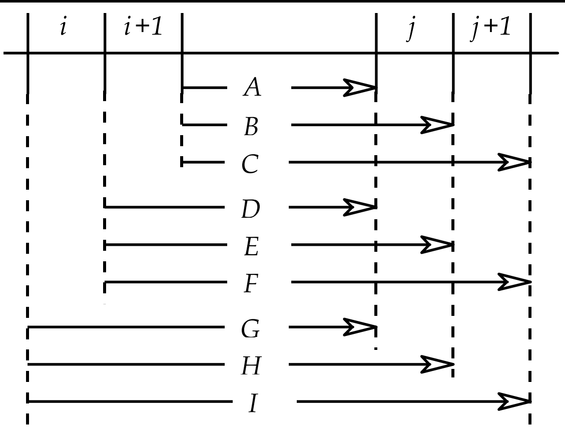


Fig. 5: Optimization of Ki_3 calls.

$$\left.\begin{aligned} p_{j,i} &= Ki_3(D)-Ki_3(G)-Ki_3(E)+Ki_3(H) \\ p_{j+1,i} &= Ki_3(E)-Ki_3(H)-Ki_3(F)+Ki_3(I) \\ p_{j,i+1} &= Ki_3(A)-Ki_3(D)-Ki_3(B)+Ki_3(E) \\ p_{j+1,i+1} &= Ki_3(B)-Ki_3(E)-Ki_3(C)+Ki_3(F) \end{aligned}\right\}. \tag{45}$$

When passing from j to $j+1$, the last two terms of p_{ji} occur again in $p_{j+1,i}$. Moreover, when passing from i to $i+1$, the first two terms—with D and E—occur again. In general, if a chord crosses R regions, the $R(R+1)/2$ involved CPs require $R(R+1)/2$ evaluations of Ki_3, and not $R(2R-1)$. For large R, this represents a saving of about a factor of four.

Second, the (partial) Bickley functions are not evaluated by a function subprogram. Instead, they are evaluated by three local statements, indicated by arrows in Fig. 6. The FORTRAN coding shown is that for the surface integration. For a given segment s and integration angle φ, the y-integration is represented by the 30-loop over the chords. For each chord there is the 20-loop over the intersections; and for each intersection the contributions to the θ levels are obtained in the 10-loop, which is where ki_3 is evaluated.

The interpolation tables INXTAU() and INXGAM(), the $A_0()$ and $A_1()$ expansion coefficients, and some other data not shown here, are generated by GENKI3. Its output is a BLOCK DATA subprogram that assigns these data to a COMMON block. Of course, for higher-order polynomials the other coefficient arrays, $A_n()$, are generated too.

```
C                                     ! For segment S.
C                                     ! IKI points at the ki3-shift table,
      IKI  = IKSEG(S)                 ! INXGAM(), at the first theta level.
C
      DO 30 NCH=1,NCHORD(S)           ! All the chords of segment S.
      TAU  = 0.0                      ! Initialize the chord to 0 mfp.
C
      DO 20 NX=1,NINTX(NCH)           ! All the intersections of NCH.
      TAU  = TAU + TREG(NX,NCH)       ! Travel TREG(NX) along chord NCH.
 -->  MIJ  = INXTAU(INT(FACTAU*TAU)+1) ! The argument is a TAU-integration interval;
C                                     ! INXTAU() is the coefficient position in
C                                     ! A0() and A1() for Ki3 (theta=pi/2).
      DO 10 NTI=1,NTH(S)              ! All the theta levels of segment S.
 -->  IA   = MIJ + INXGAM(IKI+NTI)    ! The second term is the shift in A0() and
C                                     ! A1(), relative to Ki3, for the theta level.
 -->  KI3  = A0(IA) + TAU*A1(IA)      ! The partial Bickley function!
```

Fig. 6: Coding to evaluate the partial Bickley function.

V. 7 SYMMETRY ANALYSIS

If, in the CCCP method, a space element has its own local symmetry, its CPs still must be evaluated for the entire element. That is because, in the global system of coupled space elements, it is not likely to have symmetric in-currents. Yet, significant run-time reductions can be accomplished if the symmetries of the space elements are utilized.

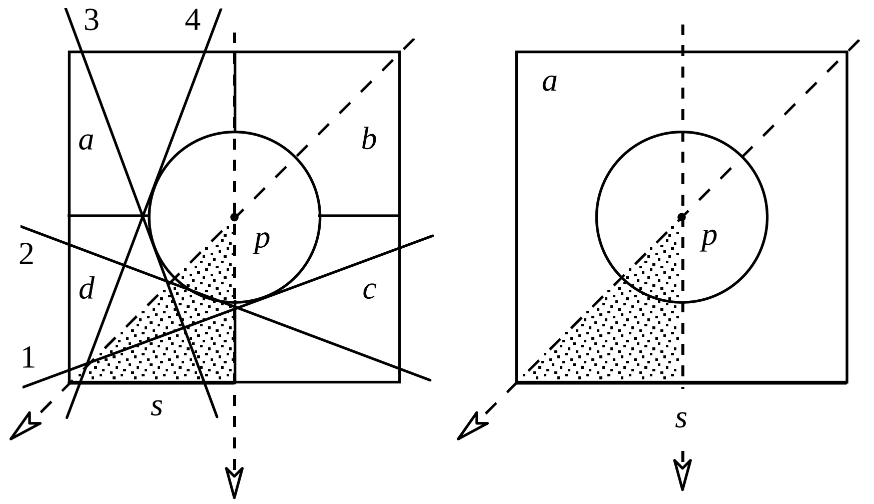


Fig. 7: Symmetry analysis of a space element.

HELIOS can detect the following symmetries: An element has *rotational* symmetry if it is the same after every rotation over its symmetry angle, the smallest possible angle—if that angle is 2π, the element is not symmetric. In addition, an element can have *specular* symmetry inside a rotational fraction. In that case the symmetry angle is half that of the rotational symmetry. Any other symmetry will not be detected.

Geometrically, both elements of Fig. 7 have $\pi/2$-rotational and specular symmetry. However, the element on the right is always symmetric, while the one on the left is symmetric only if the materials in the regions a, b, c and d, are the same. In the latter case, two sets of chords are saved. The reduced set is used if there is symmetry; if not, the complete set is used.

In the example on the left in Fig. 7, utilizing the reciprocity (14), there are five independent types (x) of p_{ji}. If the sequence of the regions is a, b, c, d, p, then the types of p_{ji} can be collected in the symmetric **T** matrix (type 5 corresponds to p_{pp}, type 4 to p_{pi} and p_{ip}, etc.):

$$\mathrm{T} = \begin{pmatrix} 1 & 2 & 3 & 2 & 4 \\ 2 & 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 4 \\ 2 & 3 & 2 & 1 & 4 \\ 4 & 4 & 4 & 4 & 5 \end{pmatrix} . \tag{46}$$

The volume integration is done with the reduced amount of chords, resulting in an incomplete p_{ji} matrix, **P**. With the aid of the type matrix **T**, the p_{ji} are evaluated per type x, and stored in an auxiliary array $X()$ as

$$X(x) = \frac{1}{N_x} \sum_{j=1}^{I} \sum_{i=1}^{I} \delta(T_{ji} - x) p_{ji} \ , \tag{47}$$

where N_x is the number of p_{ji} of type x, and $\delta(t-x)=1$ if $t=x$ and 0 otherwise. Then, using **T** again, the final p_{ji} matrix is obtained.

Similar methods are used for the surface integration. It should still be remarked that, in both space elements of Fig. 7, only segment s has to be considered. In the left element, the symmetry angle is $\varphi_{sym}=\pi$; in the right element, the segment s is twice as long but the symmetry angle can be halved, $j_{sym}=\pi/2$.

V. 8 SOME REMARKS ON THE PROGRAM

The evaluation of the first-flight probabilities involves three modules. Two of these modules, LINK10 and LINK11, are executed only once, at the start of a case. They perform the geometrical analysis as part of the input treatment of HELIOS. This analysis, which is group-independent, is done once for each (geometric) type of space element. The third module, LINK15, evaluates first-flight probabilities as part of the preparations for the transport calculation.

V. 8.1 Notes on LINK10

LINK10 deals with the symmetries of the space-element types. Initially, the symmetry is set to twice the number of sides of the element, assuming full periodicity and specular symmetry inside each periodic part. Then, it is tested for violation of this symmetry; if it is, the symmetry is reduced by one and the tests are done again until symmetry is found—a symmetry of one means no symmetry. Apart from the main administrative subroutine, MAIN10, and some general subroutines, the following subroutines are used:

FINDSY:	It controls, per element type, the determination of its symmetries for the volume and the surface integrations, and constructs its **T** matrices [Eq. (46)]. The two symmetries may differ because in the latter the angles of the angular current discretization are considered too. It calls, directly or indirectly, the routines mentioned below.
CALCEN:	It finds information for the symmetry search, like the rotation centre of the element and the midpoints of the peripheral segments.
PRESYM:	It finds more such information, like the locations of the CCS-es and the non-CCS regions, and the basic angles of CCS centres and of segment endpoints with respect to the rotation centre.
TESTSY:	It tests if the assumed symmetry of the element holds for these basic angles, the CCS-es, the segments, and the regions. This is done by calling CHKMAC, CHKCCS, CHKSEG and CHKREG.
PIJPAR:	It sets up the **T** matrix for the region-to-region probabilities.
CHKSEC:	It determines the number of peripheral-segment types, according to their location, angle, and angular-discretization sectors.
CHKSLF:	It checks the symmetric segments for specular symmetry with respect to the normals in their midpoints; see the remark after Eq. (47).
GAMPAR:	It sets up the **T** matrix for the partial blacknesses.
TRMPAR:	It sets up the **T** matrix for the first-flight transmissions.
VERSYM:	It protects against a possible loss of numerical accuracy by verifying that the number of independent probability types in **T** corresponds to the symmetry found. If not, a message is printed and the symmetry search continues.

V. 8.2 Notes on LINK11

LINK11 performs the ray tracing described in section 3. Apart from the main administrative subroutine, MAIN11, and some general subroutines, the following subroutines are used:

PRECOL:	It prepares data for the ray tracing of all the element types, like: the maximum number of intersections per chord; the "sector surfaces," A_{sk}, of Eq. (42); and segment identifiers which are equal for parallel segments. (If, within a macroband, more than one ray intersects only parallel segments, they are replaced by one ray to save cpu time.)
CHORDX:	It controls the ray tracing for the volume integration and calls, per element type, the subroutines PREROT, ANGROT, BANDIT, INTSEC, PACKDX, TSTVOL and NORCHD. These subroutines are briefly discussed below.
CHORDY:	It controls the ray tracing for the surface integration and calls, per element type, the subroutines PREROT, ANGSUR, ANGROT, BANDIT, INTSEC, PACKDX and TSTSUR.
PREROT:	It extracts from the global arrays (for all the element types) local arrays for the actual element type.
ANGROT:	It rotates the element to an integration angle α_p; see Eq. (29).
BANDIT:	This subroutine is used in the volume integration. It finds, for the element whose first-flight probabilities are to be integrated, the macrobands defined in section 3.2. It calls the subroutines APPBND and DELPNT.
ANGSUR:	This subroutine is used in the surface integration. It finds, for the segment whose first-flight probabilities are to be integrated, the macroangles defined in section 3.5. It calls the subroutines APPANG and DELPNT.
APPBND:	It checks if a possible macroband can be accepted as such, based on its size; see section 3.2.
APPANG:	It checks if a possible macroangle can be accepted as such, based on its size; see section 3.5.
DELPNT:	It limits the number of macrobands or macroangles, according to the criteria of sections 3.2 and 3.5.
INTSEC:	It finds the intersections of a chord with all the segments and arcs of the space element. It uses the integer function REGCCS.
REGCCS:	If the chord intersects one of the concentric radii of a CCS, it determines whether this is the outside radius; if so, it returns the sequence number of the CCS arc among the arcs and segments.
PACKDX:	It arranges chord-intersection data, like region thicknesses and region identifiers, in an efficient way for the later evaluation of first-flight probabilities. It calls the subroutine ANGSEC.
ANGSEC:	Given the angle of the chord and a boundary segment, it finds the corresponding angular sector on that segment.

TSTVOL: This subroutine is used in the volume integration. It tests that all the regions have been integrated by at least one chord, and it compares the integrated with the true volumes. If they differ by too much, a possible second integration is done, as described in section 3.4.

TSTSUR: This subroutine is used in the surface integration. It compares the integrated azimuthal sector of the segment under consideration, Eq. (35), with its true value. If they differ by too much, a possible second integration is done, as described in section 3.5.

NORCHD: It normalizes the region thicknesses of the volume integration according to Eq. (34).

ORDENA: This general subroutine orders the elements of an array according to increasing size. It is called by various routines.

V. 8.3 Notes on LINK15

LINK15 calculates first-flight probabilities, as well as multiple-flight response fluxes, and escape and transmission probabilities. In this section, the subroutines used to calculate the first-flight probabilities are outlined; the other subroutines are outlined in chapter IV.6.1.

COLRMD: This subroutine controls both types of calculation of LINK15; it is described in chapter IV.6.1.

COLPRO: This subroutine controls the calculation of the first-flight probabilities of a given space element in a given (energy) group. It calls all the subroutines described below, except the BLOCK DATA subprogram, KI3DAT.

CALPIJ: This subroutine calculates the region-to-region probabilities of a space element that may have one or more specular boundary segments. In fact, it is only called if there exist specular segments. Then, the pseudo-escape probabilities through these segments, the p_{si} discussed in chapter IV.3.2.2, are also calculated.

FREPIJ: This subroutine calculates region-to-region probabilities of an element without specular boundary segments. It is faster than CALPIJ.

CALGAM: It calculates the partial blacknesses and transmissions of a space element whose boundary segments may have an angular discretization with more than one polar level. Actually, it is only called if there are such segments. Pseudo-transmissions to possible specular segments, the p_{st} discussed in chapter IV.3.2.2, are also calculated.

TH1GAM: It calculates the partial blacknesses and transmissions of a space element whose boundary segments have an angular discretization of one polar level only. Pseudo transmissions to possible specular segments, the p_{st} discussed in chapter IV.3.2.2, are also calculated. It is faster than CALGAM.

CNVPIJ: If CALPIJ or FREPIJ have calculated a symmetric part of a space element, then this routine expands their probabilities to the full element; see section 7.

CNVGAM: If CALGAM or TH1GAM have calculated a symmetric part of a space element, then this routine expands their partial blacknesses to the full element; see section 7.

CNVTRM: If CALGAM or TH1GAM have calculated a symmetric part of a space element, then this routine expands their transmissions to the full element; see section 7.

VERIFY: This subroutine is called if the User has requested test output on the symmetries, or if existing symmetries must be ignored.

NORMAL: First, this routine enforces the reciprocities of Eqs (14) and (27). Next, it evaluates the escape probabilities, uses the reciprocity relation with the partial blacknesses, Eq. (23). Finally, it applies the normalizations discussed in section 4.

KI3DAT: It is a BLOCK DATA subprogram that fills COMMON /KI3FN/ with coefficients and interpolation tables for the evaluation of the (partial) Bickley functions; see section 6. This large subprogram depends on the actual choice of angular discretizations, *k*; see chapter II.3 and Fig. IV.1. It has been generated by the auxiliary program GENKI3, which can be used to generate different discretizations.

REFERENCES

ASKEW J R, FAYERS F J and KEMSHELL P B (1966) A general description of the lattice code WIMS, *J. British nucl. Energy Soc.* **5**, 564-585.

BICKLEY W G and NAYLER J (1935) A short table of the function $Ki_n(x)$ from n=1 to n=16, *Philosophical Magazine* **20**, 343-347.

CARLVIK I (1965) A method for calculating collision probabilities in general cylindrical geometry and applications to flux distributions and Dancoff factors, *U.N. Intl Conf. on the Peaceful Uses of Atomic Energy, Geneva 1964*, **2**, 225-231.

CASAL J J, STAMM'LER R J J, VILLARINO E A and FERRI A A (1991) HELIOS: Geometric capabilities of a new fuel-assembly program, *Intl Topical Meeting on Advances in Mathematics, Computations, and Reactor Physics*, Pittsburgh, Pennsylvania, April 28-May 2, 1991, **2**, 10.2.1 1-13.

GELBARD E M (1985) Refinements in the computation of collision probabilities, *Intl Meeting on Advances in Nucl. Engng Computational Methods*, Knoxville, Tennessee, April 9-11, 708-715.

PRESS W H, FLANNERY B P, TEUKOLSKY S A and VETTERLING W T (1990) *Numerical Recipes (FORTRAN Version)*, Cambridge University Press, Port Chester, New York.

ROY R and MARLEAU G (1990) Normalization techniques for collision probability matrices, *PHYSOR–90: Intl Conf. on the Physics of Reactors: Operation, Design and Computation*, Marseille, France, April 23-27, **2**, IX 40-49.

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

VILLARINO E A (1989) An efficient method for calculating angular dependent collision probabilities, *INVAP S.E. Report 0713 0000 2IASS 100 1A*, S. C. de Bariloche, Argentina.

VILLARINO E A (1990) CONDOR: Código neutrónico para el cálculo de elementos combustibles con barras, *18th Annual Conference of the Argentinian Association of Nuclear Technology*, Buenos Aires, Argentina, October 22-26, 14.

VILLARINO E A, STAMM'LER R J J, FERRI A A and CASAL J J (1992) HELIOS: Angularly dependent collision probabilities, *Nucl. Sci. Eng.* **112**, 16-31.

WEISS Z (1990) Optimal algorithm for the calculation of contributions to collision probabilities from a neutron path, *Ann. nucl. Energy* **17**, 525-530.

VI. HELIOS: THE CRITICALITY SPECTRUM

VI. 1 INTRODUCTION AND DISCUSSION

In general, the eigenvalue K of the neutron-transport calculation is not equal to 1. Usually, however, the system calculated will be part of a critical reactor, i.e., its multiplication factor will be 1. So in a real situation, the difference, $K - 1$, is compensated by net out-leakage if $K > 1$, or by net in-leakage if $K < 1$.

In HELIOS, it is assumed that the two-dimensional (2D) spatial solutions in the energy groups are not influenced by that leakage. If that is not true, the User should include more of the surroundings in the system or specify better boundary conditions. Thus, leakage adjustments to the transport solution affect only the spectrum, and are caused by spatially independent (mainly axial) leakage. Obviously, these spectrum adjustments do not influence the cross-sections (XSs) per library group. However, they become important when few-group XSs are evaluated; in particular, when the XSs are integrated over the entire spectrum to obtain the reaction rates for burnup.

Other approximations to obtain a criticality spectrum are possible, e.g., adding an amount of a $1/v$ absorber or a constant absorber (the amount is negative if $K < 1$). Such approximations have no physical justification and can cause unphysical changes in the neutron spectrum, in particular if a $1/v$ absorber is added (Stamm'ler and Abbate, 1983).

Because the spatial solutions are frozen, the system can be homogenized[1] per group, preserving its reaction and boundary-leakage rates. In this way, the system is represented by a homogenized material with a set of group XSs and a global flux spectrum. The global spectrum together with the homogenized XSs still yields the multiplication factor K of the transport calculation.

It is for this system that the leakage is sought which makes it critical. The *criticality spectrum* that corresponds to this leakage is a better representation of the spectrum than the global spectrum of the transport calculation. Unless the User specifies otherwise, it is this criticality spectrum that is enforced on the spatial transport solution.

To determine the leakage and the criticality spectrum of the homogenized material, consider an arbitrarily shaped region of the homogenized material with arbitrary boundary conditions and extraneous sources. Assume that the angular flux inside this region can be represented by a modal expansion,

$$\Phi(\vec{r},\vec{\Omega},E) = \sum_n \phi_n(\vec{r})\Psi_n(\vec{\Omega},E) \ , \tag{1}$$

with $\vec{r}$ and $\vec{\Omega}$, the space and angular coordinates, and E the energy variable.

[1]Homogenization is discussed in chapter VIII.

The modes of expansion (1) have been separated in a spatial mode and an angle-energy mode, respectively. Seeking a solution leads to an eigenvalue problem for Ψ_n, with the (material) bucklings B_n^2 as eigenvalues. If they are arranged in descending order of their real parts, then the first is always real and the only one that can be positive—it is positive if the material has $k_\infty > 1$, and negative if $k_\infty < 1$. The others always have a negative real part, the magnitude of which is larger than that of the first buckling.

The spatial modes of the expansion (1), ϕ_n, are solutions of the Helmholtz or wave equation,

$$\nabla^2 \phi_n + B_n^2 \phi_n = 0 \ . \tag{2}$$

Owing to the behaviour of B_n^2 with n, the second and higher spatial modes decrease exponentially with increasing distance from influences that excite these modes, like boundaries and extraneous sources. Thus, ϕ_1 is normally in preponderance even at short distances from such discontinuities (even if ϕ_1 decreases, the other modes decrease faster). Therefore, the first mode is called the *fundamental mode*; the others are called transient modes.

It is the fundamental-mode spectrum that will be used as criticality spectrum, because it is the spectrum inside the homogenized material, away from boundaries and sources. Moreover, the fundamental mode is the only self-sustaining solution in a bare critical reactor that is made up of the homogenized material, with zero fluxes at the extrapolated boundary (Weinberg and Wigner, 1958; Stamm'ler and Abbate, 1983).

For large systems, an overall criticality spectrum may not be what the User wants. For example, a four-assembly calculation is done to obtain data for its individual assemblies. Then, criticality spectra of the individual assemblies may be of importance for data such as material buckling, migration area, consistent diffusion coefficients, and delayed-neutron constants. HELIOS, however, cannot calculate criticality spectra in parts of the system. Therefore, the output processor ZENITH has an option that evaluates the criticality spectrum in any part of the system for which HELIOS (the User) has provided certain output data.

Section 2 of this chapter describes how the fundamental-mode spectrum is found for a given material, including first-order (linear) anisotropic scattering. Associated with this spectrum are the material or critical buckling, the migration area, and a consistent set of group diffusion coefficients. Incidentally, ZENITH also uses the method described in this section.

Section 3 deals with the complications introduced by the boundary conditions of the transport calculation. Certain integral quantities, like k_∞, material buckling and diffusion coefficients, are inherent in the homogenized system, excluding possible net leakage and up-and-down scattering at the boundaries. The criticality spectrum, on the other hand, should include the effects of net boundary leakage.

Section 4, finally, outlines the subroutines used to obtain and enforce the criticality spectrum.

VI. 2 THE B_1 METHOD FOR THE CRITICALITY SPECTRUM[2]

The fundamental-mode spectrum does not depend on space, so that it is immaterial for which geometry it is solved. For simplicity, plane geometry is chosen. The fission source is isotropic and normalized to one neutron in the homogenized system, while up to first-order (linear) scattering anisotropy is treated. Under these assumptions, substitute the fundamental mode of Eq. (1), $\phi_1 \Psi_1$, into the Boltzmann transport equation and drop the index 1. This results in the following equations for the spatial and angle-energy modes:

$$\frac{d\phi(z)}{\mathrm{d}z} = \pm iB\phi(z) \tag{3}$$

and

$$\left(\Sigma(E) \pm iB\mu\right)\Psi(\mu,E) = \frac{1}{4\pi}\int\left(\Sigma_0(E'\to E)\psi(E') + 3\mu\Sigma_1(E'\to E)J(E')\right)\mathrm{d}E' + \frac{\chi(E)}{4\pi} \ . \tag{4}$$

The imaginary unit i in Eq. (3) is chosen to be in agreement with Eq. (2), so that B^2 has the significance of a conventional buckling. In Eq. (4), $\mu = \cos\theta$, where θ is the angle between the direction of neutron travel and the z axis; $\chi(E)$ is the fission spectrum; $\psi(E)$ and $J(E)$ are the flux and net- current spectra,

$$\psi(E) = 2\pi\int\Psi(\mu,E)\mathrm{d}\mu \quad \text{and} \quad J(E) = 2\pi\int\mu\Psi(\mu,E)\mathrm{d}\mu \ . \tag{5}$$

The remainder of this section describes how the fundamental-mode flux spectrum $\psi(E)$ is obtained by solving Eq. (4) with the P_1 and B_1 methods. This solution also yields the material buckling, the net-current spectrum, and diffusion coefficients that are consistent with linearly anisotropic scattering.

VI. 2.1 The P_1 and B_1 equations

To solve Eq. (4), the angular dependence of Ψ is expanded in Legendre polynomials, $P_l(\mu)$:

$$\Psi(\mu,E) = \frac{1}{4\pi}\sum_l (2l+1)\psi_l(E)P_l(\mu) \ , \tag{6}$$

with the expansion coefficients given by

$$\psi_l(E) = \int\Psi(\mu,E)P_l(\mu)\mathrm{d}\mu \ . \tag{7}$$

Because $P_0 = 1$ and $P_1 = \mu$, the first two moments are the flux and net-current spectra of Eq. (5).

[2]A detailed derivation with discussion has been presented by Stamm'ler and Abbate (1983).

VI. 2.1.1 The P_1 approximation

In the P_1 approximation, the expansion (6) is broken off after two terms. The resulting expression for $\Psi(\mu,E)$ is substituted in Eq. (4), which is then multiplied by P_0 and P_1, whereupon both equations are integrated over μ to yield the P_1 equations:

$$\left.\begin{aligned}\Sigma(E)\psi(E) \pm iBJ(E) &= \int \Sigma_0(E'\to E)\psi(E')\mathrm{d}E' + \chi(E)\\ \pm iB\psi(E) + 3\Sigma(E)J(E) &= 3\int \Sigma_1(E'\to E)J(E')\mathrm{d}E'\end{aligned}\right\} . \tag{8}$$

VI. 2.1.2 The B_1 approximation

Notice that, in the B_1 approximation, only the anisotropic scattering term on the right of Eq. (4) causes the μ-dependence of $\Psi(\mu,E)$ to differ from $(\Sigma \pm iB\mu)^{-1}$. Therefore, first divide the entire equation by $(\Sigma \pm iB\mu)$ and only then substitute the Legendre expansion (6) for $\Psi(\mu,E)$. Multiplication of the resulting equation by $P_l(\mu)$, followed by integration over μ, yields the B_l expansion

$$\psi_l(E) = A_{l0}\left(\int \Sigma_0(E'\to E)\psi(E')\mathrm{d}E' + \chi(E)\right) + 3A_{l1}\int \Sigma_1(E'\to E)J(E')\mathrm{d}E' , \tag{9}$$

where the coefficients A_{lj} are given by

$$A_{lj} = \tfrac{1}{2}\int_{-1}^{1}\left(\frac{P_l(\mu)Pj(\mu)}{\Sigma(E) \pm iB\mu}\right)d\mu . \tag{10}$$

Now, all the $\psi_l(E)$ can be expressed in terms of the first two, $\psi(E)$ and $J(E)$. This is a consequence of limiting scattering anisotropy to first order. In other words, restricting the B_l equations to the first two (the B_1 approximation) is by no means a truncation of the expansion (6) of $\Psi(\mu,E)$. In general, the B_L approximation is exact for scattering anisotropy up to order L.

After some manipulations, the first two B_l equations can be written as:

$$\left.\begin{aligned}\Sigma(E)\psi(E) \pm iBJ(E) &= \int \Sigma_0(E'\to E)\psi(E')\mathrm{d}E' + \chi(E)\\ \pm iB\psi(E) + 3\alpha\Sigma(E)J(E) &= 3\int \Sigma_1(E'\to E)J(E')\mathrm{d}E'\end{aligned}\right\} , \tag{11}$$

where

$$\alpha = \begin{cases}\tfrac{1}{3}x^2\left(\dfrac{\arctan(x)}{x - \arctan(x)}\right) & \text{for } x^2 = (B/\Sigma)^2 > 0,\\[2ex] \tfrac{1}{3}x^2\left(\dfrac{\ln((1+x)/(1-x))}{\ln((1+x)/(1-x)) - 2x}\right) & \text{for } x^2 = -(B/\Sigma)^2 > 0.\end{cases} \tag{12}$$

For small values of B/Σ, Eq. (12) can be approximated by

$$\alpha \cong \tfrac{1}{3}y\left(\frac{1-(y/3-y^2/5+y^3/7)}{y/3-y^2/5+y^3/7}\right) \quad \text{for} \quad |y| = |B/\Sigma|^2 \le 0.1 \ . \tag{13}$$

The latter expression has an accuracy of better than 0.035%.

From Eqs. (8) and (11) follows that the P_1 and the B_1 equations differ only by the coefficient α in the second equation. Since $\alpha > 1$, the B_1 method predicts the smallest leakage. Normally, $y < 0.05$, while even in the highest group y is less than 0.2, i.e., $\alpha < 1.013$ and $\alpha < 1.055$, respectively. Thus, in most practical cases the two methods produce marginally different results with the smallest leakage (smallest buckling, B^2) in the B_1 method.

VI. 2.2 Solution of the B_1 equations

To solve the B_1 equations (11), they are converted into multigroup equations:

$$\left.\begin{aligned} \Sigma_g\psi_g - \sum_h \Sigma_{0,g\leftarrow h}\psi_h \pm iBJ_g &= \chi_g \\ 3\alpha_g(B)\Sigma_g J_g - 3\sum_h \Sigma_{1,g\leftarrow h}J_h &= \mp iB\,\psi_g \end{aligned}\right\} . \tag{14}$$

In Eq. (14), g and h are the group indexes. The occurrence of the imaginary unit i is caused by the solutions $\exp(\pm iBz)$ of Eq. (3). For $B^2 > 0$ ($k_\infty > 1$), these solutions are combinations of the functions $\sin(Bz)$ and $\cos(Bz)$. Postulating that the flux spectrum $\psi(E)$ is real, Eqs. (11) show that $iJ(E)$ is real, too. The fact that $J(E)$ is imaginary expresses that there is a phase shift of $\pi/2$ between the flux and the current, or that the current is zero where the flux has an extremum. For $B^2 < 0$ ($k_\infty < 1$), iB is real and, again, both equations (11) are real.

Introducing vector-matrix notation, the current spectrum can be obtained from the second equation of (14). The solution is

$$\vec{J} = \mp iB\mathrm{D}\vec{\psi} \quad \text{with} \quad D^{-1}_{gh} = 3\alpha_g\Sigma_g\delta_{gh} - 3\Sigma_{1,g\leftarrow h} \ , \tag{15}$$

where δ_{gh} is the Kronecker delta, which is zero if $h \neq g$ and one if $h = g$. The matrix is called **D** because it represents the diffusion coefficient in the multigroup generalization of Fick's law. Indeed, recalling Eq. (3), Fick's law can be written as $J = -D(\mathrm{d}\phi/\mathrm{d}z) = \mp iBD\phi$.

If the number of library groups is G, then **D** is found by solving

$$D^{-1}\vec{x} = \left(\vec{e}_1 \cdots \vec{e}_g \cdots \vec{e}_G\right) \ \Rightarrow \ \mathrm{D} = \left(\vec{x}_1 \cdots \vec{x}_g \cdots \vec{x}_G\right) . \tag{16}$$

where e_g is the g^{th} unit vector with components δ_{gh}. So $\mathrm{D}^{-1}\vec{x} = \vec{e}_g$ is solved G times, for G right-hand sides. The array of the G solution (column) vectors is the sought matrix, **D**. Gauss elimination is used to solve Eq. (16).

With **D** known, the next step of the solution process is to substitute Eq. (15) in the first equation of (14). This yields the following vector-matrix equation for the flux spectrum:

$$\mathrm{A}\vec{\psi} = \vec{\chi} \quad \text{with} \quad A_{gh} = \Sigma_g \delta_{gh} - \Sigma_{0,g\leftarrow h} + B^2 D_{gh} \ , \tag{17}$$

which also is solved by Gauss elimination, this time with only one right-hand side, the fission spectrum.

Observe that both i and the ± sign have disappeared from Eq. (17). Further, summing the first equation of (14) over all the groups g yields the balance equation: *absorptions* + *leakages* = (*fission*) *source* = 1. Because the fission source is normalized to one, the multiplication factor of the system, leakage included, is given by

$$k = \sum_g (\nu\Sigma_f)_g \psi_g \ . \tag{18}$$

VI. 2.2.1 The buckling iterations

The criticality spectrum is the spectrum that is associated with the material or critical buckling for which the multiplication factor is 1. The solution process consists of iterating on B^2 until its corresponding multiplication factor $k = 1$. Eqs. (14) are first solved for $B^2 = 0$. On account of Eq. (15), the current spectrum is zero—no leakage—so $k(B^2=0) = k_\infty$.

Next, a not-too-small value is taken for B^2 (presently, $2 \cdot 10^{-3}$ cm^{-2}), which is set negative if $k_\infty < 1$. With this new value of B^2, the flux and current equations are solved again, yielding the multiplication factor k_1. To obtain a good estimate for the sought $B^2(k=1)$ the two values $B^2(k_\infty)$ and $B^2(k_1)$ are used in an interpolation or extrapolation based on the well-known relation for the material buckling, $k = k_\infty/(1+M^2B^2)$. This gives

$$B^2 = \frac{1}{M^2}\left(\frac{k_\infty}{k} - 1\right) \quad \text{and} \quad \mathrm{d}B^2 = \frac{k_\infty}{M^2}\mathrm{d}\left(\frac{1}{k}\right) \ . \tag{19}$$

The coefficient k_∞/M^2 is found after the first two steps from

$$\frac{k_\infty}{M^2} = \frac{B^2(k_1)}{\dfrac{1}{k_1} - \dfrac{1}{k_\infty}} \ . \tag{20}$$

If this relation were rigorous, the sought $B^2(k=1)$ would be

$$B^2(k=1) = B^2(k_1) + \frac{k_\infty}{M^2}\left(1 - \frac{1}{k_1}\right) . \tag{21}$$

With this B^2, a new flux spectrum and its associated multiplication factor k_2 are obtained. If $|k_2 - 1| > \varepsilon$, Eq. (21) is used again but this time with k_2, to estimate $B^2(k=1)$, etc. Normally, very few iterations (<8) are enough to reach an accuracy of $\varepsilon = 5\text{x}10^{-6}$. Note that the coefficient k_∞/M^2 is evaluated only once. Reevaluating it after each iteration step could lead to divergence when the iterations have almost converged due to a loss of significant digits when $k_{i-1} \cong k_i$ and $B^2(k_{i-1}) \cong B^2(k_i)$.

It has been observed in certain cases that the initial guess of $B^2(k_1) = 2\text{x}10^{-3}$ is too big and the iterations do not converge. Therefore, if the buckling iterations do not converge within 50 iterations or negative fluxes appear in the solution of the B_1 equations, then the iteration process is started anew from a 20 times smaller initial guess of $B^2(k_1)$.

It also may happen, in particular if $k_\infty < 0.3$, that the B_1 equations produce negative fluxes or that the buckling iterations do not converge, even after restarting from another initial guess. Because it is important to do the burnup calculations in a criticality spectrum, the process is tried again—but now in the P_1 approximation—by setting $\alpha = 1$—see Eqs. (8) and (11). (It appears that the P_1 method is more stable in such situations.) From then on, only the P_1 method is used. If that happens, HELIOS notifies the User by a message.

VI. 2.2.2 Migration area and B_1 diffusion coefficients

Knowing the critical buckling and the flux and current spectra of the homogenized material, the migration area and B_1 diffusion coefficients can be found. The migration area is evaluated from the well-known formula

$$M^2 = \frac{k_\infty - 1}{B^2} . \tag{22}$$

Recalling Eq. (15), the flux and current spectra together with the critical buckling yield the group-diffusion coefficients of Fick's law as follows:

$$D_g = \frac{J_g}{|B|\psi_g} . \tag{23}$$

The above expression for the diffusion coefficient is consistent with multigroup theory and includes linear scattering anisotropy. It is for the calculation of the criticality spectrum that P_1 scattering is included in the nuclear-data library. The transport correction used in the transport calculations is already included in the diagonal of the P_0-scattering matrix and the "total" XSs; see also chapter II.5.2.

Of course, the diffusion coefficients obtained from Eq. (23) are for the homogenized material. Yet, when divided by the diffusion coefficients obtained from the (homogenized) "total" XSs, $D = 1/3\Sigma_t$, they provide global correction factors that can be used for parts of the system; see also chapter VIII.3.

VI. 3 BOUNDARY CONDITIONS

In the transport calculation, interactions with the surroundings are described by boundary conditions; see chapter II.4 and IV.3.2. In the cases of boundary coupling and specular reflection, the system is infinite (there are no surroundings). In the case of total reflection, with albedo matrix $a_{hg} = \delta_{hg}$, there is also no interaction with the surroundings—no net leakage nor albedo scattering.

Any other albedo matrix, however, implies boundary interaction - for instance, if the diagonal elements are bigger than 1, there is in-leakage; if they are less than 1, there is out-leakage. If there are non-zero off-diagonal elements, there is albedo scattering between the groups, e.g., thermal neutrons enter the surroundings where they cause fission and some of the fission neutrons enter the system. The boundary's properties are represented by additional "homogenized" absorption and scattering XSs,

$$\delta\Sigma_{ag} = \frac{\sum_b \left(1 - \sum_h a_{b,h\leftarrow g}\right) j_{bg}^-}{\sum_i \phi_{ig}} \quad \text{and} \quad \delta\Sigma_{s,h\leftarrow g} = \frac{\sum_b a_{b,h\leftarrow g} j_{bg}^-}{\sum_i \phi_{ig}} \ . \tag{24}$$

In Eq. (24), the b and i summations are over the boundary sectors and regions of the system, the fluxes are volume-integrated and the in-currents into the boundary sectors (out of the system) are surface-and-sector integrated.

Thus, the *system* (boundary conditions included) is reduced to two components: the (homogenized) *material* within the boundaries and the *boundary*. Now, there are at least four different spectra involved: two for the material, at zero buckling and at its material buckling; and two for the system, at zero and at critical buckling—notice the distinction between material and critical buckling. These spectra and their associated quantities are discussed below.

VI. 3.1 The material (boundaries excluded)

Although the material is considered separate from the boundary, its XSs include the boundary's effect on the spatial flux shape, since they have been obtained by spatial homogenization. Otherwise, because these XSs are in the same groups as those of the nuclear-data library, they are independent of the spectrum and thereby independent of the boundary and the buckling.

VI. 3.1.1 The infinite multiplication factor, k_∞

The definition of k_∞ is ambiguous. Obviously, any infinite multiplication factor worthy of its name cannot include boundary or buckling leakage in its definition. Thus, k_∞ is defined as the ratio of the production rate from fission to the loss rate from absorption, and absorption only,

$$k_\infty = \frac{P(0)}{A(0)} = \frac{\sum_g \sum_i (\nu\Sigma_f)_{ig}\phi_{ig}(0)}{\sum_g \sum_i \Sigma_{a,ig}\phi_{ig}(0)} \; . \tag{25}$$

The argument "0" in $P(0)$ and $A(0)$ indicates that these are the rates in a zero-buckling spectrum.

Strictly speaking, the k_∞ of Eq. (25) is not the true k_∞. The reaction rates still include the spatial influence of the boundary in the homogenized XSs. It would be meaningless to try to eliminate this influence on the homogenized XSs, only for the purpose to find the "true" k_∞. First, the expensive transport calculation would have to be done twice, with and without boundary interaction. Second, it can be argued that this spatial influence is an essential property of the material, which is just why the boundary conditions were specified. Third, k_∞ does not have to be "true"; it is merely a measure of the reactivity of the system, so details in its definition are not very important. Fourth, as will be seen below, the k_∞ inherent in the few-group XSs is, in any case, different from the k_∞ of Eq. (25).

In view of the above remarks, k_∞ in HELIOS is defined as the value inherent in the homogenized XSs in the library groups. It excludes spectrum effects caused by boundary and buckling leakage, while it includes spatial boundary effects on the homogenized XSs. It is evaluated in the subroutine BEEONE where it is called **KINF**. The associated infinite-medium spectrum of the material is discarded as uninteresting.

VI. 3.1.2 Material buckling, migration area, diffusion coefficients

Like k_∞, material buckling, migration area, and diffusion coefficients are properties of the material, rather than of the system. They are also evaluated without the boundary, though with buckling leakage. The material buckling is defined as that buckling, B_m^2, whose associated buckling-leakage rate, $B\left(B_m^2\right)$, renders the material critical, i.e.,

$$k(B_m^2) = \frac{P(B_m^2)}{A(B_m^2) + B(B_m^2)} = 1 \; . \tag{26}$$

This is the buckling that is evaluated according to the methods of section 2.2.

The migration area, M^2, is obtained from k_∞ and B_m^2, according to Eq. (22),

$$M^2 = \frac{k_\infty - 1}{B_m^2} \ . \tag{27}$$

Finally, the diffusion coefficients are found from the current-to-flux ratio, Eq. (23); this is done in the library groups. Since they are almost independent of the spectrum, any spectrum with non-zero currents can be used ($B^2 \neq 0$). In HELIOS, the spectrum associated with the material buckling is taken.

All of the above material data are evaluated in the subroutine BEEONE, where they are called **BSQ**, **MSQ** and **DB1()**. Empty parentheses denote an array—DB1() has the dimension of the number of library groups. Both the criticality spectrum of the material and its k_∞ (evaluated from production and absorption rates in that spectrum) are discarded as uninteresting.

VI. 3.2 The system (boundaries included)

If there is no boundary interaction, the material's criticality spectrum and its k_∞ are identical to those of the system—they are not discarded. However, if there is boundary interaction, it is the criticality spectrum of the system that is used to evaluate the group-condensed output XSs and reaction rates for the burnup calculations. The same calculations that were done for the material are then repeated for the system by adding the additional XSs of Eq. (24) to those of the material.

VI. 3.2.1 Zero buckling: the eigenvalue of the transport calculation

The infinite multiplication factor of the system, $k_{s\infty}$, is defined analogously to that of the material. Recalling Eq. (25), this yields

$$k_{s\ \infty} = \frac{P_s(0)}{A_s(0)} \ . \tag{28}$$

The subscript s indicates that the rates are for the system (boundary included); the argument "0" means at zero buckling. Also this $k_{s\infty}$ and its associated spectrum are discarded as uninteresting. Incidentally, this is the spectrum of the transport calculation.

Including in Eq. (28) the boundary-leakage rate $L(0)$, still at zero buckling, gives the already-known multiplication factor of the transport calculation,

$$k_{tr}(0) = \frac{P_s(0)}{A_s(0) + L_s(0)} = K \ . \tag{29}$$

This multiplication factor is called **EIGV**; it is defined as the ratio of the production rate to the combined loss rate from absorption and boundary leakage. Although this eigenvalue is of no interest to the User, it is saved because it is also obtained from the transport calculation. Observe that, in the absence of boundary interaction, $k_\infty = k_{s\,\infty} = k_{tr}(0)$.

VI. 3.2.2 The criticality spectrum

As mentioned at the start of this chapter, the system is assumed to be part of a critical reactor with a multiplication factor of 1. This is simulated by rebalancing the global spectrum of the transport calculation (the group fluxes, ϕ_g) with the criticality spectrum. This spectrum and its associated critical buckling are obtained in the subroutine BEEONE, according to the methods of section 2.2. Thus the critical buckling is defined from

$$k(B_{cr}^2) = \frac{P_s(B_{cr}^2)}{A_s(B_{cr}^2) + L_s(B_{cr}^2) + B(B_{cr}^2)} = 1 \ , \tag{30}$$

where $B\left(B_{cr}^2\right)$ is the buckling-leakage rate, and the argument B_{cr}^2 indicates at critical buckling. Observe that, in the absence of boundary interaction, material and critical buckling are identical.

The group-flux rebalancing enforces the criticality spectrum on the transport fluxes ϕ_{ig}, by multiplying them by the ratio of the global transport spectrum at zero buckling to the (global) criticality spectrum at critical buckling,

$$\phi_{ig}(B_{cr}^2) = \frac{\phi_{sg}(B_{cr}^2)}{\phi_{sg}(0)} \phi_{ig}(0) \ . \tag{31}$$

The global spectra in Eq. (31) are indexed by s, which means that the fluxes refer to the system with the boundary. The fluxes $\phi_{ig}(0)$ come straight from the transport calculation. Once they have been scaled to power, the $\phi_{ig}\left(B_{cr}^2\right)$are the final fluxes of HELIOS, which are used for burnup and output processing.

Similarly to $k_{s\infty}$, the $k_{cr\infty}$ associated with the criticality spectrum is

$$k_{cr \quad \infty} = \frac{P_s(B_{cr}^2)}{A_s(B_{cr}^2)} \ . \tag{32}$$

In HELIOS, the rebalancing factors, the global-spectra ratios in Eq. (31), are called **B1FACT()**, and $k_{cr\infty}$ is called **KINFB**; the critical buckling is discarded as unimportant. The 'B' appended to KINF indicates that it has been evaluated in the criticality spectrum of the system, i.e., with both boundary and buckling leakage included. $k_{cr\infty}$ is inherent in the one-group XSs (and in the burnup rates), while k_∞ is inherent in the homogenized XSs in the library-group structure. The $k_{x\infty}$ inherent in the XSs of an intermediate group structure lies somewhere in between these two k_∞ values.

VI. 3.3 Other options and summary table

There are two options to avoid the use of the criticality spectrum. Both can be specified in the RUN operator; see "User Manual AURORA." The methods option (MT) can be set to use the global transport spectrum. The buckling option (BSQ) can specify an input buckling whose spectrum is to be used instead of the criticality spectrum, unless it is nullified by the MT option.

In both options, the material quantities still are calculated, i.e., KINF, BSQ, MSQ and DB1(). If the global transport spectrum is to be used, B1FACT() is set to one. If an input buckling is given, its associated spectrum is used to define B1FACT(). The latter option is for the interpretation of critical experiments for which a buckling is given. The multiplication factor of the experiment—which is usually 1, unless it is a Monte Carlo calculation—is given by

$$k(B^2) = \frac{P_s(B^2)}{A_s(B^2) + L_s(B^2) + B(B^2)} \ . \tag{33}$$

Boundary	B^2	Quantities	Remarks
Excluded	0	KINF	Nearest to the "true" k_∞
Excluded	B_m^2	BSQ, MSQ, DB1()	Material properties
Included	0	EIGV	Transport calculation
Included	B_{cr}^2	KINFB, B1FACT()	Criticality spectrum for burnup and output XSs
Included	Input	B2, EIGVB2, KINFB, B1FACT()	For critical experiments

In HELIOS, the input buckling and its multiplication factor are called **B2** and **EIGVB2**. The table above is a compilation of the various quantities that (can) occur in the criticality calculations.

VI. 4 SOME REMARKS ON THE PROGRAM

The calculations related to the criticality spectrum are done in LINK17. This module, which is only called for the neutronics calculations, also sets up the burnup reaction rates; see chapter VII.

VI. 4.1 Notes on LINK17

Apart from the main administrative subroutine MAIN17, the following three subroutines are involved in the calculations described in this chapter:

BONEXS:	It sets up system-homogenized XSs for the B_1 calculations. It also converts boundary leakage and albedo up-and-down scattering into the additional XSs of Eq. (24).
BEEONE:	It evaluates all the quantities in the table, except EIGV and B2. This is done in two loops. In the first loop, the boundary is excluded. In the second loop, which is executed if there is boundary interaction, the boundary is included.
BURRAT:	It rebalances the transport flux with the criticality spectrum, according to Eq. (31). In this subroutine, the flux is also scaled to power, and the burnup reaction rates are set; see chapter VII.

REFERENCES

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

WEINBERG A M and WIGNER E P (1958) *The Physical Theory of Neutron Chain Reactors*, The University of Chicago Press.

VII. HELIOS: BURNUP

VII. 1 INTRODUCTION AND TERMINOLOGY

During operation of a reactor its material compositions change because of exposure to the neutron flux and because of radioactive decay. These changes affect important quantities like multiplication factor and power distribution. This introduces a time dependence of the system, which is described by a series of flux calculations (discussed in chapters III–VI) at fixed points in time. The composition changes during a time step are evaluated, using the constant reaction rates per isotope at either end of the step. These so-called burnup or depletion calculations are the topic of this chapter.

The following terminology is used:

- <u>*Neutron reactions*</u> with isotopes that can be treated in the burnup calculations are: (n,γ), (n,f), $(n,2n)$ and $(n,3n)$. All of these processes consume isotopes while, at the same time, producing others.
- <u>*The fission process*</u> is the neutron reaction (n,f). Besides ν fast neutrons and the fission energy κ, it produces two fission fragments. The initial fission fragments have a large proton deficiency and will suffer three to four β-decays before they become stable, e.g.,

$$\text{Xe-140} \xrightarrow{14\text{ s}} \text{Cs-140} \xrightarrow{67\text{ s}} \text{Ba-140} \xrightarrow{13\text{ d}} \text{La-140} \xrightarrow{40\text{ h}} \text{Ce-140 (stable)}. \quad (1)$$

- <u>*Burnup isotopes*</u> belong to the fuel proper; they are either heavy-metal isotopes or fission-product isotopes (see below). Whereas fresh fuel usually has only two burnup isotopes, U-235 and U-238, many more are produced during burnup due to neutron reactions and radioactive decay.
- <u>*Heavy-metal isotopes*</u> are burnable isotopes with atomic weights greater than 225 amu. In general, they can undergo fission (there must be at least one fissionable heavy-metal isotope). Figure 1 shows an example of a setup of heavy-metal burnup chains. Only the isotopes in the solid boxes are treated; those in the shaded areas are assumed to decay instantaneously.
- <u>*Fission-product isotopes*</u> are burnable isotopes with atomic weights not greater than 225 amu. They cannot be fissioned but are produced by the fission process (see there). They are the fission fragments and their daughters due to subsequent radioactive decays and/or neutron reactions. Figures 2-6 show possible fission-product chains. A horizontal arrow indicates neutron absorption, with estimates of σ(2200 m/s) and RI_∞ above and below the line; a vertical arrow, with its accompanying number, indicates β decay and half-life; a slanted arrow, with its accompanying number, indicates fission yield (see below) and its approximate value (in %) for Pu-239 fission in a thermal reactor.

Bifurcation fractions of reactions:

Am-241	=>	Am-242, 0.86
Am-241	=>	Am-342, 0.14
Am-242	=>	Cm-242, 0.827
Am-242	=>	Pu-242, 0.173
Np-237	=>	Np-236, 0.41 [0.36 from $(n,2n)$ + + 0.05 from (γ,n) reaction]

↑ : neutron capture
↓ : $(n,2n)$ reaction, unless otherwise stated
← : electron capture, unless otherwise stated
→ : beta decay or the decay of Am-242 to its ground state, unless otherwise stated

Am-342 is the metastable state of Am-242, into which it decays with a half life of 150 years

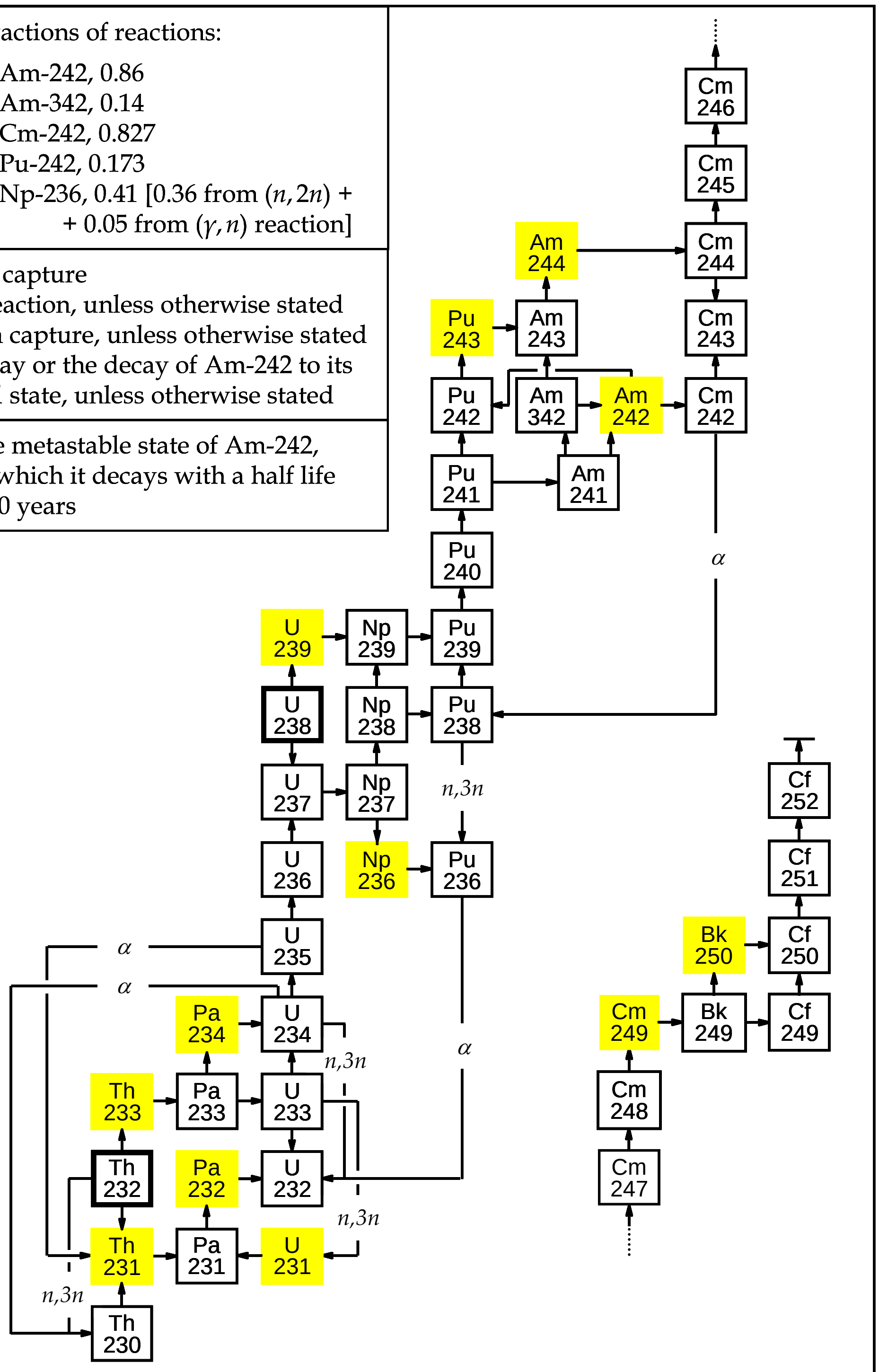


Fig. 1: Heavy-metal chains.

- *The yield* of a fission product is its probability of creation by the fission process and possible subsequent β-decays (no neutron reactions). The yields that are in the nuclear-data library are either direct yields from fission, without preceding decays, or cumulative yields, including decays from precursors in the fission-fragment chain; see Eq. (1).

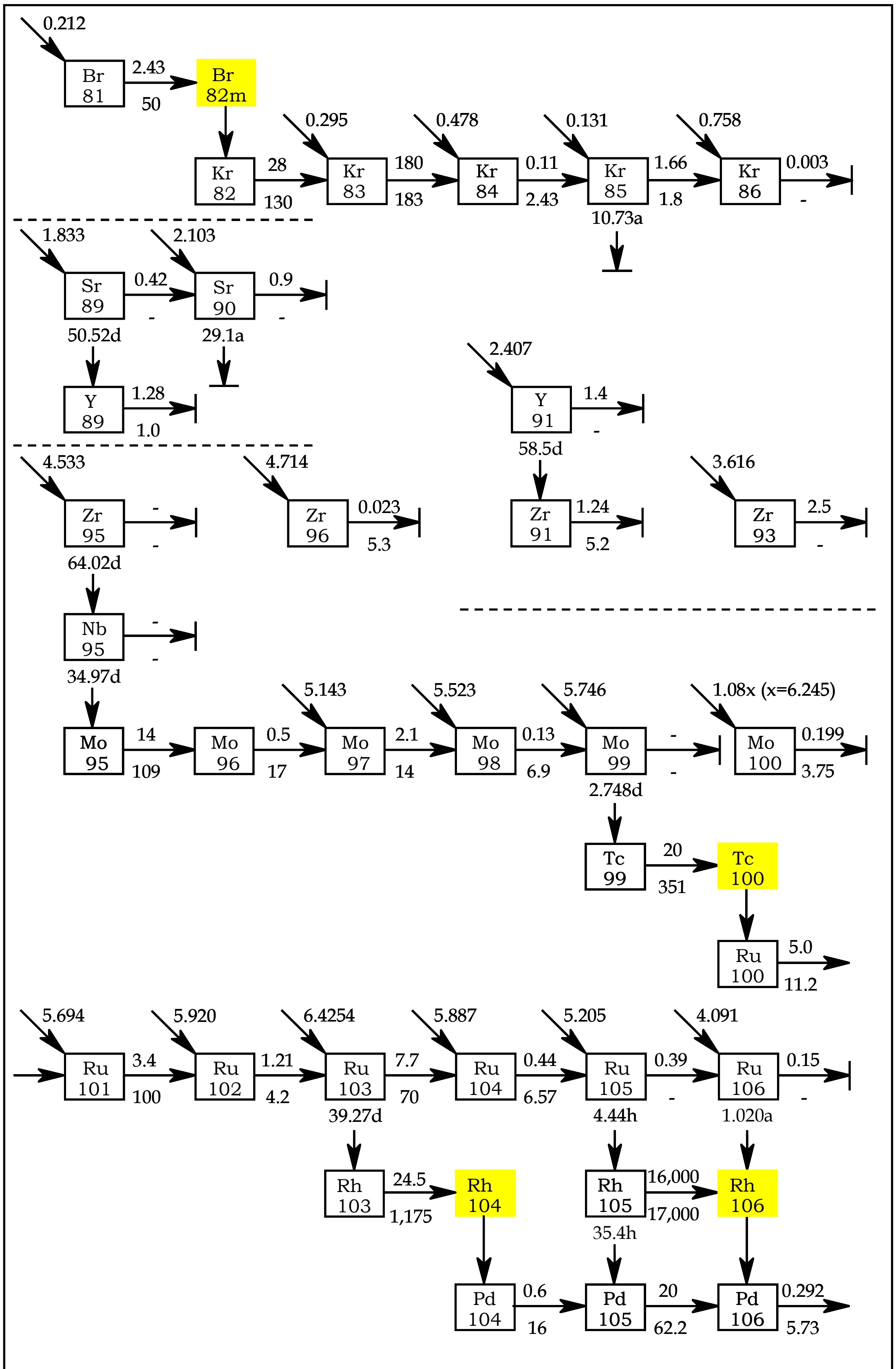


Fig. 2: Fission-product chains.

(continues ...)

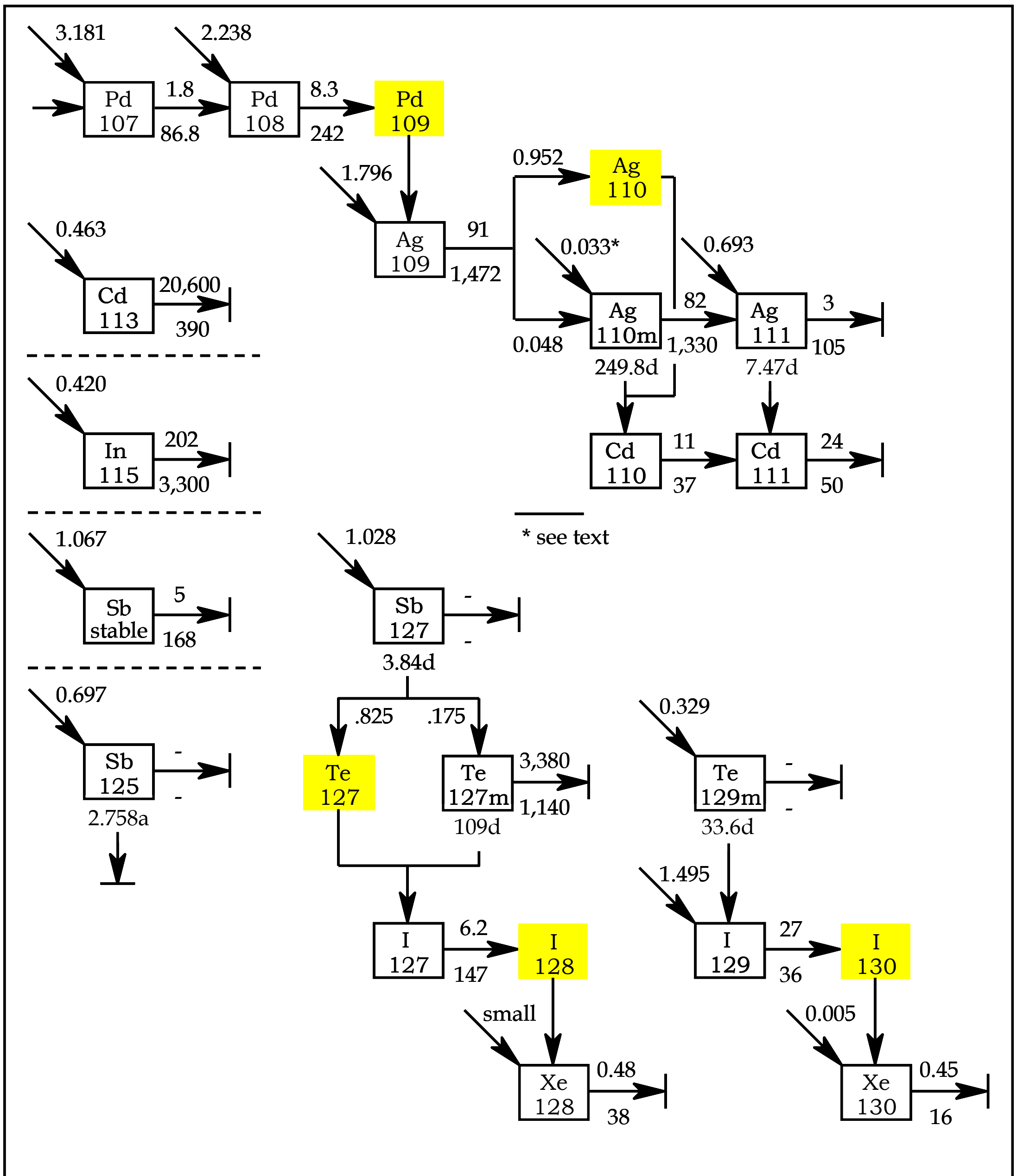


Fig. 3: Fission-product chains. (continues …)

- *Burnable-absorbers* gradually lose their initial absorptive power (and concentration) during burnup due to neutron capture. They are either burnable isotopes or burnable-absorber isotopes (see below).
- *Burnable isotopes* are burnable-absorbers that are treated explicitly, i.e., the individual isotope(s) of their burnup chain(s). Figure 7 shows examples of burnable-isotope chains.
- *Burnable-absorber isotopes (BAs)* are burnable-absorbers that are not treated as individual isotopes of one or more burnup chains. Instead, one or more chains are represented by a single component, the BA, with pre-calculated tables of effective absorption cross sections. They are discussed in chapter II.5.1.2, where they are called burnable absorbers.

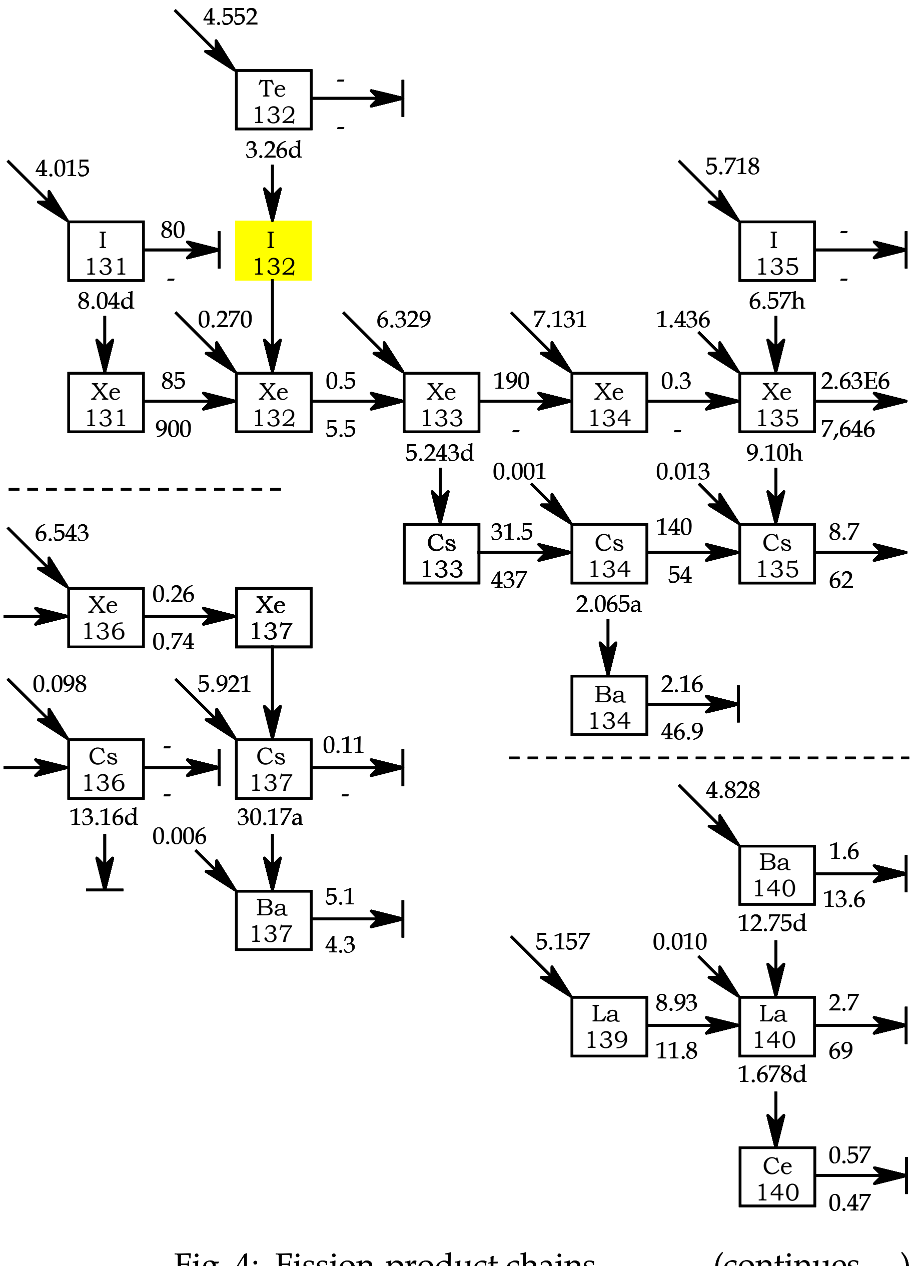


Fig. 4: Fission-product chains. (continues …)

Figs. 1–7 show that there are many isotopes in the burnup chains and that the chains are intertwined. Often, the number of fission products is reduced by representing most of them by a few *lumped or pseudo fission products*. The heavy-metal and fission-product chains in HELIOS are defined by the BLOCK DATA subprogram BUDATA—of course, the isotopes must exist in the nuclear-data library. The auxiliary program VULCAN produces this subprogram from an easy input, so a change in the isotopic chains can be readily incorporated.

From the viewpoint of this document, it is irrelevant which heavy-metal and fission-product chains are used. The chains in Fig. 1 do not correspond exactly with those in the actual HELIOS version. For instance, α-decay of U-234 and U-235 is not considered, nor are the $(n,3n)$ reactions, although they once were included.

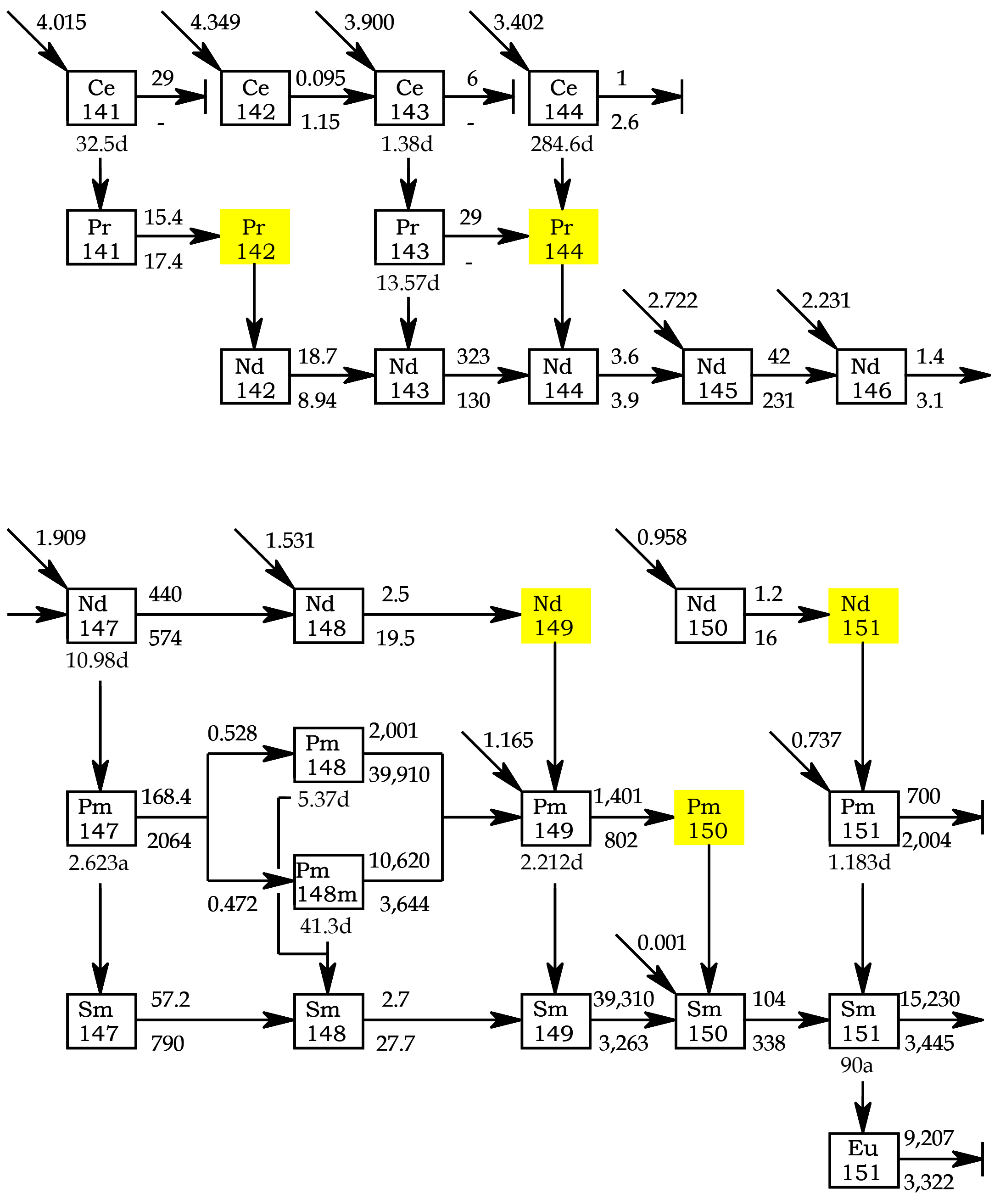


Fig. 5: Fission-product chains. (continues …)

The two main criteria for including isotopes in the burnup are their importance for the reactivity up to burnups of at least 70,000 MWd/t and their importance for the radioactivity of fuel that has cooled for at least a week. So isotopes with half-lives shorter than a couple of days are not considered, unless they delay the formation of important isotopes (Pm-149), have a large absorption cross-section competing with the decay (Xe-135), or are used in other applications (La-140 for γ-scanning of axial power distributions). Also, isotopes that build up to relatively high concentrations—such isotopes have a high cumulative yield and/or production from neutron capture by their predecessors in the burnup chains—must be considered.

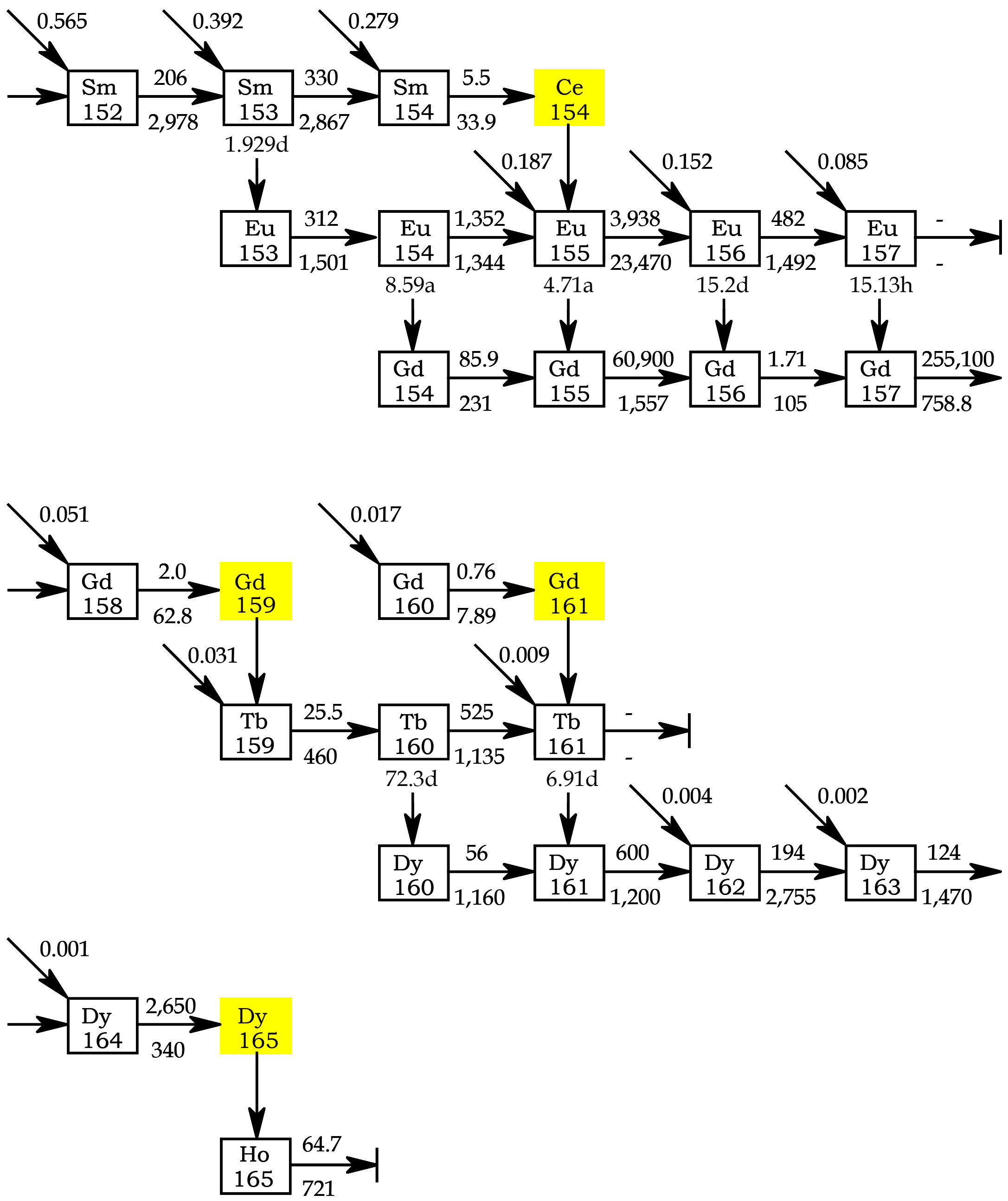


Fig. 6: Fission-product chains. (end)

The chains in Figs. 2–6 give a good description of the fission-product poisoning with burnup. According to Fredin (1991, 1992), the effect of those not present in the chains may be accounted for by minor changes in the data of some of the existing fission products. The fission yield of Mo-100 is increased by 8% to represent the effect of the remaining stable fission products. Ag-110m has a yield of 33 pcm from fission of any isotope, and the absorption cross-section of Ce-141 is increased by a factor of 1.2, to account for the effect of the remaining decaying fission products.

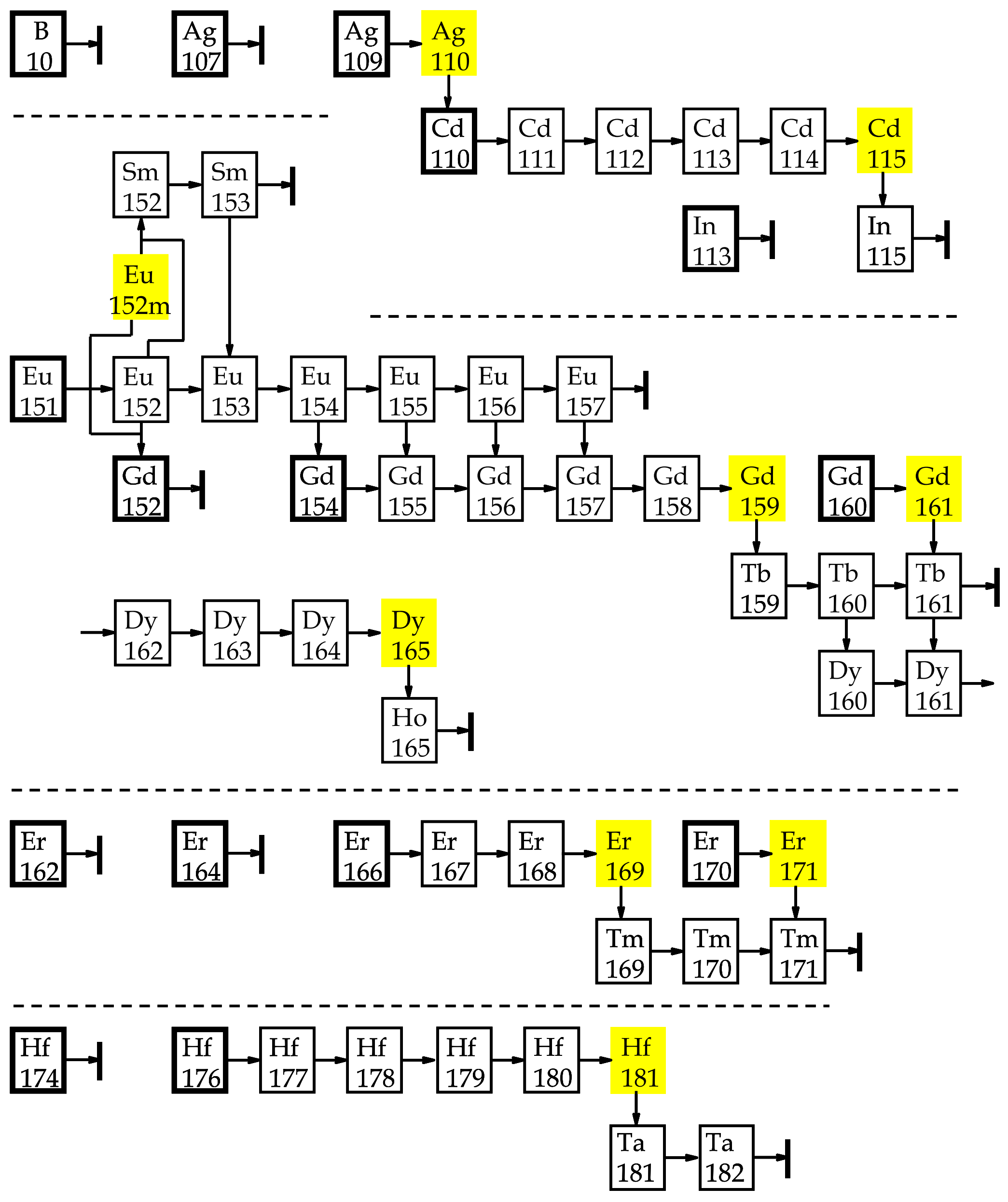


Fig. 7: Burnable-isotope chains.

The remainder of this chapter describes the solution of burnup chains by HELIOS, regardless of which isotopes are included in the chains.

VII. 2 LINEARIZATION OF THE BURNUP CHAINS

The first step in the solution of the burnup chains is to linearize the chains. In a linearized chain, each isotope, except the first and the last, has only one precursor and one successor. At the beginning of a burnup step, each isotope has an initial concentration $N(0)$, which may be zero, and during the step it may have a constant yield rate from fission, y.

$\boxed{a}$

$$\begin{tikzcd}
 & & C \arrow[dr] & & \\
A \arrow[r] & B \arrow[ur] \arrow[dr] & & E \arrow[r] & {} \\
 & & D \arrow[ur] & &
\end{tikzcd}$$

1) $A(1) \rightarrow B(1) \rightarrow C(1) \rightarrow E(1) \rightarrow$

2) $A(3) \rightarrow B(2) \rightarrow D(1) \rightarrow E(2) \rightarrow$

- -

$\boxed{b}$

$$\begin{tikzcd}
 & B \arrow[dr] & & D \arrow[dr] & & F \\
A \arrow[ur] \arrow[dr] & & C \arrow[ur] \arrow[dr] & & E \arrow[ur] \arrow[dr] & \\
 & X \arrow[ur] & & Y \arrow[ur] & & Z
\end{tikzcd}$$

1) $A(1) \rightarrow B(1) \rightarrow C(1) \rightarrow D(1) \rightarrow E(1) \rightarrow F(1)$

2) $A(3) \rightarrow B(3) \rightarrow C(3) \rightarrow D(3) \rightarrow E(3) \rightarrow Z(1)$

3) $A(3) \rightarrow B(3) \rightarrow C(3) \rightarrow Y(1) \rightarrow E(2) \rightarrow F(2)$

4) $A(3) \rightarrow B(3) \rightarrow C(3) \rightarrow Y(3) \rightarrow E(4) \rightarrow Z(2)$

5) $A(3) \rightarrow X(1) \rightarrow C(2) \rightarrow D(2) \rightarrow E(2) \rightarrow F(2)$

6) $A(3) \rightarrow X(3) \rightarrow C(4) \rightarrow D(4) \rightarrow E(4) \rightarrow Z(2)$

7) $A(3) \rightarrow X(3) \rightarrow C(4) \rightarrow Y(2) \rightarrow E(2) \rightarrow F(2)$

8) $A(3) \rightarrow X(3) \rightarrow C(4) \rightarrow Y(4) \rightarrow E(4) \rightarrow Z(2)$

Fig. 8: Chain linearizations.

Figure 8 illustrates two examples of chain linearization. In Fig. 8a, the path $A \rightarrow B$ must be calculated twice because the rest of the two linear chains differ. C and D occur only once, while the contributions to E, and to its successors via C and D, must be summed. So, linearization causes parts of the chains to be repeated. Depending on their occurrence, the isotopes are treated differently during a burnup step. This is indicated in Fig. 8 by the numbers in parentheses. The four types of treatment of an isotope are:

1) First occurrence of an isotope: $N(0)$ and y are used when solving the chain, and the number density obtained at the end of the burnup step, $\Delta N(t)$, contributes to the new number density $N(t)$ of the isotope.

2) Next occurrence of an isotope and of the rest of the chain: $N(0)$ and y are set to zero, and $\Delta N(t)$ is added to $N(t)$.

3) Next occurrence of an isotope and of the preceding chain: $N(0)$ and y are used, but $\Delta N(t)$ is not added to $N(t)$.

4) Next occurrence of an isotope, of the preceding chain and of the rest of the chain: $N(0)$ and y are set to zero, and $\Delta N(t)$ is not added to $N(t)$.

Except for radioactive decay, the loss and gain rates of an isotope, as well as its source rate from fission yield, depend on the neutron flux. The flux found from the transport calculation (chapter IV), with the criticality spectrum enforced (chapter VI), must still be scaled to the specified power level. Note that of the three flux types, it is only the neutron fluxes that must be scaled. The flux level of the resonance calculations is irrelevant, and the level of the gamma fluxes depends on the gamma sources, which are determined by the fission and absorption rates of the neutron flux (chapter IV.2.1).

Let the power level be specified in the input as P W/g, which means watts per gram of heavy metal in the fresh fuel (often only U-235 and U-238). Then the power scaling factor P_S, at any time t, is found from:

$$P_s = \frac{\sum\limits_{\substack{iso \\ A_{iso}>225}} \sum\limits_{i} N_{iso,i}(0) A_{iso} V_i}{\sum\limits_{iso}\sum\limits_{i}\sum\limits_{g} N_{iso,i}(t)(\kappa\sigma_f)_{iso,g}\phi_{ig}V_i} P \ . \tag{2}$$

In Eq. (2), the indexes iso, i, and g denote isotopes, regions, and energy groups; N_{iso} are isotopic number densities, where $t = 0$ represents fresh fuel; A_{iso} are atomic masses in grams [in the library they are stored in atomic mass units; see Eq. (II.24)]; V_i are region volumes; ϕ_{ig} are the group fluxes, which in this chapter are not region-integrated; κ is the energy liberated per fission; and σ_f is the microscopic fission cross-section.

The fluxes in Eq. (2) are the fluxes of the transport calculation multiplied by the rebalancing factors that enforce the criticality spectrum; see Eq. (VI.31). Multiplication of these fluxes by P_s yields the final fluxes. These are the fluxes that are used for burnup, output processing, and the evaluation of the gamma source. The power-scaling factor P_s, which is called PSCALE, and the rebalancing factors, which are called B1FACT(), are saved. Together with the final fluxes, they are used to restore the flux of the transport calculation as the initial guess for the next calculational point (if there is one)—see chapter IV.4.1.

The burnup equations are solved for each material that contains burnup isotopes, burnable isotopes, or BAs, unless the User has specified an otherwise burnable material as non-burnable. Therefore, from here until section 3, the region index is dropped and the indexes i, j, k, and n are used for the isotopes in a chain. The general burnup equation for the number density of isotope i in a linearized chain is

$$\frac{\mathrm{d}N_i}{\mathrm{d}t} = y_i + g_{i-1}N_{i-1}(t) - \ell_i N_i(t) \ . \tag{3}$$

In Eq. (3), t is the time, which for ease of notation is taken as zero at the beginning of each burnup step. The source rate from fission yield is assumed constant during the time step T, and equal to its average value. This gives

$$y_i = \frac{1}{T}\int \sum_j y_{i\leftarrow j}\sigma_{f,j}N_j(t)\mathrm{d}t\phi \equiv \sum_j y_{i\leftarrow j}\sigma_{f,j}\phi\overline{N}_j \ , \tag{4}$$

where the sum is over the fissionable isotopes, *j*; $y_{i\leftarrow j}$ is the yield of *i* from fission of *j*; and ϕ is the total (spectrum-integrated) final flux in the region, now defined without the volume *V* from $\phi=\Sigma\phi_g$. The gain rate depends on the reaction *x* by which *i* is produced. Thus, $g_{i\text{-}1} = \lambda_{i\text{-}1}$ for decay, and $g_{i\text{-}1} = f\sigma_{x,i\text{-}1}\phi$ for capture, (*n*,2*n*), or (*n*,3*n*), with *f* the fraction of reaction *x* that produces *i*.[1] Finally, the loss rate is the sum of the absorption rate and the possible decay rate of *i*, $\ell_i = \sigma_{a,i}\phi+\lambda_i$. Because ϕ is the total flux, σ_x in the above gain and loss rates is spectrum-averaged, i.e.,

$$\sigma_x = \frac{\sum\limits_g \sigma_{x,g}\phi_g}{\sum\limits_g \phi_g} = \frac{1}{\phi}\sum_g \sigma_{x,g}\phi_g \ . \tag{5}$$

The source, gain and loss rates in Eq. (3) are all constant during the burnup step. Their values correspond to the flux at the beginning or at the end of the step; see section 4.1.

VII. 3 THE SOLUTION OF A LINEARIZED BURNUP CHAIN[2]

With all the quantities in Eq. (3) defined, this equation for a linearized chain is solved by first taking its Laplace transform,

$$\tilde{f}(s) = \int_0^\infty f(t)\exp(-st)\mathrm{d}t \ . \tag{6}$$

This gives for the Laplace transform of $N_i(t)$:

$$\tilde{N}_i(s) = \frac{g_{i-1}}{s+\ell_i}\tilde{N}_{i-1}(s) + \frac{1}{s+\ell_i}N_i(0) + \frac{1}{s(s+\ell_i)}y_i \ . \tag{7}$$

Replacing $\tilde{N}_{i-1}(s)$ by a similar expression and repeating this back-substitution until the start of the chain yields

$$\tilde{N}_i(s) = \sum_{k=1}^{i}\left(\frac{N_k(0)}{g_i}\prod_{j=k}^{i}\frac{g_j}{s+\ell_j} + \frac{y_k}{sg_i}\prod_{j=k}^{i}\frac{g_j}{s+\ell_j}\right) . \tag{8}$$

Using partial fractions, the products in Eq. (8) are transformed into sums:

$$\prod_{j=k}^{i}\frac{1}{s+\ell_j} = \sum_{j=k}^{i}\frac{1}{s+\ell_j}\prod_{\substack{n=k\\n\neq j}}^{i}\frac{1}{(\ell_n-\ell_j)} \ . \tag{9}$$

[1] For the (*n*,2*n*) and (*n*,3*n*) reactions, $f_x = 1$.

[2] A detailed derivation has been given by Stamm'ler and Abbate (1983).

Insertion of Eq. (9) into Eq. (8) transforms the latter into

$$\tilde{N}_i(s) = \sum_{k=1}^{i} \left\{ \frac{N_k(0)}{g_i} \sum_{j=k}^{i} \frac{c_{jk}^i}{s+\ell_j} + \frac{y_k}{g_i} \left(\frac{\beta_k^i}{s} - \sum_{j=k}^{i} \frac{c_{jk}^i}{\ell_j (s+\ell_j)} \right) \right\} , \tag{10}$$

where the c and β coefficients are

$$c_{jk}^i = \prod_{n=k}^{i} g_n \Big/ \prod_{\substack{n=k \\ n \neq j}}^{i} (\ell_n - \ell_j) \quad \text{and} \quad \beta_k^i = \prod_{n=k}^{i} g_n \Big/ \prod_{n=k}^{i} \ell_n \ . \tag{11}$$

The β coefficients can also be written as a sum of partial fractions (Lindahl, 1992), using the same formula (9):

$$\beta_k^i = \prod_{j=k}^{i} g_j \sum_{j=k}^{i} \frac{1}{\ell_j} \prod_{\substack{n=k \\ n \neq j}}^{i} \frac{1}{(\ell_n - \ell_j)} = \sum_{j=k}^{i} \frac{c_{jk}^i}{\ell_j} \ . \tag{12}$$

Introducing Eq. (12) into Eq. (10) transforms the latter into

$$\tilde{N}_i(s) = \sum_{k=1}^{i} \left\{ \frac{N_k(0)}{g_i} \sum_{j=k}^{i} \frac{c_{jk}^i}{s+\ell_j} + \frac{y_k}{g_i} \sum_{j=k}^{i} \frac{c_{jk}^i}{\ell_j} \left(\frac{1}{s} - \frac{1}{s+\ell_j} \right) \right\} . \tag{13}$$

The Laplace transform of exp(-at) is $1/(s+a)$, so transformation of Eq. (13) back to the variable t yields the final result at the end of the burnup step T:

$$N_i(T) = \frac{1}{g_i} \sum_{k=1}^{i} N_k(0) \sum_{j=k}^{i} c_{jk}^i \exp(-\ell_j T) + \frac{1}{g_i} \sum_{k=1}^{i} y_k \sum_{j=k}^{i} c_{jk}^i \left(\frac{1-\exp(-\ell_j T)}{\ell_j} \right) . \tag{14}$$

The average value $\overline{N}_i = (1/T) \int N_i(t) \mathrm{d}t$ is found directly from Eq. (14),

$$\overline{N}_i = \frac{1}{g_i T} \sum_{k=1}^{i} N_k(0) \sum_{j=k}^{i} c_{jk}^i \left(\frac{1-\exp(-\ell_j T)}{\ell_j} \right) + \frac{1}{g_i T} \sum_{k=1}^{i} y_k \sum_{j=k}^{i} c_{jk}^i \left(\frac{\exp(-\ell_j T) - 1 + \ell_j T}{\ell_j^2} \right) . \tag{15}$$

Eq. (15) is not only used to get burnup-step averaged yields from Eq. (4), but also to evaluate the average fission energy produced per isotope, and to produce burnup-step averaged number densities for the output, if requested by the User.

When solving Eq. (15), care must be taken to avoid rounding errors. This is done by using England's method, which is discussed in the next section.

VII. 3.1 England's method

In Eq. (14), the two summations over j contain terms which are alternating positive and negative. Physically, a summation over j from k to i means the evaluation of the contribution to $N_i(T)$ from its predecessor k. Obviously, the further back in the chain k lies, the smaller its contribution to $N_i(T)$. Thus, the sums can become vanishingly small, while their individual terms are not small. This does not merely lead to a loss of significant digits, but may even cause large errors in the number densities of isotopes far back in a chain. This has been shown by England (1964), who proposed the following remedy.

Since the function exp() cannot be evaluated with more than N significant digits, the sums cannot be evaluated correctly when they become smaller than $|X_m|\text{x}10^{-N}$, where X_m is the largest individual term in size. This happens with the contribution to $N_i(T)$ from an isotope k far back in the chain. It also can happen between nearby isotopes when the time step is small or the power level is low. The contributions from isotopes even farther back can then be neglected. Thus, a test is made,

$$\sum_{j=k}^{i} c_{jk}^{i}\exp(-\ell_j T) < \left(\max_{j=k\to i}\left|c_{jk}^{i}\exp(-\ell_j T)\right|\right)\times 10^{-N} \ , \qquad (16)$$

where it should be known that all summations in Eq. (14) are run through in reverse direction.

As soon as criterion (16) is fulfilled, contributions to $N_i(T)$ from isotope k and its predecessors are ignored. A similar test is made for the yield contribution. This method not only avoids errors, it can also save computer time by choosing N such that unimportant contributions to $N_i(T)$ are not unnecessarily computed. At present, $N=10$.

VII. 4 BURNUP OF THE SYSTEM

In HELIOS, both regular burnup and cooling by radioactive decay are possible. The case of cooling is straightforward. The cooling period, given in the input, is the time step T for which the linearized chains in all the burnable regions must be solved. Because cooling is done at zero power (actually at 10^{-10} W/g), the test of Eq. (16) reduces the chains only to links with radioactive decay.

During a regular burnup step ΔE, given in the input, the burnup equations must be solved at the specified constant power level P. This corresponds to the time step $T = 86{,}400 \cdot \Delta E/P$. Here, 86,400 is the number of seconds per day, and ΔE is given in units of MWd/t—megawatt-days per metric ton.

Again, the linearized chains must be solved in all the burnable regions for the time step T. However, the solution of the chains, Eq. (14), is obtained with a constant flux, not a constant power level. Because the number densities of the fissionable isotopes change during the burnup step, depletion at constant power is not the same as depletion at constant flux.

Let the system contain I burnable regions, indexed by i. Then, after depleting during T with a constant flux, which at the beginning of the step gives the power level P, the burnup is not ΔE, but $\Delta E'$,

$$\Delta E' = \frac{T}{M}\sum_{i=1}^{I}\phi_i V_i \sum_{iso}\overline{N}_{iso,i}(\kappa\sigma_f)_{iso,i} \ . \tag{17}$$

In Eq. (17), the spectrum-integrated flux ϕ_i, the spectrum-averaged $\kappa\sigma_f$, and the time-step averaged number densities depend on the region i, and M is the heavy-metal mass of the fresh fuel—the numerator of Eq. (2).

If $|1 - \Delta E'/\Delta E| > \delta$ (currently, $\delta = 0.001$), the burnup during T is recalculated with the old flux multiplied by the constant factor $\Delta E/\Delta E'$. If this still yields a burnup that is not good enough, the flux is rescaled again with, in obvious notation, the factor $\Delta E/\Delta E''$. This is done until the fractional error is less than δ, but not more than four times. Usually, two burnup calculations suffice. In Eq. (17), only heavy metals contribute to ΔE; thus, only the heavy-metal chains are solved in these iterations on the exact burnup step.

VII. 4.1 The predictor-corrector method

When depleting from burnup E_1 to E_2, the flux ϕ_1 at E_1 is used, with its spectrum, shape, and level fixed. This also implies that the gain and loss rates are fixed during the burnup step. With the new isotopic number densities thus found, $N_2(\phi_1)$, the flux ϕ_2 at burnup level E_2 is calculated. This corresponds to the upper part of Fig. 9, where it is called the predictor step.

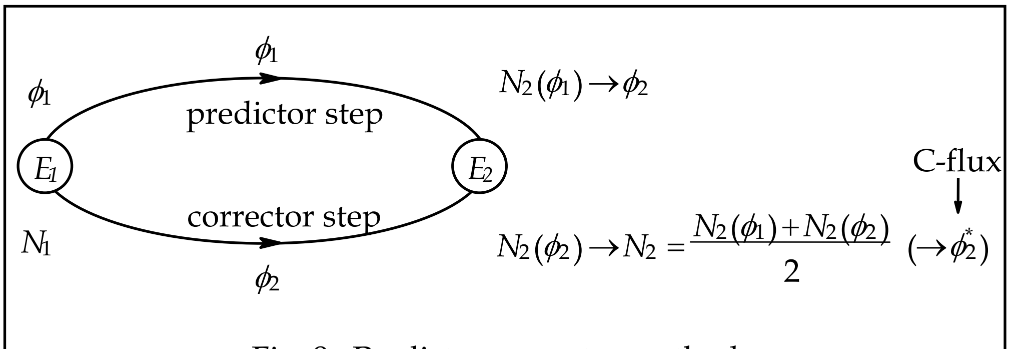


Fig. 9: Predictor-corrector method.

Often, the next burnup step is made with a constant ϕ_2, with $N_2(\phi_1)$ as the initial number densities; and so on. However, in particular when burnable absorbers are present, this process becomes impractical, as very small burnup steps must be taken to justify the assumption of a constant flux spectrum and shape during the burnup steps.

Therefore, in HELIOS, the burnup from E_1 to E_2 is done with a step-averaged flux. Referring to Fig. 9, the burnup from E_1 to E_2 is done again, but with the loss and gain rates pertinent to ϕ_2. This is the corrector step shown in the lower part of Fig. 9.

It produces the number densities $N_2(\phi_2)$. The final number densities at E_2 are then taken as the average of $N_2(\phi_1)$ and $N_2(\phi_2)$. It is these N_2 that are used for the output and as the initial number densities of a possible next burnup step.

To be consistent, the flux at E_2 should be re-evaluated with the final number densities N_2. This flux is called the *C-flux* (from corrector flux). Often, this is not considered necessary, and output and burnup are done with the flux ϕ_2, which is called the *P-flux* (because it is evaluated in the predictor step). The User has full freedom to use P- or C-fluxes anywhere in the calculations.

VII. 4.2 Equilibrium isotopes

Certain isotopes are considered to be in instant equilibrium after a change of power level or after a material change in the system (e.g., a change in void or moving a control rod). Which isotopes can be treated as equilibrium isotopes is determined by the BLOCK DATA subprogram BUDATA and by the isotopes that are present in the nuclear-data library. Currently, these isotopes are I-135 and Xe-135.

The equilibrium concentration of an isotope is obtained from Eq. (3) by setting the left side to zero. The result is:

$$N_i = \frac{y_i + g_{i-1}N_{i-1}}{\ell_i} \ , \tag{18}$$

where for the first isotope in a chain $g_{i\text{-}1}N_{i\text{-}1} = 0$, in which case the yield term must be nonzero.

The flux evaluated with the old number densities of the equilibrium isotopes is called the P-flux. The flux obtained with the new (equilibrium) number densities of the equilibrium isotopes is called the C-flux. This is the same terminology as that of the predictor-corrector method.

Isotopes that are not specified as equilibrium isotopes in BUDATA cannot be treated as such. On the other hand, the User has the option to switch off the special treatment of the equilibrium isotopes.

VII. 5 BREAK ISOTOPES

The number of linearized chains, and also their lengths, can cause the burnup calculations to become costly in computer time. To remedy this, the concept of *break isotopes* is introduced. A break isotope is an isotope that breaks off a chain. However, the cut-off part of the chain is represented by a dummy isotope. During the predictor step, the dummy isotope acts as a collector of the contributions to the cut-off part of the chain. This is achieved by assigning to it a zero initial number density and, during the step, zero yield and loss rates.

During the corrector step, the number density of the dummy isotope at the end of the predictor step, $\Delta N_d(T)$, is passed on to the cut-off part of the chain. This is done by converting $\Delta N_d(T)$ into an (additional) yield of the first isotope of the cut-off part of the chain,

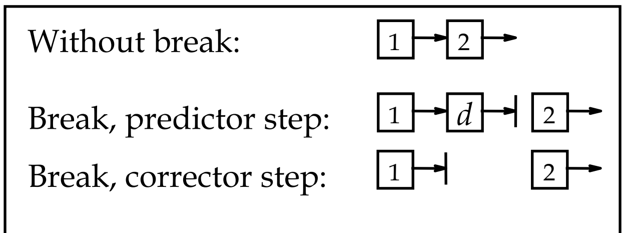


Fig. 10: Illustration of a break isotope.

$$\Delta y = \Delta N_d(T)/T \ . \tag{19}$$

The following example illustrates the principle of break isotopes for a two-isotope chain; see Fig. 10, where *d* is the dummy isotope. According to Eqs. (11) and (14), the exact solution of the chain without break is:

$$\left.\begin{aligned} N_1(T) &= N_1(0)\exp(-\ell_1 T) + y_1\left(\frac{1-\exp(-\ell_1 T)}{\ell_1}\right) \\ N_2(T) &= N_2(0)\exp(-\ell_2 T) + y_2\left(\frac{1-\exp(-\ell_2 T)}{\ell_2}\right) + \\ &\quad + \frac{g_1 N_1(0)}{\ell_2-\ell_1}\big(\exp(-\ell_1 T) - \exp(-\ell_2 T)\big) + \frac{g_1 y_1}{\ell_2-\ell_1}\left(\frac{1-\exp(-\ell_1 t)}{\ell_1} - \frac{1-\exp(-\ell_2 t)}{\ell_2}\right) \end{aligned}\right\} . \tag{20}$$

If the chain is broken, the dummy isotope's density at T is equal to that part of the concentration of $N_2(T)$ that is caused by $N_1(0)$ and y_1, but with $\ell_2 = 0$, i.e.,

$$N_d(T) = \frac{g_1 N_1(0)}{\ell_1}\big(1-\exp(-\ell_1 T)\big) + \frac{g_1 y_1}{\ell_1}\left(\frac{\exp(-\ell_1 t) - 1 + \ell_1 t}{\ell_1}\right) . \tag{21}$$

The $N_d(T)$ thus evaluated is added to y_2, which then contributes to $N_2(T)$ as

$$\left.\begin{aligned} \Delta N_2(T) = {} & g_1 N_1(0)\left(\frac{1-\exp(-\ell_1 T)}{\ell_1}\right)\left(\frac{1-\exp(-\ell_2 T)}{\ell_2}\right) + \\ & + g_1 y_1\left(\frac{\exp(-\ell_1 T) - 1 + \ell_1 T}{\ell_1}\right)\left(\frac{1-\exp(-\ell_2 T)}{\ell_2}\right) \end{aligned}\right\} . \tag{22}$$

The error in $N_2(T)$ caused by the chain interruption is the difference between the exact contribution of $N_1(0)$ and y_1 to $N_2(T)$—the last two terms of Eq. (20)—and the approximate contribution of Eq. (22). Upon expansion of the exponential functions in series, it can be shown that this difference is:

$$\varepsilon = \tfrac{1}{12}(g_1 T)(\ell_1 T)(\ell_2 T)N_1(0) - \tfrac{1}{12}(g_1 T)(\ell_2 T)(y_1 T) \ . \tag{23}$$

Thus, the error caused by the introduction of a break isotope is proportional to the third power of the time-integrated reaction rates—y_1 is also proportional to a reaction rate, the fission rate. The error increases with the length of the time step.

There is another source of error. The contribution to the rest of the chain, $\Delta N_2(T)$, is calculated during the predictor step. During the corrector step, this contribution, multiplied by 2, is passed on as an additional yield of N_2. So the predictor-corrector method is here not applied. Nevertheless, by a judicious choice of break isotopes, the run time of the burnup calculations has been reduced by a factor of about 10.

An advantage of the break isotopes is that feedback in the chains can be treated indirectly. For instance, the feedback from Cm-242 to Pu-238 in the chains of Fig. 1 cannot be described with linearized chains; the involved chain would become infinitely long. However, introducing a break after Pu-238 resolves this situation.

VII. 6 SOME REMARKS ON THE PROGRAM

The burnup calculations involve three modules. One of them, LINK08, is executed only once, at the start of a case. It sets up administrative arrays for the burnup calculations, using the input and the contents of the BLOCK DATA subprogram BUDATA. The other two, LINK17 and LINK18, prepare isotopic reaction rates and perform the actual burnup calculations, respectively.

VII. 6.1 Notes on LINK08

LINK08 processes material input and linearized-chain information; see also chapter II.5.6. Apart from two administrative subroutines, MAIN08 and MATTER (called by MAIN08), and some general subroutines, the following subroutines are used:

SOLVMT: It controls the setup of the administrative burnup arrays and completes the input materials with the isotopes of the burnup chains. It loops through the number of materials and calls the subroutines DENSMT and DEPLMT.

DENSMT: It evaluates for all the fuel materials (all the fuel regions) their initial heavy-metal weights. Prior to that, it converts the input isotope identifiers into their sequence numbers in the nuclear-data library.

DEPLMT: It is called per material and uses the linearized-chain information of BUDATA to complete its isotopes. Isotopes that occur in a linearized chain after an existing isotope, and that do not yet exist, are added to the material's isotopes. Isotopes that do not occur before an existing isotope in a chain are not added. So, fuel that initially contains none of the isotopes from Th-230 up to U-234 does not get these isotopes added; see Fig. 1. It calls, amongst others, OCURMT and BREAKS.

OCURMT: It assigns to each chain isotope its occurrence type; see section 2 and Fig. 8. It is called for every chain that is added to a given material.

BREAKS: It rearranges the isotopes of a burnable material, if it contains break isotopes. The sequence number of a dummy isotope in the library is set to zero, and so is its number density. Originally, such an isotope has the negative identifier of the first isotope of the cut-off part of the chain. The dummy isotope is placed directly after that first isotope.

VII. 6.2 Notes on LINK17

LINK17 has a dual task: it calculates the criticality spectrum (chapter VI) and evaluates the isotopic reaction rates for the burnup chains. Apart from the main administrative subroutine MAIN17, the only burnup-related subroutine is BURRAT.

BURRAT: It calculates the scaling factor P_s of Eq. (2), scales the flux to the input power level, at the same time rebalancing it with the criticality spectrum (chapter VI). Then it evaluates, amongst others, isotopic burnup reaction rates for the chains: $\sigma_a\phi$, $\sigma_f\phi$, $\kappa\sigma_f\phi$, $\sigma_{n2n}\phi$, and $\sigma_{n3n}\phi$.

VII. 6.3 Notes on LINK18

LINK18 performs the burnup calculations. Apart from the main administrative subroutine MAIN18, it contains the following subprograms:

BUDATA: It is a BLOCK DATA subprogram, which loads COMMON/CHAIN/ with CHAIN(20 000). This array contains the linearized chains, including the break isotopes.

BURNER: It determines the time step depending on the type of burnup: regular burnup, C-flux calculation or decay. It also reads/writes the isotopics to/from the working file, evaluates the predictor-corrector number densities, and calls BURNHM for heavy metals—in a loop iterating on ΔE (section 4) and BURNFP for all the other burnable isotopes.

BURNHM: It solves the heavy-metal chains for all the burnable materials by calling ALBION. Further, it evaluates the burnup of the system and of each material according to Eq. (17), as well as the fission rates for the fission yields of Eq. (4), $N_j\sigma_{fj}\phi$. It also calls the subroutine PREBUR.

BURNFP: It solves all the burnup chains, except the heavy-metals chains, for all the burnable materials, by calling ALBION. Before that, it sets up the time-integrated yields of the fission products from T, $y_{i\leftarrow j}$ and the $N_j\sigma_{fj}\phi$ found in BURNHM. It also calls the subroutine PREBUR.

PREBUR: It stores the time-integrated reaction rates (such as ℓ_jT) for either the heavy metals or for the other isotopes of a given material. Also, the three factors containing exp($-\ell_jT$) are stored. All these data are used in BURNHM and BURNFP to prepare, one chain at a time, the input of ALBION. A remark on the $(n,2n)$ and $(n,3n)$ cross-sections is here due.

The absorption and scattering cross-sections in the nuclear-data library include the (*n*,2*n*) and (*n*,3*n*) cross-sections, as follows:

$$\sigma_a \to \sigma'_a = \sigma_a - \sum_k (k-1)\sigma_{(n,kn)}; \quad \sigma_s \to \sigma'_s = \sigma_s + \sum_k k\sigma_{(n,kn)} \ . \tag{24}$$

This representation preserves the total cross-section and the number of secondaries, which is the number of neutrons emerging from a collision, or

$$c = \left(\nu\sigma_f + \sum_k k\sigma_{(n,kn)} + \sigma_s\right)/\sigma \ . \tag{25}$$

In the library, the spectrum of the emerging neutrons, normalized to 1, has been preserved by redefining it as follows:

$$f_{s,g'\leftarrow g} = f'_{s,g'\leftarrow g} = \frac{\sigma_{sg} f_{s,g'\leftarrow g} + \sum_k k\sigma_{(n,kn),g} f_{(n,kn),g'\leftarrow g}}{\sigma_{sg} + \sum_k k\sigma_{(n,kn),g}} \ . \tag{26}$$

In PREBUR, the true loss rates are obtained as the losses due to true absorption plus those due to (*n*,2*n*) and (*n*,3*n*) reactions,

$$\ell = (\sigma'_a + \sum_k k\sigma_{(n,kn)})\phi \ . \tag{27}$$

The true gain rates must be due to either capture or decay,

$$g = (\sigma'_a + (k-1)\sum_k k\sigma_{(n,kn)} - \sigma_f)\phi \quad \text{or} \quad \lambda \ . \tag{28}$$

ALBION: It solves the burnup equation for a linearized chain. Its results are both the number densities at the end of the step *T* and their averaged values over *T*; Eqs. (13) and (14). This subroutine got its name because it uses the method of England.

REFERENCES

ENGLAND T R (1964) *CINDER – A One-Point Depletion and Fission Product Program,* Westinghouse Report WAPD-TM-334 (Revised), Bettis Atomic Power Laboratory, Pittsburgh, Pennsylvania..

FREDIN B (1991) *Comprehensive Set of Fission Product Chains for Use in Accurate Fuel Depletion Calculations*, ABB Atom Report BR 91-722/NDP-91-03, Västerås, Sweden.

FREDIN B (1992) *Recommended completion of the fission product chains of proposed comprehensive model including recent improvements in fission product lumping,* ABB Atom Report BR 92-532/NDP-92-100 Rev 2 (1994), Västerås, Sweden.

LINDAHL S-Ö (1992) Private communication, ABB Atom, Västerås, Sweden.

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

VIII. HELIOS: OUTPUT

VIII. 1 INTRODUCTION AND OBJECTIVES

As shown in Fig. 1 of chapter I, HELIOS gets its input from the HERMES data-base and saves its output in the same data-base. This data-base has a hierarchical structure—the location of a data list is defined by a path of catalog (directory) names and the name of the list itself. The contents of each data list consist of two parts: header and data. The header contains information on the list, such as its size, creation date, and the program by which it has been created. Per list, the data must be of the same type: integer, real or character.

Clearly, an input consisting of data lists is not easy to set up for the User. Therefore, the input processor AURORA was written to convert the User's input into data lists in the data-base. Its input is described in the User Manual of AURORA. For the same reason, the output processor ZENITH was written to extract the User's output from the output data lists. Its input is described in the User Manual of ZENITH.

The basic objective of the output is that <u>any quantity that can be evaluated</u> from input data and the basic HELIOS results should be available to the User. Another objective is that HELIOS should be affected as little as possible by unforeseen output requirements. Therefore, almost all output editing is done by ZENITH, which not only accepts formulas as input, but also additional information. For instance, gamma smearing, detector sensitivities, tables and maps of any quantity (and their format) can be specified in the ZENITH input. Also, results of different cases can be compared, and properties of special regions or isotopes can be listed—e.g., the fast fluxes in the three pins nearest to the detector of the ten pins with the ten highest Pu-239 concentrations.

In principle, it is enough to store, at each reactivity point, the basic HELIOS results in the data-base. These results are: (i) particle fluxes in all regions and all library groups; (ii) particle currents at all interface sectors (also specular) and all library groups; (iii) number densities of all isotopes in all regions; and (iv) microscopic cross-sections of the resonance isotopes in the resonance groups for all the regions where they occur. However, the number of these basic results can easily grow too large for the data-base. That is why some elementary output processing is done by HELIOS - to allow the User to significantly reduce the amount of output to the data-base.

A special output is the <u>restart dumps</u> of number densities and related information of certain (fuel) materials for later use. These data are also stored in data lists, though in a data-base that does not have to be the same as the regular data-base that is used for input and output (I/O).

The output data that can be stored in the data-base are described in section 2, while section 3 discusses the evaluation of these data by homogenization (region averaging) and condensation (group collapsing). Section 4 describes the data of the restart dumps. Section 5, finally, sketches the subroutines that are used to produce the HELIOS output and restart dumps.

VIII. 2 DATA SAVED IN THE I/O DATA-BASE

There are eight data categories stored in the data-base for later use by ZENITH. They are discussed in the sections 2.1–2.8, where the following conventions are used in the description of the hierarchical paths to their data lists.

Each path is given as a string of catalog names, separated by backslashes (\) and terminated by the list name. User-specified catalog names are in italics, and the root catalog is indicated by the number symbol (#). The beginning of each path is '#\HELIOS*case*', with *case* the name of the CASE operator.

The list names, which are not User-specified, are followed by '(*c*)', '(*i*)', or '(*r*)', to indicate character, integer, or real data. If a data list depends on the reactivity points, an extra catalog is created per reactivity point. Its name, *rpp*, directly precedes the list name. *rpp* is the ASCII representation of the integer output identifier, which is 10,000 times the sequence number of the PATH or TREE operator (see the AURORA User Manual) + the sequence number of the reactivity point in that operator. It is negative for TREE operators.

The data-base, which is a so-called HERMES file, can be inspected, copied, converted to ASCII, and modified with the auxiliary program HFCARE.

VIII. 2.1 AURORA's final input

Both the AURORA input and output (which is the HELIOS input) are stored in the data-base. Only two data lists of the AURORA input are used by ZENITH. They are not used or produced by HELIOS, but are mentioned here for completeness' sake:

#\HELIOS*case*\AINPUT\KAROPF(*i*),

#\HELIOS*case*\AINPUT\FINPUT(*c*).

<u>KAROPF()</u> contains the number of characters of each input operator in the final input. For ZENITH, only the CASE operator, the first input operator of AURORA, is of importance. This operator contains special character strings that may be used by the ARRAY operator of ZENITH. For details, see the descriptions of the CASE and ARRAY operators in the User Manuals of AURORA and ZENITH, respectively.

<u>FINPUT()</u> is the final ASCII input, with blanks and comments eliminated, and with the include sets and parameters resolved. Using KAROPF(1), ZENITH reads the entire CASE operator, and extracts the special character string(s).

VIII. 2.2 Status output

The status output is the only output that is always produced by HELIOS. It contains a few global data for each of the reactivity points, which are stored in the following two data lists:

#\HELIOS*case*\ZINPUT\STATUX\rpp\ZSTNAM(*c*),

#\HELIOS*case*\ZINPUT\STATUX\rpp\ZSTAT(*r*).

<u>ZSTNAM()</u> contains *path name* and *state name*, which are the names of the PATH or TREE operator, and the STATE operator for which the data are given; see the User Manual of AURORA.

<u>ZSTAT()</u> contains the following real data:

LPATH	The number of characters of *path name*.
LSTAT	The number of characters of *state name*.
RPP	The output identifier corresponding to the catalog name *rpp*.
XRP	Sequence number of the calculation at the same *path name*, *state name* and *UBURN* ($XRP = 0, 1, ...$). Non-zero values of *XRP* indicate additional calculations that do not involve burnup, e.g., after a cooling step (radioactive decay), or the extra step that puts the equilibrium isotopes in equilibrium.
UBURN	User-specified burnup (MWd/t) in *path name*.
PBURN	Actual burnup (MWd/t) in *path name*, which is very near to *UBURN*, unless very large burnup steps are taken.
TBURN	True burnup (MWd/t) in *path name*; it depends on the individual materials' histories, so it may differ from *PBURN*.
PTIME	The time (s) in *path name* up to the actual reactivity point; zero for TREE operator output.
PTIMX	The time (s) in *path name* up to the preceding reactivity point; zero for TREE-operator output.
TDAYS	The time (s) of the actual decay step; zero if there is no decay.
PWLVL	The power level in units of W/g, i.e., watts per gram of heavy metal in fresh fuel.
PSCALE	The scaling factor between the flux of the transport solution and the flux at power.
EIGV	The eigenvalue of the transport calculation; see also chapter VI.
EIGVB2	The eigenvalue of the system for an input buckling, *B2*; see also chapter VI.
B2	Input buckling (cm^{-2}). Its associated spectrum is superimposed on the transport spectrum, instead of the criticality spectrum; see also chapter VI.
KINF	The infinite multiplication factor of the system, i.e., the ratio of fission-neutron production to absorption; see also chapter VI.
KINFB	The infinite multiplication factor of the system, either with the criticality spectrum or the spectrum associated with *B2* imposed; see also chapter VI.
BSQ	The material buckling (cm^{-2}) of the homogenized system, without possible border leakage; see also chapter VI.
MSQ	The migration area (cm^2) of the homogenized system, excluding possible border leakage; see also chapter VI.

VIII. 2.3 Group output

The amount of output is reduced by group condensation, as shown in section 3. HELIOS can produce condensed data in all the group structures that have been defined by the GROUP operators of AURORA. For neutrons, each GROUP operator creates two types of data lists, one of which is per reactivity point, while for gammas only one data list is created. The lists are:

#\HELIOS*case*\ZINPUT\GROUP\group name\ZGRDAT(*r*),

#\HELIOS*case*\ZINPUT\GROUP\*group name**rpp*\ZDRAT(*r*).

ZGRDAT() contains the number of groups, which is negative for gammas, and the lower energy limits in eV, NGL and $E_1, \dots E_{|NGL|}$. This list, which is independent of the reactivity points, is stored only once.

ZDRAT() is for neutrons only; it is stored for each reactivity point. It contains, for the materials of the entire system homogenized into one material, the ratios of the B_1 diffusion coefficients (see chapter VI) to the diffusion coefficients obtained from the total transport-corrected cross-sections (XSs) as $1/3\Sigma$, i.e.

$$3\Sigma D_{B_1}(1 \rightarrow |NGL|). \tag{1}$$

These ratios provide global correction factors that can be used for parts of the system; see also section 3.2.

VIII. 2.4 Area output

The amount of output is reduced by limiting the output to areas. Areas are combinations (homogenizations) of one or more, not necessarily coherent, regions. Instead of storing $\kappa\Sigma_f$ for all regions, it is enough to store them for the fuel regions of the pins. For pins with more than one fuel region, these regions may be homogenized into one fuel region so that there are as many areas as fuel pins. If data for an entire fuel assembly are needed, there is just one area; all the regions of the assembly combined.

The areas are defined by the AREA operators of AURORA. Per AREA operator, there is one data list which is independent of the reactivity points:

#\HELIOS*case*\ZINPUT\AREA\area name\ZARDAT(*r*).

ZARDAT() contains the following data:

NRA	The number of areas of *area name*.
$X_1, \dots X_{NRA}$	The x coordinates of the gravity centres of the NRA areas.
$Y_1, \dots Y_{NRA}$	The y coordinates of the gravity centres of the NRA areas.
$V_1, \dots V_{NRA}$	The volumes, per cm height, of the NRA areas.

The centres of gravity of the areas are used by ZENITH to locate the data of these areas in maps. Their coordinates are calculated as the volume-weighted sum of the corresponding coordinates of the regions that make up the area; see Eq. (II.2).

VIII. 2.5 Face output

The output amount is reduced by limiting the output to faces. Faces are combinations of one or more, not necessarily coherent, segments that are used in the current coupling. Although currents at specular segments are not used in the coupling, these currents are registered, too (see chapter IV.2.2), so that specular segments may be included in faces.

As opposed to the AREA operator, which can define more than one area, the FACE operator of AURORA defines a single face. Since it is a geometric operator, its only data list is independent of the reactivity points. That list is:

#\HELIOS*case*\ZINPUT\FACE\face name\ZFADAT(*r*).

ZFADAT() contains the following data: *X*, *Y* and *S*. These are the *x* and *y* coordinates of the centre point (gravity centre), and the surface, per cm height, of the face. Observe that sectors do not occur in the definition of a face, so it is not possible to get output currents per sector, only sector-integrated partial (in or out) currents through segments.

VIII. 2.6 Macro output

Utilizing group structures and areas, the output of macroscopic cross-sections (XSs) is reduced by requesting them only in group structures and areas where they are needed. A further output reduction is achieved by requesting only those data types that are needed, e.g., the (n,γ) production matrix is seldom necessary.

The MACRO operator of AURORA controls which material properties are stored in the data-base. The information is in four data lists, two of which are for administration and do not depend on the reactivity points. The lists are:

#\HELIOS*case*\ZINPUT\MACRO\macro name\ZMAINF(*i*),

#\HELIOS*case*\ZINPUT\MACRO\macro name\ZMANAM(*c*),

#\HELIOS*case*\ZINPUT\MACRO\*macro name**rpp*\ZMAFLX(*r*),

#\HELIOS*case*\ZINPUT\MACRO\*macro name**rpp*\ZMACDT(*r*).

ZMAINF() contains the following integer data:

LGR	The number of characters of the name *group* of the GROUP operator that defines the output groups.
LGRNG	The number of characters of the name *grng* of the GROUP operator that defines the emitting neutron-group structure for (n,γ) output. If no (n,γ) output is requested, $LGRNG = 0$.
LAR	The number of characters of the name *area* of the AREA operator that defines the output areas.
NDT	The number of data types for which output is requested.
NGL	The number of output groups, negative for gamma output.

NNG	The number of emitting neutron groups for (n,γ) output. If no (n,γ) output is requested, $NNG = 0$.
NRA	The number of output areas.

ZMANAM() contains the following name information:

group	The name of the GROUP operator defining the output groups.
grng	The name of the GROUP operator that defines the emitting neutron group structure for (n,γ) output. If no (n,γ) output is requested, *grng* is not given.
area	The name of the AREA operator defining the output areas.
dt_1	The name of the first data type whose output will be stored.
⋮	
dt_{NDT}	The name of the last data type. The data types are evaluated as shown in section 3. They are denoted by two characters:

bu	burnup, only for neutrons and in one group;
dn	delayed-neutron data, only for neutrons;
ed	kinetic energy deposited by scattering (and absorption for γ's), only in one group;
tr	transport XS, $\Sigma_g = \Sigma_{ag} + \Sigma_{0,g} - \Sigma_{1,g}$, twice (see section 3.1);
ab	absorption XS, Σ_{ag};
fi	fission XS, Σ_{fg}, only for neutrons;
nf	fission-neutron production XS, $\nu\Sigma_{fg}$, only for neutrons;
kf	fission-energy production XS, $\kappa\Sigma_{fg}$, only for neutrons;
ch	fission spectrum, χ_g, only for neutrons;
p0	P_0-scattering matrix, $\Sigma_{0,g'\leftarrow g}$, transport-corrected for g's;
p1	P_1-scattering matrix, $\Sigma_{1,g'\leftarrow g}$;
p2	P_2-scattering matrix, $\Sigma_{2,g'\leftarrow g}$;
p3	P_3-scattering matrix, $\Sigma_{3,g'\leftarrow g}$;
ng(*gr*)	(n,γ) production matrix, the name of the emitting neutron group structure must be appended in parentheses.

ZMAFLX() contains the fluxes and, if requested and possible (see the AURORA User Manual), their time-averaged values during the preceding burnup step. The fluxes are not multiplied by the region volumes. These data are given per group and per area, with the group index varying fastest, i.e.,

$$\phi(1 \rightarrow |NGL|, 1 \rightarrow NRA) \quad [\text{and } \bar{\phi}(1 \rightarrow |NGL|, 1 \rightarrow NRA)] \; . \tag{2}$$

ZMACDT() contains the data types, followed by their time-averaged values during the preceding burnup step—if requested and possible. These data are arranged as follows:

$$\left.\begin{array}{c} dt_1\,([1 \rightarrow |\,NGL|\,,]1 \rightarrow NGX, 1 \rightarrow NRA) \\ \vdots \\ dt_{NDT}\,([1 \rightarrow |\,NGL|\,,]1 \rightarrow NGX, 1 \rightarrow NRA) \\ \\ [\text{and } \overline{dt}_1(\cdots)\cdots\overline{dt}_{NDT}(\cdots)] \end{array}\right\} , \qquad (3)$$

where $NGX = |NGL|$ for scattering matrices, $NGX = NNG$ for the (n,γ) matrix, and $NGX = 1$ otherwise.

VIII. 2.7 Micro output

Utilizing group structures and areas, the output of microscopic cross-sections (XSs) is reduced by requesting them only in group structures and areas where they are needed. A further reduction is achieved by requesting only those data types that are needed for the isotopes of interest.

The MICRO operator of AURORA controls which isotopic data are stored in the data-base. The information is in five data lists, three of which do not depend on the reactivity points. The lists are:

#\HELIOS*case*\ZINPUT\MICRO\micro name\ZMIINF(*i*),

#\HELIOS*case*\ZINPUT\MICRO\micro name\ZMINAM(*c*),

#\HELIOS*case*\ZINPUT\MICRO\micro name\ZMIISO(*r*),

#\HELIOS*case*\ZINPUT\MICRO\*micro name**rpp*\ZMIFLX(*r*),

#\HELIOS*case*\ZINPUT\MICRO\*micro name**rpp*\ZMICDT(*r*).

<u>ZMIINF()</u> contains, in addition to the seven integer data of ZMAINF(), an eighth datum, *MISO*, which is the number of isotopes for which output will be stored in the data-base. If these isotopes are not present in some, or all, of the regions of an area, they are temporarily set to 10^{-20} (per barn–cm). In that way, reaction rates of detector isotopes, which are not present in the transport and burnup calculations, can be obtained.

<u>ZMINAM()</u> contains the same name information as ZMANAM(), though now for the MICRO operator—the data types *bu*, *dn* and *ch* cannot occur.

<u>ZMIISO()</u> contains information on the isotopes whose output will be produced. Each isotope has a code indicating its type, i.e., the XSs it has available in the nuclear-data library.[1] The interpretation of the code, which is listed in the Table below, allows another reduction of the output amount—only existing data are stored, no zeroes for non-existing data.

[1]Although the library is not stored in the data-base, most of the library data can be made available by HELIOS/ZENITH through the MICRO output. Some exceptions are: $(n,2n)$ and $(n,3n)$ XSs, decay constants, fission spectra, and fission yields.

code	*scatter*	*fission*	*γ data*
1	yes	yes	yes
2	yes	yes	no
3	yes	no	yes
4	yes	no	no
5	no	no	yes
6	no	no	no

The data in the list ZMIISO() are:

$id_1, \ldots id_{MISO}$	Ten times the library identifiers + their codes (see Table above).
$aw_1, \ldots aw_{MISO}$	The atomic weights in amu.

ZMIFLX() contains the initial heavy-isotope densities (g/cm^3), the fluxes and the number densities per barn–cm, and possibly time-averaged fluxes and number densities. These data are arranged as follows:

$$\left.\begin{array}{r}\rho(1\rightarrow NRA),\phi(1\rightarrow|NGL|,1\rightarrow NRA),N(1\rightarrow NRA,1\rightarrow MISO)\\ [\text{and } \bar{\phi}(1\rightarrow|NGL|,1\rightarrow NRA),\bar{N}(1\rightarrow NRA,1\rightarrow MISO)]\end{array}\right\}. \tag{4}$$

ZMICDT() contains the data types, followed by their time-averaged values during the preceding burnup step—if requested and possible. These data are arranged in the same way as those of ZMACDT(), however, now they are also given per isotope, i.e.,

$$\left.\begin{array}{r}dt_1([1\rightarrow|NGL|,]1\rightarrow NGX,1\rightarrow NRA,1\rightarrow MISO)\\ \vdots\qquad\qquad\qquad\qquad\qquad\\ dt_{NDT}([1\rightarrow|NGL|,]1\rightarrow NGX,1\rightarrow NRA,1\rightarrow MISO)\\ \\ [\text{and } \overline{dt_1}(\cdots)\cdots\overline{dt_{NDT}}(\cdots)]\end{array}\right\}. \tag{5}$$

VIII. 2.8 Current output

Utilizing group structures and faces, the output of the partial currents can be reduced by requesting them only in group structures and at faces where they are needed. The CUR operator of AURORA controls the current output to the data-base. The information that is stored is in three lists, two of which do not depend on the reactivity points. The lists are:

#\HELIOS*case*\ZINPUT\CUR\cur name\ZCUINF(*i*),

#\HELIOS*case*\ZINPUT\CUR\cur name\ZCUNAM(*c*),

#\HELIOS*case*\ZINPUT\CUR\*cur name**rpp*\ZCURDT(*r*).

ZCUINF() contains the following integer data:

LGR	The number of characters of the name *group* of the GROUP operator that defines the output groups.
LFA_1	The number of characters of the first face name in the CUR operator of the faces for the current output.
⋮	
LFA_{NFA}	The number of characters of the last face name in the CUR operator of the faces for the current output.
NGL	The number of output groups; negative for gamma output.
NFA	The number of faces in the CUR operator; for each face output currents will be produced.

ZCUNAM() contains the following name information:

group	The name of the GROUP operator defining the output groups.
fa_1	The first face name in the CUR operator.
⋮	
fa_{NFA}	The last face name in the CUR operator.

ZCURDT() contains the out- and in-currents through the faces and in the group structures specified in the CUR operator—time-averaged currents are possible, too. The direction of these currents, out or in, is with respect to the space element(s) for which the segments making up a face have been defined. A segment that is defined from node 1 to node 2 belongs to the space element to the right of the direction 1→2; the current to the left is then an out-current. The data in this list are arranged as follows:

$$\left.\begin{array}{c} J^{+}(1 \to |NGL|, 1 \to NFA), \\ J^{-}(1 \to |NGL|, 1 \to NFA) \\ [\text{and } \overline{J^{+}}(\cdots), \overline{J^{-}}(\cdots)] \end{array}\right\} . \tag{6}$$

VIII. 3 EVALUATION OF THE DATA TYPES

This section presents the formulas used to evaluate the data types stored by HELIOS in the HERMES data-base. The lower case indexes *i, s,* and *g* denote regions, segments, and groups of the transport calculation. The upper case indexes *I, S,* and *G* denote areas, faces, and collapsed groups. The index *iso* denotes isotopes. Fluxes and currents are not volume- or surface-integrated.

Area volumes and face surfaces are obtained by summation (integration):

$$V_I = \sum_{i \in I} V_i \quad \text{and} \quad S_S = \sum_{s \in S} S_s \ . \tag{7}$$

Initial heavy-metal densities and number densities, ρ and N, are volume-averaged:

$$\rho_I = \frac{1}{V_I}\sum_{i\in I}\rho_i V_i \quad \text{and} \quad N_{iso,I} = \frac{1}{V_I}\sum_{i\in I} N_{iso,i} V_i \ . \tag{8}$$

Fluxes and currents are group-integrated and volume-averaged:

$$\phi_{IG} = \frac{1}{V_I}\sum_{i\in I}\sum_{g\in G}\phi_{ig} V_i \quad \text{and} \quad J^{\pm}_{SG} = \frac{1}{V_I}\sum_{s\in S}\sum_{g\in G} J^{\pm}_{sg} S_s \ . \tag{9}$$

The macroscopic XSs *tr*, *ab*, *fi*, *nf*, and *kf* are flux-volume averaged, while their microscopic counterparts are also averaged with the isotopic densities. With x and X being the generic microscopic and macroscopic XS,

$$X_{ig} = \sum_{iso} N_{iso,i}\, x_{iso,ig} \ , \tag{10}$$

and their flux-volume (and number-density) averaged values are:

$$X_{IG} = \frac{\sum_{i\in I}\sum_{g\in G} X_{ig}\phi_{ig} V_i}{\phi_{IG} V_I} \quad \text{and} \quad x_{iso,IG} = \frac{\sum_{i\in I}\sum_{g\in G} N_{iso,i}\, x_{iso,ig}\,\phi_{ig} V_i}{N_{iso,I}\,\phi_{IG} V_I} \ . \tag{11}$$

The macroscopic and microscopic matrices *p0*, *p1*, and *ng* are averaged as the XSs and are also integrated over the receiving groups:

$$\left.\begin{aligned} X_{I,G'\leftarrow G} &= \frac{\sum_{i\in I}\sum_{g'\in G'}\sum_{g\in G} X_{i,g'\leftarrow g}\,\phi_{ig} V_i}{\phi_{IG} V_I} \quad \text{and} \\ x_{iso,I,G'\leftarrow G} &= \frac{\sum_{i\in I}\sum_{g'\in G'}\sum_{g\in G} N_{iso,i}\, x_{iso,i,g'\leftarrow g}\,\phi_{ig} V_i}{N_{iso,I}\,\phi_{IG} V_I} \end{aligned}\right\} \ . \tag{12}$$

Of course, for data type *ng*, the emitting groups, g and G, are neutron groups, while the receiving groups are gamma groups.

The fission spectrum per region is weighted with a preset estimate of $\nu\Sigma_f\phi$ according to Eq. (II.39). The corresponding data type *ch* is weighted with number densities and volumes, and group-integrated:

$$\chi_{IG} = \frac{\sum_{g\in G}\sum_{i\in I}\sum_{iso} w_{iso} N_{iso,i} V_i\, \chi_{iso,g}}{\sum_{g}\sum_{i\in I}\sum_{iso} w_{iso} N_{iso,i} V_i\, \chi_{iso,g}} \ . \tag{13}$$

The area burnup, E_I, is averaged with the initial heavy-metal weights,

$$E_I = \frac{\sum_{i \in I} \rho_i V_i E_i}{\sum_{i \in I} \rho_i V_i}, \quad \text{where} \quad E_i = \sum_b \Delta E_{ib} \ . \tag{14}$$

In Eq. (13), the burnup of a region, E_i, is the sum over the burnup increments, ΔE_{ib}, to which it has been exposed. Using Eq. (VII.17), and recalling that the predictor-corrector method and flux rescaling are used during a burnup step, ΔE_{ib} can be written as:

$$\Delta E_{ib} = \frac{T_b}{2\rho_i V_i} \sum_x \sum_g \sum_{iso} \overline{N}^x_{iso,ib} (\kappa\sigma_f)_{iso,g} f^x_b \phi^x_{ig,b} V_i \ . \tag{15}$$

In Eq. (15), the superscript x denotes the predictor or the corrector step, f^x_b is the final flux rescaling factor [see the discussion after Eq. (VII.17)], and T_b is the time in seconds of the b^{th} burnup step.

Delayed-neutron data are stored in the data-base for six precursor groups, d. The library contains, per fissionable isotope, the fractions of all fission neutrons emitted per fission β_d that appear from each precursor group, d. These fractions, summed per material and averaged with the fission-neutron production rate, are stored in the data-base as data type *dn*; they are given by

$$\beta_{IG,d} = \frac{\sum_{i \in I} \sum_{g \in G} \sum_{iso} N_{iso,i} (\nu\sigma_f)_{iso,g} \beta_{iso,d} \phi_{ig} V_i}{\sum_{i \in I} \sum_{g \in G} \sum_{iso} N_{iso,i} (\nu\sigma_f)_{iso,g} \phi_{ig} V_i} \ . \tag{16}$$

The $\beta_{IG,d}$ of Eq. (16) can be processed further by ZENITH to obtain effective delayed-neutron yields; see section 3.3.

The formula for the rate at which the kinetic energy of the particles is deposited in an area, e_I, depends on the type of particle. For neutrons it is the kinetic energy lost (or gained) by scattering. For gammas, it is the scattering energy lost plus the energy of the gammas that are absorbed. If E_g is the lethargy midpoint of group g, then the formula for e_I is:

$$e_I = \frac{1}{V_I} \sum_{i \in I} \sum_g \sum_{iso} N_{iso,i} \phi_{ig} V_i \left(\sum_{g'} (E_g - E_{g'}) \sigma_{iso,g' \leftarrow g} + \overset{\text{gammas only}}{\overline{E_g \sigma_{iso,a,g}}} \right) . \tag{17}$$

VIII. 3.1 Leakage averaging of transport XS and P_1 matrix

An alternative way to condense transport XSs and the P_1-scattering matrix is by current-averaging (chapter V.5 of Stamm'ler and Abbate, 1983). However, the currents are not available in all regions. Instead, for what it is worth, HELIOS evaluated also leakage-volume averaged values of the data types *tr* and *p1*. In the data lists, the values of each of these data types came directly after the values of their regular flux-volume averaged values. This averaging has been eliminated since version 1.9. Still, in version 1.9, two identical sets of flux-averaged data were produced for *tr* and *p1*. But since version 1.10 there is no second set of *p1* output.

VIII. 3.2 The diffusion coefficient

The only diffusion coefficients that HELIOS produces are a by-product of the B_1 calculations, the D_{B_1} of Eq. (VI.23). They are always for the entire system and are stored in the data-base as the ZDRAT() output of expression (1). It is up to the User to produce diffusion coefficients from the data-base output and appropriate formula input to ZENITH. Although this is not a matter of the HELIOS output, a few possible approaches are outlined below.

For both particle types, the trivial definition of the diffusion coefficient is

$$D_{IG} = 1/(3\Sigma_{IG}) \ , \tag{18}$$

where Σ is the transport-corrected total XS. This definition requires the above formula as input to ZENITH, using flux-volume averaged Σ_{IG}'s—the first or the second set of data type *tr* in the ZMACDT() output.

Equation (18) suggests the use of another average, that of $1/\Sigma$ instead of Σ. While HELIOS does not do this averaging, it still is possible with ZENITH. This is done by requesting *tr* output in a group structure whose groups h contain the groups G as a subset, and in areas j that together make up the area I. Then the formula to be given to ZENITH as input must be Eq. (11) with X replaced by $1/\Sigma$ and i and g by j and h.

A third approach is the use of diffusion coefficients consistent with the B_1 method. For the entire system this is straightforward in any group structure: use flux-volume averaged *tr* output for the entire system in the desired group structure and give ZENITH an input formula that solves D_{B_1} from (1).

How to get diffusion coefficients consistent with the B_1 method for parts of the system is described in section 3.3. This is not always recommendable, nor straightforward for small systems. In general, the B_1 method is applied to systems larger than just a fuel pin, its cladding, or a control blade. It is difficult to see how sensible B_1 calculations can be done for systems without fissionable material.

But it is possible, for parts of the system, to adjust diffusion coefficients calculated according to Eq. (18) to the D_{B_1} of the entire system. Remember that the ZDRAT() output of expression (1) is the ratio of D_{B_1} to the trivial diffusion coefficient, with Σ flux-volume averaged. So (1) is a factor that adjusts the trivial diffusion coefficient of the entire system to D_{B_1}. This same factor could be used to obtain B_1-adjusted diffusion coefficients for parts of the system.

VIII. 3.3 Sundries

The $\beta_{IG,d}$ of Eq. (16) must still be flux-averaged over all the macrogroups G to obtain the effective delayed-neutron yields in the area I, $\beta_{I,d}$. Moreover, they also must be corrected for the fact that the fission spectrum of delayed neutrons is softer than that of the prompt neutrons. This is often done by multiplying them by the ratio of the delayed-neutron multiplication factor to the total multiplication factor. The final formula is

$$\beta_{I,d}^{eff} = \frac{k_I^x}{k_I} \frac{\sum \beta_{IG,d} (\nu\Sigma_f)_{IG} \phi_{IG} V_I}{\sum (\nu\Sigma_f)_{IG} \phi_{IG} V_I} \quad \text{with} \quad k_I = \frac{\sum (\nu\Sigma_f)_{IG} \phi_{IG}}{\sum (\Sigma_{a,IG} + D_{IG} B^2) \phi_{IG}} \; . \tag{19}$$

The summations in Eq. (19) are over all macrogroups G. The formula for k_I^x is the same as for k_I, except that the summations over G are over the groups below about 0.45 MeV. The D_{IG} are the B_1 diffusion coefficients.

The reason that data type *dn* contains group-dependent $\beta_{IG,d}$ values is that HELIOS cannot produce the delayed-neutron multiplication factor. ZENITH can do this, provided that data type *dn* is in a group structure that has at least a cutoff near 0.45 MeV. In fact, ZENITH offers the possibility to produce directly effective delayed-neutron yields; see its User Manual.

Normally, the area I is the entire system and $k_I = 1$, so long as the criticality spectrum has been enforced in HELIOS; see chapter VI. However, the system can be a set of fuel assemblies, while delayed-neutron data are required for the individual assemblies. Then, each assembly needs a B_1 calculation to obtain the quantities used in Eq. (19).

Because HELIOS cannot perform B_1 calculations for parts of the system, ZENITH has the option to do the B_1 calculation in a single area in any group structure. HELIOS stores the required data types automatically in the data-base if *dn* output has been requested. To be meaningful, B_1 calculations that produce a criticality spectrum in part of the system should be performed in a sufficient number of groups, and the area must have at least one region with fissionable material. Because this option also produces the ZDRAT() output of expression (1), it can be used to adjust diffusion coefficients to the D_{B1} of parts of the system, not to the D_{B1} of the entire system.

VIII. 4 DATA SAVED IN RESTART DUMPS

In a restart dump, the isotopics of a set of User-specified regions (materials), at User-specified burnups, can be saved in a HERMES data-base. In fact, more than one set can be saved, at the same or different burnups, in the same or another data-base. This is described in *User Manual AURORA*.

To each of the DUMP input operators correspond three data lists that do not depend on burnup and two catalog types, *state* and *rpp* (for the meaning of *rpp*, see Chapter VIII.2). The number of *state* and *rpp* catalogs is equal to the number of states and reactivity points in the PATH from which the dump had been made. Each *state* catalog contains two state-independent data lists (in a sense, these lists are burnup-dependent since a PATH may contain more than one state), while each *rpp* catalog contains three burnup-dependent data lists. The structure of the data-base for a given

DUMP, *dump*, is shown below, followed by a description of the contents of the data lists.

```
                         |-- ZDUINF(i): LAR, LPATH, LAREA, NRA
                         |
                         |-- ZDUNAM(c): area, path
                         |
                         |-- ZDUARE(c): contents of area
                         |
                         |            |-- ZDUDIM(i): NDISO, NBA
                         |-- state -- |
                         |      :     |-- ZDUISO(i): isotope identifiers
                         |      :
                         |      :     |--
#\HELIOS\case\DUMP\dump --|-- state -- |
                         |            |--
                         |
                         |            |-- ZDUBUR(i): LSTAT, 1000*UBURN
                         |            |
                         |            |-- ZDUSTA(c): state
                         |            |
                         |--  rpp  -- |                 |- RHO(1→NRA)
                         |      :     |                 |
                         |      :     |                 |- E(1→NRA)
                         |      :     |-- ZDUDAT(r):-   |
                         |      :                       |- DNSDUM(1→NDISO)
                         |      :                       |
                         |      :                       |- FRBA(1→NBA)
```

<u>ZDUINF()</u> contains the following integer data:

LAR	The number of characters of the name *ar* of the AREA operator that defines the regions whose restart data are dumped.
LPATH	The number of characters of the name *path* of the PATH operator from which restart data are dumped.
LAREA	The number of characters of the entire input of the AREA operator *ar*, i.e., "*ar*=*AREA*(*aid*, … *aid*)"—see *User Manual AURORA*.
NRA	The number of regions (materials) defined by *ar*.

<u>ZDUNAM()</u> contains *arpath*, i.e., the names of the AREA and PATH operators combined in a single text string, padded to the right with blanks until the next multiple of four characters.

<u>ZDUARE()</u> contains the text string "*ar*=*AREA*(*aid*, … *aid*)", padded to the right with blanks until the next multiple of four characters (=*LAREA*).

<u>ZDUDIM()</u> contains the number of isotopes of *state* that are dumped, *NDISO*, and the number of BAs with effective cross-section tables among them, *NBA*.

<u>ZDUISO()</u> contains library identifiers of all the involved isotopes, BAs with effective absorption cross-section tables included.

<u>ZDUBUR()</u> contains the following real data:

LSTAT	The number of characters of the name *state* of the state, on *path*, whose restart data are dumped—see *User Manual AURORA*.
UBURN	The User burnup at *path* of the actual dump, in kWd/t.

ZDUSTA() contains *state*, i.e., the name of the actual state on *path*.

ZDUDAT() contains the initial heavy-metal densities, the true burnups, the number densities of the *MISO* isotopes, and the remaining fractions of the *NBA* BA isotopes. These data are given for all the regions of *ar*, i.e.,

$$\left.\begin{array}{rcl} \rho_{ini}(1 \rightarrow NRA) & - & [\mathrm{g/cm^3}] \\ E(1 \rightarrow NRA) & - & [\mathrm{MWd/t}] \\ N(1 \rightarrow NDISO) & - & [\#/\mathrm{cm^3}] \\ f_{BA}(1 \rightarrow NBA) & - & [0 \le f_{BA} \le 1] \end{array}\right\} . \tag{20}$$

VIII. 5 SOME REMARKS ON THE PROGRAM

The output processing involves two modules, LINK09 and LINK20. LINK09 is executed once, at the start of each case. It sets up the administrative arrays for the output processing, using as input the AURORA-processed User input that is stored in the data-base. LINK20, which can be executed more than once after the transport calculations, performs the actual output processing.

VIII. 5.1 Notes on LINK09

LINK09 interprets the AURORA-processed input of the edit operators. Apart from the administrative subroutine MAIN09, an error-message routine, and some general subroutines, it involves 23 subroutines. They set up administrative output arrays (also for restart dumps), create data-base catalogs, and store the data lists that do not depend on the reactivity points in their data-bases. The remainder of this section sketches these subroutines.

CHANAL:	It is called at the start of MAIN09 and finds the maximum number of chains in any AREA operator and the total number of chains in all AREA operators to do the variable dimensioning in MAIN09.
DBSTAX:	It creates the catalog STATUX of section 2.2. The catalogs per reactivity point, *rpp*, which come under this catalog, as well as their data lists, are created and stored in LINK20.
GREDIT:	It sets up administrative arrays for each GROUP operator and evaluates their energy arrays with E_g and (E_g–$E_{g'}$) for Eq. (17).

DBGRUP: It creates the catalogs GROUP and *group name* and stores the data list ZGRDAT() in the data-base; see section 2.3. The *rpp* catalogs which come under these catalogs, as well as their data lists, are created and stored in LINK20.

AREDIT: It sets up the administrative arrays for the AREA operators, including ZARDAT() of section 2.4. It also resolves the chains that define the HELIOS regions that make up the areas, for which it calls the subroutines LINKSS, LINKST, LINKRG and DECLIN.

DBMACR: It sets up the administrative arrays for the macro output, including ZMAINF() and ZMANAM() of section 2.6, which are stored in the data-base. This creates the MACRO and *macro name* catalogs. The *rpp* catalogs which come under these catalogs, as well as their data lists, are created and stored in LINK20.

PREMIC: It sets up administrative arrays for the micro output, in particular isotope-related arrays.

DBMICR: It completes the construction of administrative arrays for the micro output that was begun in PREMIC, including the data of ZMIINF(), ZMINAM and ZMIISO() of section 2.7, which are saved in the database. This creates the catalogs MICRO and *micro name*. The *rpp* catalogs which come under these catalogs, as well as their data lists, are created and stored in LINK20.

FACEDT: It sets up the administrative arrays for the FACE operators, including ZFADAT() of section 2.5. It also resolves the chains that define the sectors that make up an output face, for which it calls the subroutines LINKSS, LINKST and LINKNN—the former two subroutines are also called by AREDIT.

DBARFA: It stores the area and face data lists, ZARDAT() and ZFADAT(), in the data-base; see sections 2.4 and 2.5. This creates at the same time the catalogs AREA, *area name*, FACE and *face name*.

DBCURX: It sets up the administrative arrays for the CUR operators, and constructs the data lists ZCUINF() and ZCUNAM() of section 2.8, which it stores in the data-base. This creates the catalogs CUR and *curname*. The *rpp* catalogs which come under these catalogs, as well as their data lists, are created and stored in LINK20.

DBDUMX: It sets up the administrative arrays for the DUMP operators, and the burnup-independent data lists, ZDUINF(), ZDUNAM(), ZDUARE(), ZDUDIM() and ZDUISO() of section 4, which are stored in their databases. This creates the catalogs DUMP and *dump*. The *rpp* catalogs and their data lists are created and stored in LINK19.

LINKSS: It resolves the subsystems of an AREA chain; see the User Manual of AURORA. It is called by AREDIT and FACEDT.

LINKST: It resolves the structures of an AREA chain; see the User Manual of AURORA. It is called by AREDIT and FACEDT.

LINKRG: It resolves the CCS-es and regions of an AREA chain; see the User Manual of AURORA. It calls the subroutine GETREG which provides, finally, the region numbers. It is called by AREDIT and FACEDT.

LINKNN: It resolves the nodes of an AREA chain and from these determines the sectors that make up a face. It is called by FACEDT.

GETREG: It obtains the involved region numbers at the end of a chain. It is called by LINKRG.

DECLIN: This general subroutine, here called by AREDIT, orders the regions indicated by a chain according to the decreasing indicators of the (homogenized) areas to which they belong. It is similar to the subroutine DECREA of chapter IV.6.2, which is for real numbers.

GETNAM: It obtains the User-defined name of an input operator; see the User Manual of AURORA. It is called by all the DB**** subroutines; also, those of LINK20.

NAMINT: It packs names of different sizes into a single string, padded with blanks to the next multiple of four characters. It is called by DBCURX, DBMACR, DBMICR and DBSTAT/LINK20.

OUTP9A: It prints test output of the administrative arrays related to the output operators GROUP, AREA, MACRO and MICRO.

OUTP9B: It prints test output of the administrative arrays related to the output operators FACE and CUR.

DBEXPO: It prints the contents of a data list and its catalog path, and is called from all the DB**** subroutines, except DBSTAX.

VIII. 5.2 Notes on LINK19

LINK19 is executed after each transport calculation, whenever one or more restart dumps have been requested. Apart from the administrative subroutine MAIN19 and a few general subroutines to access the scratch file and a HERMES data-base, it contains only one subroutine.

DBDUMP: It creates and stores the contents of the burnup-dependent data lists—ZDUBUR(), ZDUSTA() and ZDUDAT()—for the dumps that have been requested at the actual reactivity point. Their corresponding *rpp* catalogs are then also created.

VIII. 5.3 Notes on LINK20

LINK20 is executed after each transport calculation to produce the output per reactivity point and to store it in the data-base. It is also called after each predictor and corrector step to build up time-averaged output, if requested. Apart from the administrative subroutine MAIN20 and a few general subroutines, it involves 19 subroutines. They perform the output processing, create data-base catalogs, and store the data lists that depend on the reactivity points in the data-base. The remainder of this section outlines the functions of these subroutines.

SETISM:	It constructs an array that associates output isotopes with materials, and an array with their number densities.
DIMDET:	It finds the dimensions of temperature-interpolation arrays of *detector isotopes*. These are output isotopes that are not present in the materials, but whose infinitely-dilute effective XSs have been requested in one or more MICRO operators; see section 2.7.
INTDET:	It constructs indexes and coefficients for temperature interpolations in the library of detector isotopes.
RHODNS:	It stores volume-integrated initial heavy-isotope densities and number densities in ZMIFLX(), i.e., the numerators of Eqs. (8); see also section 2.7. This is done for neutrons or gammas at the actual reactivity point, or after a predictor or corrector step to build up time-averaged values.
NEUEDT:	It fills ZMAFLX(), ZMACDT(), ZMIFLX(), and ZMICDT() with volume and flux-volume integrated neutron data, the numerators of Eqs. (9), (11–13) and (16–17). These data are built up per group by calling the subroutines SIGMA5, SIGMA6, MACED1, and MICED1—data type *bu*, the burnup, is already completed. This is done at the actual reactivity point or after a predictor or corrector step when building up the time-averaged values.
GAMEDT:	It does for gammas what NEUEDT does for neutrons, though only for data types that are relevant for gammas.
SIGMA5:	It evaluates for all the materials the macroscopic XSs of their individual isotopes in one energy group. These are factors of the individual terms in the sums in the numerators of Eqs. (11–13) and (16–17). They are used in MICED1. Per material, they are also summed over the isotopes and passed on to MACED1.
SIGMA6:	It does the same as SIGMA5, but now for the detector isotopes associated with the materials, and only for data types that can be specified in the micro input. These isotopes have a number density of 10^{-20} per barn–cm. Nothing is passed on to MACED1.
SIGMA7:	It does the same as SIGMA5, but now for gammas, and only for data types that are relevant for gammas.

SIGMA8:	It does the same as SIGMA6, but now for gammas, and only for data types that are relevant for gammas.
MACED1:	It completes the individual terms of the numerators of Eqs. (9), (11–13), and (16–17), and sums them for the macro output. The summation over the groups is done by calling MACED1 per group. It also evaluates the burnup.
MICED1:	It does the same as MACED1, but now for the micro output, so there is no summation over the isotopes.
MACED2:	It completes the macro output, i.e., ZMAFLX() and ZMACDT() of Eqs. (2-3). It is done for neutrons and gammas, both for regular and time-averaged data.
MICED2:	It completes the micro output, i.e., ZMIFLX() and ZMICDT() of Eqs. (4-5). It is done for neutrons and gammas, both for regular and time-averaged data.
CUREDT:	It evaluates the data of ZCURDT() of Eq. (6), according to Eqs. (7) and (9). This is done for neutrons or gammas at the actual reactivity point, or after a predictor or corrector step when building up time-averaged values. It is done for all CUR operators.
DBSTAT:	It fills ZSTAT() with data (see section 2.2) and stores it in the data-base for the actual reactivity point. The corresponding *rpp* catalog is then also created.
DBDRAT:	It evaluates the ZDRAT() of Eq. (1) and stores it in the data-base for the actual reactivity point. The corresponding *rpp* catalog is then also created. It is done for all GROUP operators.
DBMAMI:	It stores two data lists with macro or micro output in the data-base for the actual reactivity point. These lists are ZMAFLX() and ZMACDT() or ZMIFLX() and ZMICDT(). The corresponding *rpp* catalogs are then also created. It is done for all MACRO and MICRO operators.
DBCURR:	It stores the current output, ZCURDT(), in the data-base for the actual reactivity point. The corresponding *rpp* catalog is then also created. It is done for all CUR operators.

REFERENCES

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

IX. HELIOS: THE LIBRARY

IX.1 INTRODUCTION AND DATA GENERATION

This chapter outlines the data generation for the neutron and gamma cross-section library of HELIOS, and describes the structure and contents of the direct-access file that contains this library. This library has the same structure as the PHOENIX library, although it has been expanded to contain additional HELIOS-specific data. Therefore, much of the material presented here has been taken from an ASEA-ATOM report describing the PHOENIX library (Stamm'ler, 1985).

The structure of the HELIOS library described here applies from the original version *hy94-1a* (Ferri, 1994) through later releases, *hy98-1a* (Ferri, 1996, 1998), until the present release. The only major additions are, since mid-1996, subgroup constants (see chapter III) and since end-2001, resonance-scattering data.

IX.1.1 Generation of the library

Almost all the data in the library are based on the ENDF/B-VI data files (Rose and Dunford, 1990). The exceptions are the cross-sections of the erbium and thulium isotopes, and the (n,γ) matrices of many fission products. Their evaluation is briefly discussed at the end of this section.

Figure 1 shows the flow scheme of the library generation. The ENDF/B-VI files are processed with a version of NJOY91.13—with updates through version 91.105—(MacFarlane *et al.*, 1991) that includes the code RABBLE as a module (Decher, 1994). The main task of NJOY is to generate infinite-dilution neutron and gamma cross-sections in 190 neutron and 48 gamma groups, the "smooth" data, which are contained in GENDFs—group-evaluated nuclear-data files. Another NJOY task is to generate PENDFs—which are files with cross-sections in many thousands of energy points—which are input to RABBLE. While all isotopes must be processed with NJOY, only those to be treated as resonance isotopes by HELIOS must be processed by RABBLE. This is shown in the top part of Fig. 1.

The GENDF group cross-sections and the resonance-shielded cross-section produced by RABBLE are collected for all the isotopes in a single hierarchical HERMES file, the master data-base. The creation of this data-base is one of the activities of the HEBE code (Ferri, 1996). Figure 2 shows the structure of the data-base—not shown are the data lists in the SPECTRA catalogs that contain different flux spectra for group-collapsing the cross-section data.

Reverting to Fig. 1, another activity of HEBE is to construct a (master) library for HELIOS or PHOENIX from selected data in the (master) data-base. The structure and contents of this library, and of condensed libraries, are described in the remaining sections of this chapter. With the master library, HELIOS runs are made for small systems to generate flux spectra in different regions, at different temperatures, and at different burnups.

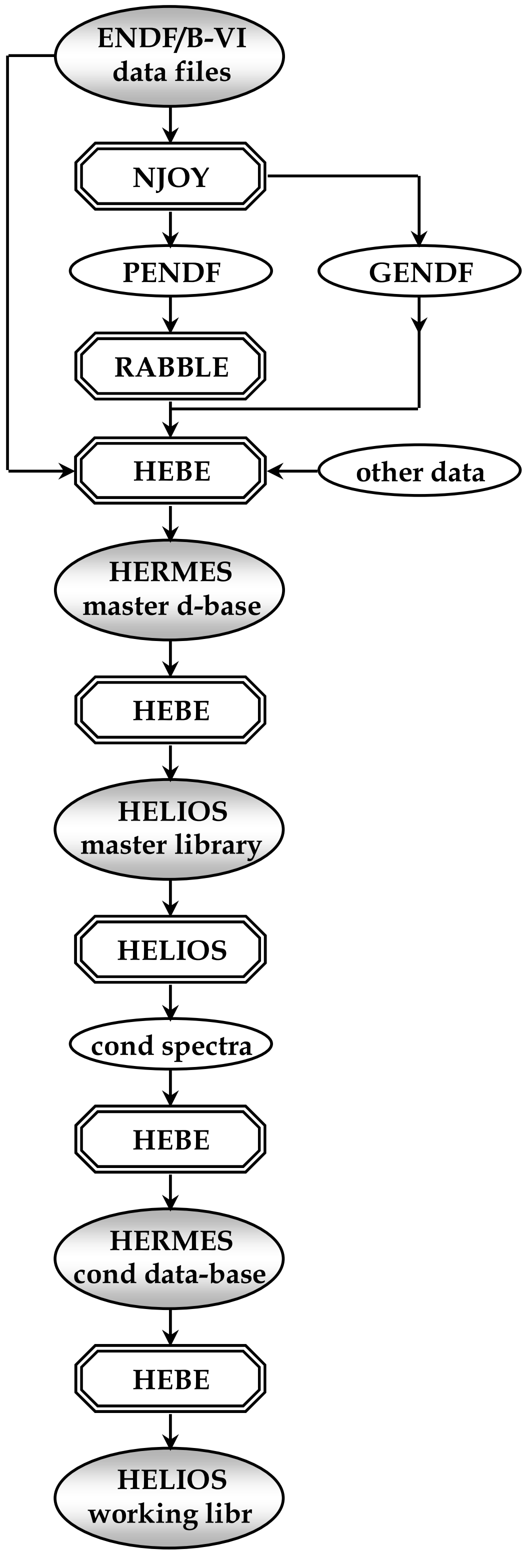


Fig.1: Flow scheme of library generation.

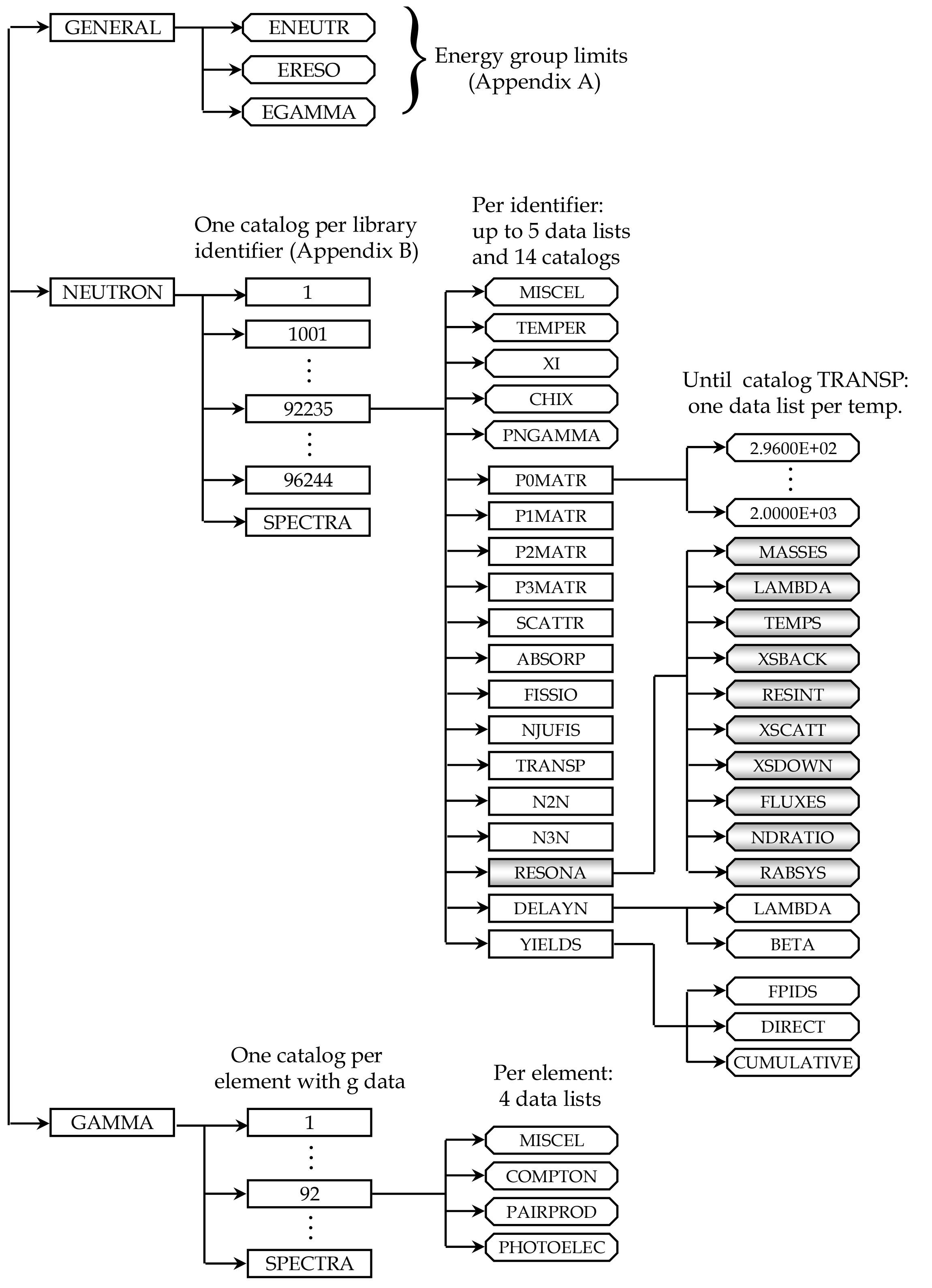


Fig 2: Catalogs and data lists in the HERMES data base.

In a third activity, HEBE adds these spectra to the (master) data-base. In a fourth activity, they are used to group-collapse the data-base to a condensed data-base, from which a condensed library is made. Presently, three libraries exist: the master library with 190/48 neutron/gamma groups; a fast-reactor library with 112/18 neutron/ gamma groups; and a production library with 47/18 neutron/gamma groups. The group structures are shown in Appendix A.

Erbium and thulium data have not been taken from the ENDF/B-VI files. Instead, they have been obtained as described by Sing and Jonsson (1993). In the thermal and resolved-resonance energy regions, below 2 keV, the cross-sections of the erbium and thulium isotopes are constructed by the RABBLE code from resonance parameters given by Mughabghab *et al.* (1984). In the unresolved resonance and fast energy regions, smooth cross-sections are taken from curves given by Mughabghab, *et al*.

The (n,γ) matrices of many fission products do not exist in ENDF/B-VI. For these fission-product isotopes, a procedure developed by Lipiec (1994) is used to produce their (n,γ) data. It is based on a technique of Perkins, *et al.* (1975) and has been added to the special NJOY version as a separate module, the PHHR module.

IX.2 LIBRARY IDENTIFIER AND DIMENSIONING DATA

The first part of the library consists of 100 data. Apart from the library identification, these data are: basic dimensioning data, such as the number of groups, isotopes, etc.; start addresses for the main sections of the library; and the sizes of data arrays whose lengths cannot be directly derived from the basic dimensioning data. In detail:

(*LIBID*(*I*),*I*=*1*,8)	8A4 text with the library identification. It is usually of the type: **HELIOS LIBRARY: 11 AUG 1994**
I0	Total number of data in the library. If the library contains also gamma data, they start at address *I0* – 1.
I1	Size of *RSTAB1*()—temperature and sigma-potential entries for all the resonance tables, scattering included..
I2	Size of *RESDAT*() per group—smooth resonance data.
I3	Size of *RSTAB2*() per group—absorption and $\nu\cdot$fission tables.
I4	Size of *XTEMP*() and *ISTART*().
I5	Size of *XSBAFR*().
I7	Maximum size of *XSDATA*() in any group.
I8	Size of *P1TEMP*().
I10	Maximum size of *PnDATA*() (n = 1,2,3) in any group.

INDRES	Start address for resonance data—absorption and ν–fission tables and subgroup constants; see also *INDRSC* below.
INDXSD	Start address for cross-section data.
INDBUR	Start address for burnup data.
INDP1M	Start address for P_1 data (negative if there are gamma data, too).
NOG	Number of groups.
NOFG	Number of fast groups (above the resonance range).
NORG	Number of resonance groups.
NOTG	Number of groups with up-scattering. In the library, a cutoff limit of ≥ 2 eV has been chosen and is enforced by the library program. Thus, $NOFG+NORG+NOTG=NOG$ does not necessarily hold.
NCHI	Number of groups with fission source *CHI*().
NELT	Number of isotopes in the library, including burnable absorbers.
NELR	Number of isotopes with complete cross-section sets, including fission and P_0 scattering (transport-corrected).
NRES	Number of isotopes with tabulated resonance integrals. They belong to the *NELR* isotopes above.
NBUR	Number of burnup isotopes. Only heavy metals and fission products are counted here; the other burnable isotopes, like B-10 and burnable absorbers (BAs), are recognized by HELIOS in another way.
NBA	Number of burnable-absorber isotopes with tables of effective absorption cross-sections as a function of the remaining BA fraction, generated by a program like FOBUS (Fredin and Stamm'ler, 1977).
NFIS	Number of fissionable isotopes.
NP1	Number of isotopes with P_1 (P_2, P_3) scattering data; they belong to the *NELR* isotopes above.
INDN2N	Start address for $(n,2n)$ data.
INDN3N	Start address for $(n,3n)$ data.
N2N	Number of $(n,2n)$ isotopes.
N3N	Number of $(n,3n)$ isotopes.

INDCHX	Start address for fission spectra.
NCHIX	Number of isotopes with a fission spectrum. The other isotopes get the fission spectrum $(CHI(G),G=1,NCHI)$.
INDP2M	Start address for P_2 data (0 if no P_2 data are available). For PHOENIX, *INDP2M* = 0.
INDP3M	Start address for P_3 data (0 if no P_3 data are available). For PHOENIX, *INDP3M* = 0.
INDSG	Start address for subgroup data. If 0, HELIOS evaluates the subgroup data in the subroutine LIBRES. For PHOENIX, *INDSG* = 0.
...	This position is reserved for PHOENIX.
INDKAP	Start address for energy-per-fission data for neutrons at zero incident energy, κ. If $\neq 0$, HELIOS uses these values to evaluate κ as a function of energy. If =0, averaged κ-values, stored at index *INDBUR*, are used.
INDRSC	Start address for resonance-scattering and down-scattering data.
*IR*3	Size of *RSTAB3*() per group—scattering and down-scattering tables.
…	54 zeroes (as yet not used).

IX.3 GENERAL DATA

Before presenting the general data, two observations should be made:

– All fission spectra are set to zero as soon as >99.99% of their intensity is reached and then normalized to one.

– The isotope identifiers $EID()$ are arranged such that:
$EID(N)$ is a resonance isotope if $N \leq NRES$;
is an isotope with a complete cross-section set if $N \leq NELR$;
is an isotope with only absorption cross-sections if $N > NELR$;
is a BA isotope if $N > NELT - NBA$.

The general data start right after the identification and dimensioning data, i.e., at address 100 (addressing begins at 0). They are:

$(U(G),G=1,NOG)$	Group lethargies, upper limits—see Appendix A.
$(CHI(G),G=1,NCHI)$	Fission spectrum, normalized to 1.
$(EID(N),N=1,NELT)$	Isotope identifiers—see Appendix B.

$(IDCODE(N), N=1, NELT)$	0 - not burnable (nor fissionable), 1/2/3 - burnable and not/fast/fully fissionable.
$(AW(N), N=1, NELT)$	Atomic weights in atomic mass units.
$(ALFA(N), N=1, NELR)$	$\alpha=(m\text{-}1)^2/(m+1)^2$, with m the atomic weight (amu).
$(DELTA(N), N=1, NELR)$	$\Delta=\text{-}\ln(\alpha)$
$((BATEXT(I,N), I=1,20), N=1, NBA)$	20 A4 explanatory text for each BA isotope.

IX.4 FISSION-SPECTRUM DATA

The fission spectrum data start at address *INDCHX*. This position contains the first element of the fission-spectrum index array for the *NCHIX* isotopes that have a fission spectrum.

$((INFCHX(K,NF), K=1,3), NF=1, NCHIX)$

$INFCHX(1,NF)$	Library sequence numbers of the isotopes that have a fission spectrum; it points at $EID()$.
$INFCHX(2,NF)$	Absolute address in the library file of the first component of *NF*'s fission spectrum; it corresponds to the first neutron group.
$INFCHX(3,NF)$	Absolute address in the library file of the last component of *NF*'s fission spectrum; it corresponds to the last neutron group with a non-negligible fission spectrum. Observe that no isotope may have more than *NCHI* (section IX.2) components of the fission spectrum.

$$(CHIX(K), K = \min_{NF=1\to NCHIX}(INFCHX(2,NF), \max_{NF=1\to NCHIX}(INFCHX(3,NF))$$

Fission spectra of all the *NCHIX* isotopes.

So the fission spectrum of the N^{th} isotope, with $N=INFCHX(1,NF)$, starts at address $INFCHX(2,NF)$ and ends at $INFCHX(3,NF)$:

$$((CHIX(G,NF), G=1, INFCHX(3,NF)-INFCHX(2,NF)+1) \;. \tag{1}$$

IX.5 $(n,2n)$ AND $(n,3n)$ DATA

The $(n,2n)$ cross-section data start at address *INDN2N*. They are given for a few isotopes only:

$(IDN2N(N2), N2=1, N2N)$	Library sequence numbers of the $(n,2n)$ isotopes;

and per group G:

$(XSN2N(N2), N2=1, N2N)$	The $(n,2n)$ cross-sections of *N2N* isotopes.

The $(n,3n)$ cross-section data start at address *INDN3N*. They are given for a few isotopes only. Similarly to the $(n,2n)$ data, one has:

$(IDN3N(N3), N3=1, N3N)$	Library sequence numbers of the $(n,3n)$ isotopes;

and per group G:

$(XSN3N(N3), N3=1, N3N)$	The $(n,3n)$ cross-sections of *N3N* isotopes.

IX.6 RESONANCE DATA

There are three categories of resonance data: (i) the traditional resonance-integral (RI) data for absorption and $\nu\cdot$fission; (ii) since version 1.4, their subgroup constants; and (iii) since version 1.7, RI data for scattering and down-scattering. Prior to version 1.4, the RI data are for homogeneous mixtures of hydrogen and the resonance isotope, while in later libraries most RI data are based on heterogeneous calculations.

IX.6.1 Smooth data and resonance integrals for absorption and $\nu\cdot$fission

The first address of the resonance-integral data is *INDRES*. These data are divided into group-independent and group-dependent data. The former contains addressing information, and the temperature and sigma-potential entries for the RI tabulations. The group-dependent data are located directly after the group-independent data, and their structure is repeated *NORG* times, once per resonance group. They contain the so-called smooth resonance data for all the *NELR* isotopes, immediately followed by the RI tables. The data are:

$(NTEMP(N), N=1, NRES)$	$NTEMP(N) = 100 * NTADR + NTRES\ .$ (2) For resonance isotope *N*, the start address *NTADR* of the *NTRES* resonance temperatures *T* in *RSTAB1*().
$(NPOT(N), N=1, NRES)$	$NPOT(N) = 100 * NPADR + NPRES\ .$ (3) As above, but for the potential XS, σ_p.

$(NTAB(N), N=1, NRES)$

$NTAB(N)$ is the start address in $RSTAB2()$ of the RI tables of isotope N. The lengths of these tables are $NTRES*NPRES$; twice that if the isotope is fully fissionable, i.e., $IDCODE(N)=3$.

$(RSTAB1(I), I=1, I1)$

$$I1 = \sum_{N=1}^{NRES} (NTRES(N) + NPRES(N)) \ . \tag{4}$$

Here, all temperature and sigma-potential entries of the RI tables are given. In fact, the data stored are $\sqrt{T}$ and $\sqrt{\sigma_p}$.

The remaining data are repeated for each of the *NORG* resonance groups:

$(RESDAT(I), I=1, I2)$

$$I2 = 4*NELR \ . \tag{5}$$

The smooth resonance data for all the *NELR* isotopes. Specifically, for isotope *N*:

$\lambda\sigma_p$	for $I2=4N-3$;
σ_s	for $I2=4N-2$;
$\xi\sigma_s/\Delta u$	for $I2=4N-1$;
ν	for $I2=4N$.

$(RSTAB2(I), I=1, I3)$

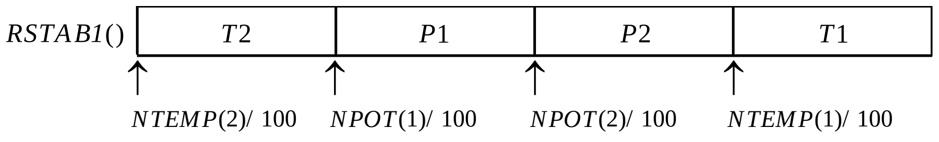
$$I3 = \sum_{N=1}^{NRES} (\delta_N * NTRES(N) * NPRES(N)) \ , \tag{6}$$

$$\text{with } \begin{cases} \delta_N = 2 & \text{if } IDCODE(N) = 3; \\ \delta_N = 1 & \text{otherwise.} \end{cases}$$

These tables contain absorption and $\nu\cdot$fission RIs, divided by du; listed per isotope, per reaction type (absorption first, then $\nu\cdot$fission), per temperature, and per sigma-potential.

Explanatory notes

- Example of addressing in $RSTAB1()$ for two resonance isotopes:



The sizes of the four blocks are, from the left to the right: MOD($NTEMP$(2),100), MOD($NPOT$(1),100), MOD($NPOT$(2),100) and MOD($NTEMP$(1),100).

- The smooth resonance data are as follows; for their use, see Stamm'ler and Abbate (1983):

$\lambda\sigma_p$	Goldstein's intermediate resonance factor times the microscopic potential cross-section;
σ_s	microscopic scattering cross-section;
$\xi\sigma_s/\Delta u$	microscopic slowing-down power divided by the lethargy width;
ν	average number of neutrons released per fission.

- All RI tables that belong to an isotope are coherent in *RSTAB2*(); the values are arranged according to increasing temperature and sigma-potential. Per isotope, the RI tables do not have to occur in the same sequence as in *EID*(). In *RESDAT*(), however, the data are strictly sequenced according to *EID*().

IX.6.2 Resonance-scattering data

The first address of the resonance-scattering data is *INDRSC*. If there are no resonance-scattering data in the library, *INDRSC* is zero. In that case, HELIOS will use in the resonance groups the "smooth" scattering data which, for each isotope, have been shielded with a typical sigma-potential, σ_p. The data are:

((*XSBFIX*(*G*,*N*),*G*=1,*NORG*+1),*N*=1,*NRES*)	It contains, per resonance isotope, the number-density ratio *N*(hydrogen)/*N*(isotope) for which its resonance XSs are tabulated and its corresponding background XSs in the *NORG* resonance groups. If its RIs are based on homogeneous calculations, it is a true number-density ratio; otherwise, it is an equivalent one. In the library, square-root values of these quantities are stored.
(*NTABRS*(*N*),*N*=1,*NRES*)	*NTABRS*(*N*) is the start address in *RSTAB*3() of the resonance-scattering tables of isotope *N*. The size of these tables is *NTRES*∗*NPRES* for scattering and the same for down-scattering.
(*RSTAB*3(*I*),*I*=1,*IR*3)	$IR3 = \sum_{N=1}^{NRES}\left(2 * NTRES(N) * NPRES(N)\right).$ These tables contain resonance-shielded scattering and down-scattering cross sections. They are listed per isotope, per reaction type (scattering first, then down-scattering), per temperature, and per sigma-potential.

IX.6.3 Subgroup data

The first address of the subgroup data is *INDSG*. If there are no subgroup data in the library, *INDSG* is zero. In that case, HELIOS will evaluate all the data that can be accessed under *INDSG*. Otherwise, these data are:

NSUBG	Number of resonance subgroups (the same for absorption and fission). These are the subgroups, indexed by n, in Chapter III of this manual.
NFLX	Number of resonance flux calculations per group and category. These are indexed by m in Chapter III, so $NFLX \leq NSUBG$.
NXSG	Size of *XSG*() per group.
NWSG	Size of *WSG*() per group.
NWASGF	Size of *WASGF*() per group.

Then follow arrays with subgroup data. (Chapter III of this manual.)

NWADD()	Per resonance isotope, its first positions in *XSG*(), *WSG*(), and *WASGF*().
XSG()	Subgroup cross-section sets, of *NSUBG* data each, for absorption (and ν–fission) per isotope and per resonance group.
WSG()	Subgroup weight sets of *NSUBG* data each, for absorption (and ν-fission) per temperature, per isotope, and per resonance group.
XASGF()	Subgroup absorption XS sets of *NFLX* data each, per isotope, and then per resonance group plus one (if the entire resonance range is treated as one "group").
WASGF()	Subgroup weight sets of *NFLX* data each, per temperature, per isotope, and per resonance group, and once again per resonance group (if the entire resonance range is treated as one "group").

IX.7 CROSS-SECTION DATA

The first address of the cross-section data is *INDXSD*. The cross-section data are divided into group-independent and group-dependent data. Apart from addressing information, the former contain, per isotope, the temperatures for which its cross-sections are given, and for BA the fractions of their original concentration (at zero burnup $fr = 1$) for which their effective absorption cross-sections are given. The group-dependent data are repeated for each of the *NOG* groups. Besides addressing information, they contain the cross-section data. For the first *NELR* isotopes they are, for group *G*,

$$\sigma_a,\ \sigma_f,\ \nu\sigma_f,\ \sigma_{tr},\ \sigma_s,\ \sigma(GMIN \to G), \dots\ \sigma(GMAX \to G)\ , \tag{7}$$

where *GMIN* and *GMAX* are the first and last group with non-zero in-scattering.

For the remaining isotopes, only σ_a is given. However, for BA a set of effective cross-sections at the various BA fractions is given:

$$\sigma_a \quad \text{or, for BA} \quad \hat{\sigma}_a(fr=0), \dots \hat{\sigma}_a(fr=1)\ . \tag{8}$$

Observe that the following convention has been used:

$$\sigma(G' \to G) = \begin{cases} \sigma^0(G' \to G) & \text{for } G' \neq G, \quad \mathrm{P}_0 \text{ scattering matrix,} \\ \sigma^0(G \to G) - \Delta\sigma, & \text{transport - corrected diagonal;} \end{cases} \tag{9}$$

$$\sigma_s = \sigma_s^0 = \sum_{G'} \sigma^0(G \to G')\ ; \tag{10}$$

$$\sigma_{tr} = \sigma_a + \sum_{G'} \sigma(G \to G')\ ; \tag{11}$$

The transport correction, $\Delta\sigma$ in Eq.(9), is either the default outflow correction or—for H and D in water, Be, C, O and Na—the inflow correction:

$$\left.\begin{array}{ll} \Delta\sigma = \sigma_s^1 = \sum_{G'} \sigma^1(G \to G') & \text{outflow correction} \\ \Delta\sigma = \sum_{G'} \sigma^1(G' \to G) J_{G'} / J_G & \text{inflow correction} \end{array}\right\}. \tag{9a}$$

In Eq.(9a), $\sigma^1(G \to G')$ is the P_1 scattering matrix and J_G are the group currents in a finite homogeneous system of the isotope with a number density of $0.01 \times 10^{24}\ \mathrm{cm}^{-3}$, obtained by solving the P_1 equations. To simulate an almost critical system, leakage is described by a small buckling, $B^2 = 2.5 \times 10^{-5}\ \mathrm{cm}^{-2}$. These calculations are done by the code HEBE while constructing the library.

The cross-section data are:

$(NXST(N), N=1, NELT)$ $\qquad NXST(N) = 100 * NXADR + NXSS\ .$ (12)

For isotope *N,* the start position *NXADR* of its temperatures in *XSTEMP*() and of its group-dependent addressing information (one number for each temperature tabulation) in *ISTART*(). *NXSS* is the number of temperatures.

$(NXSBA(N), N=1, NBA)$ $\quad NXSBA(N) = 100 * NXBAD + NXBAS \; . \qquad (13)$

$NXBAD$ is the start address of the BA-fractions in $XSBAFR()$ and $NXBAS$ is the number of fractions.

$(XSTEMP(I), I=1, I4)$ $\quad I4 = \sum_{N=1}^{NELT} NXSS(N) \; . \qquad (14)$

The temperatures (Kelvin) for which cross-section sets are given. Unlike in the two preceding arrays, the sequence of the isotopes is irrelevant. However, for each isotope the temperatures must increase.

$(XSBAFR(I), I=1, I5)$ $\quad I5 = \sum_{N=1}^{NBA} NXBAS(N) \; . \qquad (15)$

These are the BA-fractions per BA isotope. The isotope sequence is irrelevant, but the fractions are the same at all temperatures, and in increasing order.

Per group G:

$((ISTART(K,I), K=1,3), I=1, I4)$ Addressing information for $XSDATA()$:

$ISTART(1,I) = NFIRST$ is the start position of the cross-sections in $XSDATA()$ at the I^{th} temperature;

$ISTART(2,I) = NSELF$ is the self-scattering position in the cross-section set (zero if it belongs to an isotope with $N > NELR$);

$ISTART(3,I) = NSIZE$, is the length of this set:
$6 + GMAX - GMIN$ for isotope N with $N \le NELR$;
1 for isotope N with $NELR < N \le NELT - NBA$; and
$NXBAS$ for isotope with $N > NELT - NBA$.

$(XSDATA(\mathrm{I}), I=1, I6)$ $\quad I6 = \sum_{I=1}^{I4} NSIZE(I) \; , \qquad (16)$

from which follows

$$I7 = \max_G \{I6(G)\} \qquad (17)$$

This set contains, per temperature:

σ_a	$= XSDATA(NFIRST)$,
σ_f	$= XSDATA(NFIRST+1)$,
$\nu\sigma_f$	$= XSDATA(NFIRST+2)$,
σ_{tr}	$= XSDATA(NFIRST+3)$,
σ_s	$= XSDATA(NFIRST+4)$,

$$\begin{aligned}
\sigma(GMIN\rightarrow G) &= XSDATA(NFIRST+5),\\
&\vdots\\
\sigma(G\rightarrow G) &= XSDATA(NFIRST+NSELF-1),\\
&\vdots\\
\sigma(GMAX\rightarrow G) &= XSDATA(NFIRST+NSIZE-1).
\end{aligned}$$

For isotopes with $N > NELR$, there is only σ_a, except for BAs which have:

$$\begin{aligned}
\hat{\sigma}_a(fr=0) &= XSDATA(NFIRST),\\
&\vdots\\
\hat{\sigma}_a(fr=1) &= XSDATA(NFIRST+NSIZE-1).
\end{aligned}$$

The sequence in which these cross-section sets occur is given by $ISTART()$, so it does not have to be the same as that of the isotopes in $EID()$.

IX.8 BURNUP DATA

The start address of the burnup data is *INDBUR*. The burnup data are:

$(IDBUR(NB), NB=1, NBUR)$	The sequence numbers $N \leq NELT\text{-}NBA$ in $EID()$ of the burnup isotopes NB. According to section IX.3, $IDCODE(IDBUR(NB)) > 0$. Not all the isotopes with $IDCODE() > 0$ need to be present in $IDBUR()$, but then they cannot participate in the burnup.
$(IDFIS(NF), NF=1, NFIS)$	The sequence numbers $N \leq NELR$ in $EID()$ of the fissionable isotope NF. According to section IX.3, $IDCODE(IDFIS(NF)) > 1$. The same remark as above applies here for isotopes with $IDCODE() > 1$. However, $IDFIS()$ must be a subset of $IDBUR()$.
$(XEN(NF), NF=1, NFIS)$	Energy released per fission (Watt·second) of isotope NF in $IDFIS()$. These values are about 3×10^{-11}; they include prompt and delayed β and γ radiation, and they also account for capture γ's. They have been averaged over a typical PWR spectrum. Values at zero energy are given at *INDKAP*; see below.
$((BETA(I,NF), I=1,6), NF=1, NFIS)$	The delayed-neutron yields, arranged in six groups in order of decreasing precursor half-life.
$((DECAY(NB), NB=1, NBUR)$	Decay constants (sec^{-1}) of the $NBUR$ isotopes of $IDBUR()$. Typical values are about 10^{-5}–10^{-6} or zero.

$((YIELD(NF,NB),NF=1,NFISS),NB=1,NBUR)$	Fission product yields of isotopes *NB* from fission of isotopes *NF*. Of course, if *NB* is a heavy metal, it has zero yields.

Since HELIOS-1.7, energy release per fission as a function of incident neutron energy can be calculated from the values at zero incident neutron energy. Therefore, since library version 1.7, *XEN0*() values are stored at the start address *INDKAP*:

$(XEN0(NF),NF=1,NFIS)$	Energy released per fission (Watt–second) of isotope *NF* in *IDFIS*() at zero incident neutron energy. The values are about 3×10^{-11}; they include prompt and delayed β and γ radiation, but do not account for capture γ's (which is done in HELIOS).

IX.9 P_n SCATTERING DATA (n=1,2,3)

The P_1, P_2, and P_3 data have the same structure but start at different addresses. The P_1 data start at *INDP1M* (or –*INDP1M* if gamma data are present in the library), the P_2 data start at *INDP2M,* and the P_3 data start at *INDP3M.*

Each isotope with P_1 data also has P_2 and P_3 data. These data are tabulated at the same set of temperatures ($NP3=NP2=NP1$). Therefore, the arrays *IDNP1*(), *NP1T*(), *P1TEMP*(), which are the same for all P_n data ($n=1,2,3$), are tabulated only once. The description that follows is for P_1 data but applies also to P_2 and P_3 data.

The P_n data are not given together with the other cross-sections. That is because, until version 1.9, only a few isotopes with $N\le NELR$ had P_n data in the library and, if they did, they were usually given at fewer temperatures than the other cross-section data. Appendix B shows which isotopes have P_1, P_2, and P_3 data. They are present in the library in a similar manner as the other cross-section data, so for details one is referred to section XIV.7.

$(IDNP1(NP),NP=1,NP1)$	Sequence number $N\le NELR$ in *EID*() of the NP^{th} isotope with P_1 scattering data.
$(NP1T(NP),NP=1,NP1)$	$NP1T(NP)=100*NP1ADR+NP1S\ . \qquad (18)$ This is for the NP^{th} isotope in *IDNP1*() the start position *NP1ADR* in *P1TEMP*() and in *IP1STA*(). *NP1S* is the number of temperatures.
$(P1TEMP(I),I=1,I8)$	$I8=\sum_{NP=1}^{NP1}NP1S(NP)\ . \qquad (19)$ The temperatures (degrees Kelvin) at which the P_1 (P_2, P_3) data are given.

And now per group *G* and per scattering order *n*:

$((IPnSTA(K,I),K=1,3),I=1,I8)$	Addressing information for $P1DATA()$:
	$IPnSTA(1,I)=NPnST$ is the start position of the $\mathrm{P_n}$ data set in PnDATA() for the I^{th} temperature.
	$IPnSTA(2,I)=NPnSL$ is the self-scattering position in this set.
	$IPnSTA(3,I)=NPnSZ$ is the number of data in this set, $2+GMAX-GMIN$.

$(PnDATA(I),I=1,I9)$

$$I9=\sum_{I=1}^{I8}(2+GMAX(I)-GMIN(I)) \ , \tag{20}$$

from which follows

$$I10=\max_{G(P_1,P_2,P_3)}\{I9(G)\} \tag{21}$$

This set contains for each temperature the following data:

$$\sigma_n=\sum_{G'=1}^{NOG}\sigma^n(G\to G')=PnDATA(NPnST), \tag{22}$$

$$\begin{aligned}
\sigma^n(GMIN\to G) &= PnDATA(NPnST+1),\\
\vdots \qquad & \qquad \vdots\\
\sigma^n(G\to G) &= PnDATA(NPnST+NPnSL-1),\\
\vdots \qquad & \qquad \vdots\\
\sigma^n(GMAX\to G) &= PnDATA(NPnST+NPnSZ-1).
\end{aligned}$$

IX.10 GAMMA DATA

If there are gamma data, this is indicated by a negative value of *INDP1M* (with the right magnitude) among the dimensioning data of section IX.2. The start address of the gamma data, *INDGAM*, is stored in the last position of the library, which is *I0*-1 since addressing starts at zero. *I0=I0+IG0* is now the size of the neutron and gamma parts combined.

The gamma data consist of a general part, followed, per group, by collections of pure gamma cross-sections and gamma-production spectra due to neutron interactions. The general gamma data are:

$(GAMLIB(I),I=1,12)$	This is the 12A4 identification of the gamma data.
$(DATE(I),I=1,3)$	The date, in 3A2 format (yy mm dd), of the last time isotopes were added to or deleted from the gamma library.

IG0	The total number of data in the gamma library, i.e., *I0–INDGAM*.
IG1	Size of $TEMPNG()$ and $NGSTAR()$; see below.
IG2	Maximum size of $XSGDAT()$ in any gamma group; see below.
IG3	Maximum size of $XNGDAT()$ in any gamma group; see below.
IG4	Not used.
NOGG	Number of gamma groups.
NGISO	Number of isotopes with gamma data.
GPCUT	Last gamma group with non-zero pair-production cross-sections, i.e., the group containing 1.022 MeV.
GPAIR	The gamma group where the pair appears, i.e., the group containing 0.511 MeV.
$(EG(G), G = 1, NOGG+1)$	The limits of the gamma groups in eV, in order of descending energy.
$(ISOG(N), N = 1, NGISO)$	Identifiers of the gamma isotopes.
$(TEXT(I,N), I = 1,12), N = 1, NGISO)$	Per isotope, a short 12A4 text that describes the origin of its data.
$(ZG(N), N = 1, NGISO)$	The atomic numbers (real) of the gamma isotopes.
$(NGADR(N), N = 1, NGISO)$	$NGADR(N) = 100 * NADR + NSIZ\ . \quad (23)$ For the N^{th} isotope of $ISOG()$, the start position $NADR$ of its temperatures in $TEMPNG()$ and of its group-dependent addressing information (one number for each temperature tabulation) in $NGSTAR()$. $NSIZ$ is the number of temperatures.
$(TEMPNG(I), I = 1, IG1)$	$IG1 = \sum_{N=1}^{NGISO} NSIZ(N)\ . \quad (24)$ This contains the temperatures (Kelvin) for which gamma-production spectra are given. Unlike in the three preceding arrays, the sequence of the isotopes here is irrelevant. Per isotope, however, the temperatures must be in increasing sequence.

The remaining gamma data are given per gamma group *G*:

GDATA	Number of pure gamma cross-section data in group *G*, i.e., the size of *GSTART*() and *XSGDAT*() combined.
NGDATA	Number of gamma-production data in group *G*, i.e., the size of *NGSTAR*() and *XNGDAT*() combined.

$((GSTART(K,N),K{=}1,2),N{=}1,NGISO)$ Addressing information for *XSGDAT*():

$GSTART(1,N){=}N1ST$ is the start address for the N^{th} isotope;

$GSTART(2,N){=}NGMIN$ is, for isotope *N*, the lowest gamma group (highest energy) with non-zero in-scattering into group *G*.

$(XSGDAT(I),I{=}1,IG6)$

$$IG6 = \sum_{N=1}^{NGISO}(4+G-NGMIN(N)) \qquad (25)$$

from which follows that

$$IG2 = \max_{G}\{IG6(G)\}. \qquad (26)$$

This set contains the pure gamma cross-sections per isotope:

$$\begin{aligned}
\sigma_a(G) &= \sigma_{PE}(G)-\sigma_{PP}(G) &&= XSGDAT(N1ST),\\
\sigma_{tr}(G) &= \sigma_{PE}+\sigma_{PP}+\sigma_C &&= XSGDAT(N1ST+1),\\
\sigma_{PP}(G) & &&= XSGDAT(N1ST+2),\\
\sigma(NGMIN\rightarrow G) & &&= XSGDAT(N1ST+3),\\
\vdots & && \vdots\\
\sigma(G\rightarrow G) & &&= XSGDAT(N1ST+3+G-NGMIN),
\end{aligned}$$

where the subscripts *PE* and *PP* denote the photo-electric effect and pair production, *C* denotes the transport corrected Compton cross-section, and the scattering kernel includes pair production:

$$\sigma(G1\rightarrow G) = \sigma_C(G1\rightarrow G) + 2\sigma_{PP}(G1)\delta_{G,GPAIR}\,. \qquad (27)$$

$(NGSTAR(K,I),K=1,3),I=1,IG1)$ Addressing information for *XNGDAT*(); *I* is found from *NGADR*():

$NGSTAR(1,I)=N1ST$ is the start address of the gamma production vector in *XNGDAT*() for the I^{th} temperature;

$NGSTAR(2,I)=NGMIN$ is the first neutron group that contributes to gamma production;

$NGSTAR(3,I) = NGMAX$ is the last neutron group that contributes to gamma production.

$(XNGDAT(I), I = 1, IG7)$

$$IG7 = \sum_{I=1}^{IG1} (NGMAX(I) - NGMIN(I) + 1) \qquad (28)$$

from which follows that

$$IG3 = \max_{G}\{IG7(G)\}\,. \qquad (29)$$

This array contains, per temperature, the gamma-production cross-sections. For resonance isotopes in the resonance groups and for BAs in all the groups, these data are per neutron absorption. In all other cases, these are the elements of the (n,γ) production matrix. Thus, with ng denoting neutron groups:

$$p_{ng\to g} = \begin{cases} \dfrac{\sigma_{ng\to g}}{\sigma_{a,ng}} & \text{BAs and reso isotopes,} \\ \sigma_{ng\to g} & \text{all other situations.} \end{cases} \qquad (30)$$

$$\begin{aligned} p(NGMIN \to G) &= XNGDAT(N1ST) \\ &\vdots \\ p(NGMAX \to G) &= XNGDAT(N1ST + NGMAX - NGMIN + 1) \end{aligned}$$

REFERENCES

DECHER U (1994) *Users Manual for RABBLE, DIT Cross Section Library Preparation*, Rev. 1, Nuclear Data Project Report NDP-032, ABB Combustion Engineering Nuclear Fuel.

FERRI A A (1994) *Generation and Contents of the ENDF/B-VI Based HELIOS Library hy941a*, Scandpower Technical Note TN2/44.70.13.

FERRI A A (1996) *HEBE-1.4 and the new HELIOS Library hy961a*, Scandpower Technical Note TN18/41.16.15.

FERRI A A (1998) *HELIOS Library HY981A: General Description*, Studsvik Scandpower Technical Note TN6/44.70.13.

FREDIN B and STAMM'LER R J J (1977) *FOBUS - User's Manual and a Short Program Description*, ASEA ATOM Report PM RCB 77-166.

LIPIEC W (1994) *Procedure and Sample Documentation for Running NJOY: Gamma Production Cross Section Generation for Nuclides Without Evaluated Gamma Production Data,* Nuclear Data Project Report NDP-166, ABB Combustion Engineering Nuclear Fuel.

MACFARLANE R E *et al.* (1991) *NJOY91.13, A Code System for Producing Pointwise and Multigroup Neutron and Photon Cross Sections from ENDF/B Evaluated Nuclear Data,* Radiation Shielding Information Center Peripheral Shielding Routine Collection, PSR-171, Oak Ridge National Laboratory.

MUGHABGHAB S F and GARBER D I (1984) *Neutron Cross-Sections*, Vols 1 and 2, Academic Press, London, New York.

PERKINS S T, HAIGHT R C and HOWERTON R J (1975) *A Technique for Calculation of Continuum Photon and Conversion-Electron Production Cross Sections and Spectra from Neutron-Induced Reactions,* Nucl. Sci. Eng. **57**, 1-11.

ROSE P F and DUNFORD C L (1990) *ENDF-102 Data Formats and Procedures for the Evaluated Nuclear Data File ENDF-6,* BNL-NCS-4495, Brookhaven National Laboratory, Upton, New York.

SINGH U N and JONSSON A (1993) *Procedure for Multigroup Data Preparation for Non-ENDF/B-VI Materials Including the Processing of Tm-169,* Nuclear Data Project Report NDP-119, ABB Combustion Engineering Nuclear Fuel.

STAMM'LER R J J (1985) *The PHOENIX Neutron and Gamma Library: Structure and Service Program PHOEBE,* ASEA ATOM Report UR 85-153, Västerås, Sweden.

STAMM'LER R J J and ABBATE M J (1983) *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London.

APPENDIX A — GROUP STRUCTURES

Lower limits of the 190/112/47-group libraries.

Group	Lower limit (eV)		Group	Lower limit (eV)		Group	Lower limit (eV)	
000	2.0000E+07							
001	1.7000E+07		071	3.3546E+03		141	1.2351E+00	27
002	1.4919E+07		072	2.9604E+03		142	1.1664E+00	28
003	1.3380E+07		073	2.6126E+03		143	1.1254E+00	29
004	1.2000E+07		074	2.3056E+03		144	1.0987E+00	
005	1.0000E+07		075	2.0347E+03	10	145	1.0722E+00	30
006	8.8250E+06		076	1.7956E+03		146	1.0623E+00	
007	7.7880E+06		077	1.5846E+03		147	1.0525E+00	
008	7.4082E+06		078	1.3984E+03		148	1.0427E+00	
009	6.0653E+06	1	079	1.2341E+03		149	1.0137E+00	31
010	5.2205E+06		080	1.0891E+03		150	9.9200E-01	
011	4.7237E+06		081	9.6112E+02		151	9.7100E-01	32
012	4.4933E+06		082	8.4818E+02		152	9.5065E-01	
013	4.0657E+06		083	7.4852E+02		153	9.1000E-01	33
014	3.6788E+06	2	084	6.6057E+02		154	8.7642E-01	
015	3.1664E+06		085	5.8295E+02		155	8.3368E-01	
016	2.8650E+06		086	5.1445E+02		156	7.8208E-01	34
017	2.7253E+06		087	4.5400E+02		157	7.3000E-01	
018	2.4660E+06		088	4.0065E+02		158	6.7000E-01	
019	2.3650E+06		089	3.5358E+02		159	6.2506E-01	35
020	2.3457E+06		090	3.1203E+02		160	5.7000E-01	
021	2.2313E+06	3	091	2.7536E+02		161	5.3000E-01	
022	2.0189E+06		092	2.4301E+02		162	5.0323E-01	36
023	1.8268E+06		093	2.1445E+02		163	4.5000E-01	
024	1.7377E+06		094	1.8926E+02		164	4.1704E-01	
025	1.5724E+06		095	1.6702E+02		165	3.5767E-01	37
026	1.3534E+06	4	096	1.4739E+02		166	3.2063E-01	
027	1.1649E+06		097	1.3007E+02	11	167	3.0112E-01	
028	1.0540E+06		098	1.1479E+02		168	2.9074E-01	
029	1.0026E+06		099	1.0130E+02		169	2.7052E-01	38
030	8.2085E+05	5	100	8.9398E+01		170	2.5103E-01	
031	7.0651E+05		101	7.8893E+01	12	171	2.2769E-01	
032	6.3928E+05		102	6.9623E+01		172	1.8443E-01	39
033	6.0810E+05		103	6.1442E+01		173	1.5230E-01	
034	4.9787E+05	6	104	5.4222E+01		174	1.4572E-01	40
035	4.2852E+05		105	4.7851E+01	13	175	1.1157E-01	41
036	3.8774E+05		106	4.2229E+01		176	8.1968E-02	42
037	3.6883E+05		107	3.7267E+01		177	6.7000E-02	
038	3.0197E+05		108	3.2888E+01		178	5.6922E-02	43
039	2.5991E+05		109	2.9023E+01	14	179	5.0000E-02	
040	2.3518E+05		110	2.5613E+01		180	4.2755E-02	44
041	2.2371E+05		111	2.2603E+01		181	3.5500E-02	
042	1.8316E+05	7	112	1.9947E+01		182	3.0613E-02	45
043	1.4996E+05		113	1.7603E+01		183	2.5500E-02	
044	1.4264E+05		114	1.5536E+01		184	2.0492E-02	
045	1.2907E+05		115	1.3710E+01	15	185	1.2396E-02	46
046	1.1109E+05		116	1.2099E+01	16	186	6.3247E-03	
047	8.6517E+04		117	1.0677E+01		187	2.2769E-03	
048	6.7379E+04	8	118	9.4225E+00		188	7.6022E-04	
049	5.2474E+04		119	8.3153E+00	17	189	2.5399E-04	
050	4.0868E+04		120	7.3382E+00	18	190	1.0000E-04	47
051	3.6066E+04		121	6.8680E+00				
052	3.1828E+04		122	6.4760E+00	19			
053	2.8088E+04		123	5.7150E+00	20			
054	2.6058E+04		124	5.0435E+00	21			
055	2.4788E+04		125	4.4509E+00	22			
056	2.1875E+04		126	3.9279E+00	23			
057	1.9305E+04		127	3.4663E+00				
058	1.7036E+04		128	3.0590E+00				
059	1.5034E+04		129	2.6996E+00				
060	1.3268E+04		130	2.3824E+00	24			
061	1.1709E+04		131	2.1024E+00				
062	1.0333E+04		132	1.8554E+00	25			
063	9.1188E+03	9	133	1.7896E+00				
064	8.0473E+03		134	1.7261E+00				
065	7.1017E+03		135	1.6592E+00				
066	6.2673E+03		136	1.5949E+00				
067	5.5308E+03		137	1.5246E+00				
068	4.8810E+03		138	1.4574E+00	26			
069	4.3074E+03		139	1.3806E+00				
070	3.8013E+03		140	1.3079E+00				

The 112-group library has boundaries that are the following sub-set of the 190 groups:

1-110, 132, 190

Range	Groups
Fast	001-063
Reso	064-132
Up-scatt	127-190

48 gamma groups (lower limits of 18-group library indicated).

Group	Lower limit (eV)		Group	Lower limit (eV)		Group	Lower limit (eV)	
00	5.0000E+07							
01	3.0000E+07		21	2.3330E+06		41	1.0000E+05	
02	2.0000E+07		22	2.0000E+06	7	42	8.0000E+04	
03	1.7000E+07		23	1.8750E+06		43	6.0000E+04	16
04	1.4000E+07		24	1.6600E+06		44	4.5000E+04	
05	1.2000E+07		25	1.5000E+06	8	45	3.0000E+04	17
06	1.0000E+07		26	1.3300E+06		46	2.0000E+04	
07	9.0000E+06		27	1.2000E+06		47	1.0000E+04	
08	8.0000E+06	1	28	1.1250E+06		48	1.0000E+03	18
09	7.5000E+06		29	1.0000E+06	9			
10	7.0000E+06	2	30	9.0000E+05				
11	6.5000E+06		31	8.0000E+05	10			
12	6.0000E+06	3	32	7.0000E+05	11			
13	5.5000E+06		33	6.0000E+05	12			
14	5.0000E+06	4	34	5.2500E+05				
15	4.5000E+06		35	5.0000E+05				
16	4.0000E+06	5	36	4.5000E+05				
17	3.3000E+06		37	4.0000E+05	13			
18	3.0000E+06	6	38	3.0000E+05	14			
19	2.6660E+06		39	2.0000E+05				
20	2.5000E+06		40	1.5000E+05	15			

APPENDIX B — LIST OF ISOTOPES

List of isotopes (part 1)

Identifier	Description	Type	*k*	Identifier	Description	Type	*k*
1	1/*v* absorber	abso	-	28064	28-Ni-64	full	-
x 1001	1-H-1	full	-	29063	29-Cu-63	full	-
x 1002	1-H-2	full	2	29065	29-Cu-65	full	-
1003	1-H-3	abso	-	35581	35-Br-81	abso (FP)	-
x 1006	H in polyethylene	full	-	36582	36-Kr-82	abso (FP)	2
x 1040	H in ZrH	full	-	36583	36-Kr-83	abso (FP)	3
2003	2-He-3	abso	-	36584	36-Kr-84	abso (FP)	2
3006	3-Li-6	full	-	36585	36-Kr-85	abso (FP)	-
x 3007	3-Li-7	full	2	36586	36-Kr-86	abso (FP)	3
x 4009	4-Be-9	full	2	38589	38-Sr-89	abso (FP)	-
x 5000	natural boron	full	-	38590	38-Sr-90	abso (FP)	-
5010	5-B-10	full (B)	-	39589	39-Y-89	abso (FP)	2
5011	5-B-11	full	2	39590	39-Y-90	abso (FP)	-
x 6000	natural C	full	-	39591	39-Y-91	abso (FP)	-
x 6001	graphite	full	-	x 40000	natural Zr	full	2
x 7014	7-N-14	full	2	x 40001	Zr in ZrH_2	full	-
x 8001	8-O16 in UO_2	full	-	x 40002	Zircaloy-2	full	-
x 8016	8-O-16	full	-	x 40010	natural Zr	reso	2
x 9019	9-F-19	full	2	40090	40-Zr-90	full	-
x 11023	11-Na-23	full	2	40591	40-Zr-91	abso (FP)	2
x 12000	natural Mg	full	-	40593	40-Zr-93	abso (FP)	-
x 13027	13-Al-27	full	2	40595	40-Zr-95	abso (FP)	-
x 14000	natural Si	full	-	40596	40-Zr-96	abso (FP)	2
x 15031	15-P-31	full	-	x 41093	41-Nb-93	full	3
x 16000	natural S	full	-	41595	41-Nb-95	abso (FP)	-
x 17000	natural Cl	full	-	x 42000	natural Mo	full	3
19000	natural K	full	-	42095	42-Mo-95	abso	-
x 20000	natural Ca	full	-	42595	42-Mo-95	reso (FP)	-
x 22000	natural Ti	full	-	42596	42-Mo-96	abso (FP)	-
x 23000	natural V	full	-	42597	42-Mo-97	abso (FP)	-
x 24000	natural Cr	full	2	42598	42-Mo-98	abso (FP)	-
24050	24-Cr50	full	2	42599	42-Mo-99	abso (FP)	-
24052	24-Cr52	full	2	42600	42-Mo-100	abso (FP)	-
24053	24-Cr53	full	2	43599	43-Tc-99	reso (FP)	2
24054	24-Cr54	full	2	44600	44-Ru-100	abso (FP)	-
x 25055	25-Mn-55	full	2	44601	44-Ru-101	full (FP)	-
x 26000	natural Fe	full	2	44602	44-Ru-102	abso (FP)	-
x 26001	stainless steel 304	full	-	44603	44-Ru-103	abso (FP)	-
26054	26-Fe-54	full	-	44604	44-Ru-104	abso (FP)	-
26056	26-Fe-56	full	-	44605	44-Ru-105	abso (FP)	-
26057	26-Fe-57	full	-	44606	44-Ru-106	abso (FP)	-
26058	26-Fe-58	full	-	45103	45-Rh-103	full	2
x 27059	27-Co-59	full	2	45603	45-Rh-103	full (FP)	2
x 28000	natural Ni	full	2	45605	45-Rh-105	full (FP)	-
x 28001	Inconel 625	full	-	46604	46-Pd-104	abso (FP)	-
28058	28-Ni-58	full	-	46605	46-Pd-105	full (FP)	-
28059	28-Ni-59	full	-	46606	46-Pd-106	abso (FP)	-
28060	28-Ni-60	full	-	46607	46-Pd-107	full (FP)	-
28061	28-Ni-61	full	-	46608	46-Pd-108	reso (FP)	-
28062	28-Ni-62	full	-	x 47107	47-Ag-107	reso (B)	3

(continues …)

(… continued)

List of isotopes (part 2)

Identifier	Description	XS type	*k*	Identifier	Description	XS type	*k*
x 47109	47-Ag-109	reso (B)	3	55634	55-Cs-134	abso (FP)	-
47609	47-Ag-109	full (FP)	3	55635	55-Cs-135	abso (FP)	-
47611	47-Ag-111	abso (FP)	-	55636	55-Cs-136	abso (FP)	-
47710	47-Ag-110m	abso (FP)	-	55637	55-Cs-137	abso (FP)	-
x 48000	natural Cd	full	-	56634	56-Ba-134	abso (FP)	-
x 48110	48-Cd-110	full (B)	-	56637	56-Ba-137	abso (FP)	-
x 48111	48-Cd-111	full (B)	-	56640	56-Ba-140	abso (FP)	-
x 48112	48-Cd-112	full (B)	-	57639	57-La-139	abso (FP)	-
x 48113	48-Cd-113	full (B)	2	57640	57-La-140	abso (FP)	-
x 48114	48-Cd-114	full (B)	-	58640	58-Ce-140	abso (FP)	-
48610	48-Cd-110	abso (FP)	-	58641	58-Ce-141	abso (FP)	-
48611	48-Cd-111	abso (FP)		58642	58-Ce-142	abso (FP)	-
48613	48-Cd-113	abso (FP)	2	58643	58-Ce-143	abso (FP)	-
x 49113	49-In-113	reso (B)	-	58644	58-Ce-144	abso (FP)	-
49115	49-In-115	reso (B)	-	59641	59-Pr-141	abso (FP)	3
49615	49-In-115	full (FP)	-	59643	59-Pr-143	abso (FP)	-
x 50000	natural Sn	full	-	60642	60-Nd-142	abso (FP)	-
50112	50-Sn-112	full	-	60643	60-Nd-143	full (FP)	3
50114	50-Sn-114	full	-	60644	60-Nd-144	abso (FP)	-
50115	50-Sn-115	full	-	60645	60-Nd-145	full (FP)	3
50116	50-Sn-116	full	-	60646	60-Nd-146	abso (FP)	3
50117	50-Sn-117	full	-	60647	60-Nd-147	abso (FP)	-
50118	50-Sn-118	full	-	60648	60-Nd-148	abso (FP)	3
50119	50-Sn-119	full	-	60650	60-Nd-150	abso (FP)	3
50120	50-Sn-120	full	-	61647	61-Pm-147	full (FP)	3
50122	50-Sn-122	full	-	61648	61-Pm-148	abso (FP)	-
50124	50-Sn-124	full	-	61649	61-Pm-149	abso (FP)	-
50125	50-Sn-125	abso	-	61651	61-Pm-151	abso (FP)	-
51000	natural Sb	full	-	61748	61-Pm-148m	full (FP)	-
51121	51-Sb-121	full	-	x 62152	62-Sm-152	reso (B)	3
51123	51-Sb-123	full	-	x 62153	62-Sm-153	full (B)	-
51500	natural Sb	abso (FP)	-	62647	62-Sm-147	full (FP)	3
51625	51-Sb-125	abso (FP)	-	62648	62-Sm-148	abso (FP)	-
51627	51-Sb-127	abso (FP)	-	62649	62-Sm-149	full (FP)	3
52632	52-Te-132	abso (FP)	-	62650	62-Sm-150	full (FP)	2
52727	52-Te-127m	abso (FP)	-	62651	62-Sm-151	full (FP)	3
52729	52-Te-129m	abso (FP)	-	62652	62-Sm-152	full (FP)	3
53627	53-I-127	abso (FP)	3	62653	62-Sm-153	abso (FP)	-
53629	53-I-129	abso (FP)	-	62654	62-Sm-154	abso (FP)	-
53631	53-I-131	abso (FP)	-	x 63151	63-Eu-151	reso (B)	3
53635	53-I-135	abso (FP)	-	x 63152	63-Eu-152	reso (B)	3
54628	54-Xe-128	abso (FP)	3	x 63153	63-Eu-153	reso (B)	3
54630	54-Xe-130	abso (FP)	3	x 63154	63-Eu-154	reso (B)	3
54631	54-Xe-131	reso (FP)	3	x 63155	63-Eu-155	reso (B)	3
54632	54-Xe-132	abso (FP)	3	x 63156	63-Eu-156	full (B)	-
54633	54-Xe-133	abso (FP)	-	x 63157	63-Eu-157	full (B)	-
54634	54-Xe-134	full (FP)	3	63651	63-Eu-151	abso (FP)	3
54635	54-Xe-135	abso (FP)	-	63653	63-Eu-153	full (FP)	3
54636	54-Xe-136	abso (FP)	3	63654	63-Eu-154	full (FP)	3
55633	55-Cs-133	reso (FP)	2	63655	63-Eu-155	full (FP)	3

(continues …)

(… continued)

List of isotopes (part 3)

Identifier	Description	XS type	*k*	Identifier	Description	XS type	*k*
63656	63-Eu-156	abso (FP)	-	x 73181	73-Ta-181	full (B)	3
63657	63-Eu-157	abso (FP)	-	x 73182	73-Ta-182	full (B)	3
64152	64-Gd-152	full (B)	2	x 74000	natural W	full	3
x 64154	64-Gd-154	full (B)	2	77191	77-Ir-191	abso	-
x 64155	64-Gd-155	reso (B)	2	77193	77-Ir-193	abso	-
x 64156	64-Gd-156	reso (B)	2	x 79197	79-Au-197	full	-
x 64157	64-Gd-157	reso (B)	2	x 82206	82-Pb-206	full	-
x 64158	64-Gd-158	reso (B)	2	x 82207	82-Pb-207	full	-
x 64160	64-Gd-160	full (B)	2	x 82208	82-Pb-208	full	-
64654	64-Gd-154	abso (FP)	2	x 83209	83-Bi-209	full	3
64655	64-Gd-155	full (FP)	2	90230	90-Th-230	full	3
64656	64-Gd-156	full (FP)	2	x 90232	90-Th-232	reso	3
64657	64-Gd-157	full (FP)	2	x 91231	91-Pa-231	full	3
64658	64-Gd-158	full (FP)	2	x 91233	91-Pa-233	full	3
64660	64-Gd-160	abso (FP)	2	92232	92-U-232	full	3
65159	65-Tb-159	abso (B)	-	x 92233	92-U-233	reso	3
65160	65-Tb-160	abso (B)	-	x 92234	92-U-234	full	3
65161	65-Tb-161	abso (B)	-	x 92235	92-U-235	reso	3
65659	65-Tb-159	abso (FP)	-	x 92236	92-U-236	reso	3
65660	65-Tb-160	abso (FP)	-	x 92237	92-U-237	full	3
65661	65-Tb-161	abso (FP)	-	x 92238	92-U-238	reso	3
66160	66-Dy-160	reso (B)	-	x 93237	93-Np-237	full	3
66161	66-Dy-161	reso (B)	-	93238	93-Np-238	full	-
66162	66-Dy-162	reso (B)	-	93239	93-Np-239	full	3
66163	66-Dy-163	reso (B)	-	94236	94-Pu-236	full	3
66164	66-Dy-164	reso (B)	3	94238	94-Pu-238	reso	3
66660	66-Dy-160	abso (FP)	-	x 94239	94-Pu-239	reso	3
66661	66-Dy-161	abso (FP)	-	x 94240	94-Pu-240	reso	3
66662	66-Dy-162	abso (FP)	-	x 94241	94-Pu-241	reso	3
66663	66-Dy-163	abso (FP)	-	x 94242	94-Pu-242	reso	3
66664	66-Dy-164	abso (FP)	3	x 95241	95-Am-241	reso	3
67165	67-Ho-165	abso (B)	3	95242	95-Am-242	full	-
67665	67-Ho-165	abso (FP)	3	95243	95-Am-243	full	3
68166	68-Er-166	reso (B)	-	95342	95-Am-242m	full	3
68167	68-Er-167	reso (B)	-	96242	96-Cm-242	full	3
68168	68-Er-168	full (B)	-	96243	96-Cm-243	full	3
68170	68-Er-170	full (B)	-	96244	96-Cm-244	full	3
69169	69-Tm-169	reso (B)	-	96245	96-Cm-245	full	3
69170	69-Tm-170	reso (B)	-	96246	96-Cm-246	full	3
69171	69-Tm-171	abso (B)	-	96247	96-Cm-247	full	3
71176	71-Lu-176	full	3	96248	96-Cm-248	full	3
72174	72-Hf-174	full (B)	2	97249	97-Bk-249	full	3
x 72176	72-Hf-176	reso (B)	2	98249	98-Cf-249	full	3
x 72177	72-Hf-177	reso (B)	2	98250	98-Cf-250	full	3
x 72178	72-Hf-178	reso (B)	2	98251	98-Cf-251	full	3
x 72179	72-Hf-179	reso (B)	2	98252	98-Cf-252	full	3
x 72180	72-Hf-180	reso (B)	2				

(ends)

Notes:

1. Types: *abso* isotopes have only σ_a data;
   full isotopes/identifiers have complete cross-section sets;
   reso isotopes are *full* and have also resonance data;
   FP isotopes are fission products;
   B isotopes are burnable, though no fission products;
   k = -/2/3 = the isotope has no/(n,2n)/(n,2n) and (n,3n) data.

2. 1/v absorber (identifier 1) — has $\sigma_{2200} = 1$ barn and has no γ data.

3. Natural boron (identifier 5000)

Natural boron	19.8 at%	B-10,
(identifier 5000)	80.2 at%	B-11.

4. Stainless steel 304 (identifier 26001)

Stainless steel 304	70.351 wt%	Fe,
(identifier 26001)	19.152 wt%	Cr,
	8.483 wt%	Ni,
	2.014 wt%	Mn.

5. Inconel 625 (identifier 28001)

Inconel 625	65.231 wt%	Ni,
(identifier 28001)	25.293 wt%	Cr,
	5.477 wt%	Fe,
	5.102 wt%	Mo,
	1.038 wt%	Co,
	0.559 wt%	Mn.

6. Zircaloy-2 (identifier 40002)

Zircaloy-2	98.230 wt%	Zr,
(identifier 40002)	1.500 wt%	Sn,
	0.120 wt%	Fe,
	0.100 wt%	Cr,
	0.050 wt%	Ni.

7. Identifiers with Pn data: are marked with an 'x' before the identifier.

8. Identifiers with the inflow transport correction:

1001	1002	4009	6000	6001
8001	8016	11023		

Studsvik Scandpower

USER MANUAL

ZENITH

01 April 2008

DOCUMENT CONTROL

This document will be updated periodically, whenever revisions are made to the Manual. This front page, together with a page specifying the pages to be deleted and/or added, will be provided with each new revision. The User should check that the Manual corresponds to the Program Version in use and that all revisions are included.

INPUT MANUAL		PROGRAM		REMARKS	Rev'd	Appr'd
Rev. No	Date	Version	Date		by	by
0	11Aug93	ZE931A zenith-1.0	11Oct93	Original release		
0	-	ZE941A zenith-1.1	08Feb95	Unchanged		
0	-	zenith-1.2	30Nov95	Unchanged		
0	-	zenith-1.3	30Jan96	Unchanged		
0	-	zenith-1.4	05Jul96	Unchanged		
1	24Sep98	zenith-1.5	01Oct98	Minor changes		
2	01Apr00	zenith-1.6	01Apr00	Re-edited, examples added		
3	31Oct01	zenith-1.7	31Oct01	No more leakage-averaging; minor changes.		
4	06Nov03	zenith-1.8	06Nov03	PRINT operator added; INDEX/SORT improved; other minor changes.		
5	01Dec05	zenith-1.9	29Jul05	Unchanged		
6	01Apr08	zenith-1.10	01Apr08	Introduction of P_2 and P_3 output; other minor changes and typos		

Revision 1: 24 September 1998

The whole manual is replaced.

Revision 2: 01 April 2000

The manual has been re-edited.
Minor changes made.
Three larger examples have been added to the FOR operator.

Revision 3: 31 October 2001

Leakage-averaging is no longer supported.
Minor changes made.

<u>Pages replaced:</u>

Cover page	
Document Control	pages 1-2
Chapter 1	pages 1-2
Chapter 2	pages 1-43
Chapter 5	pages 1-8

Revision 4: 06 November 2003

PRINT operator added.
INDEX/SORT improved by adding a dimensional sorting option.
Minor changes made.

<u>Pages replaced:</u>

Cover page	
Document Control	pages 1-2
Contents	page 1
Chapter 1	pages 1-2
Chapter 2	pages 1-46

Revision 5: 01 December 2005

Unchanged, except cover page and document control.

<u>Pages replaced:</u>

Cover page	
Document Control	pages 1-2

Revision 6: 01 April 2008

Introduction of data types *p2*, *p2ta*, *p3* and *p3ta*
Added a version check to assure version compatibility
Changed catalog STATUS to STATUX to provide graceful failure for earlier ZENITH versions with new database files
Minor changes made and typos corrected.

Pages replaced:

Cover page
Document Control ...... pages 1-3
Chapter 2 .... pages 1-46

CONTENTS

CHAPTER 1

SUMMARY DESCRIPTION

The underlying concept of ZENITH is a simple, yet powerful, programming language that allows the user to operate with the basic data generated by HELIOS. This approach provides maximum flexibility for calculating derived quantities, such as reaction rates or detector constants. It also permits to customize the processing of the HELIOS results according to specific needs.

The most important elements of the ZENITH language are the input operators. Each operator performs a well-defined task, like reading a certain type of data from a HERMES file or printing data columnwise in the output file. Some operators define 5-dimensional data structures (arrays), while others produce symbolic names for files, data sets, and output maps and lists.

The input file goes through a pre-processor module before the input operators are interpreted. The pre-processor is used to activate and deactivate parts of the input file according to the given instructions. It filters the input file leaving only the input lines required for the particular application.

The interface with the HERMES files containing the HELIOS cases is taken care of by a dedicated operator. Each reference to this operator retrieves one data type from file -about 60 different data types are defined- and loads its values into an array.

A number of operations involving arrays is supported, including arithmetics with unrestricted level of parentheses, built-in functions, and other operations like sorting and permuting dimensions.

The array values can be printed in the output file either columnwise in a selected dimension, or as X-Y maps. Several options are available for format and layout control, including a feature that allows zooming in a particular map area. Since ZENITH-1.8, a PRINT operator is available to write arrays in a formatted file for easy data transfer to other codes.

Arrays also can be written into the output HERMES file, which has the structure required by TABGEN for further processing. Two-way file conversions between the HERMES binary format and the TABGEN-ASCII format are supported.

The input file preparation can be simplified with the use of parameters and sets. Sets are like "subroutines" of input operators that are stored in a HERMES file by the set-management feature of AURORA. They may contain parameters to add flexibility to their use.

The command line to run Zenith has three parameters, the last two are optional:

```
zenith-version     input-file-name     [MAXA]     [MAXC]
```

The first parameter, the name of the input file, is mandatory. The other two, MAXA and MAXC, allow to define at run time the sizes (in words) of the integer/real working area and of the character working area. Their default values are 15,000,000 and 1,500,000, respectively.

CHAPTER 2

INPUT DESCRIPTION

General description of the input format

1) The entire input is format-free. The valid input data field extends from column 1 through column 132.

2) Blank spaces and tabs are ignored unless belonging to a character string enclosed in apostrophes ('). Input lines containing only blank spaces or tabs are ignored.

3) Character strings enclosed in apostrophes cannot span over more than one line.

4) On any input line, anything enclosed by exclamation marks (!) is a user comment field. Comment fields, if not closed, are always closed at the end of the line.

5) Input elements may extend over several lines with no need of special marks to indicate continuation lines. They are recognized using syntax rules.

6) User-defined names and the input operator names are case-sensitive. All other names, like keywords to select operator options, are case insensitive.

7) User-defined names are: left-hand side names of operators, file names, titles, parameter names and pre-processor variable names. The following rules apply:
   - They must start with a letter and be followed by letters or numbers. Otherwise they must be enclosed in apostrophes.
   - The maximum allowed name length is 80 characters.

8) The components of the input file are:
   - Pre-processor variables and statements, identified by a percent sign (%) as their first non-blank character.
   - The mandatory case delimiters BEGIN() and END(), that separate the individual cases of a series of cases.
   - The input operators.

ZENITH pre-processor

ZENITH has a pre-processor that is executed before any input operator is read. The functions of the pre-processor are to set the global options and to modify the input file as indicated by conditional if-then-else statements.

The input lines containing instructions for the pre-processor are identified by a per-cent (%) sign as their first non-blank character. They are called "% statements". The valid % statements are:

%options %if %else %endif

They will be described later in this manual.

The %if statement acts on pre-processor variables that were previously defined. These variables can only take single integer values through a definition of the form:

%*name* = *constant*

where *name* is the variable name and *constant* a valid integer number.

The use of pre-processor statements is optional.

General syntax of input operators

The general syntax of the input operators is:

name = OPER(***field1/.../fieldN***)

name The name of the entity the input operator defines. The possible entities are:

1. A name to access a HELIOS case in a HERMES file.
2. An array with 5 dimensions, but not all of them are always used.
3. A name to access a data set generated by the BONE operator.
4. A name of an output ZENITH case saved in a HERMES file.
5. List or map to print.

OPER The input operator name. The valid names are given below, in the order they are likely to appear:

BEGIN	IMPORT	ARRAY	BONE
SELECT	INDEX	FORMULA	LIST
MAP	PRINT	EXPORT	CONVERT
PARAMETER	SET	END	

With the exception of BEGIN, which must be fully spelled, only the first three letters of the operator names are mandatory and must be capitalized.

The BEGIN and END operators do not define a left- hand side name. They mark the beginning and end of a ZENITH case.

fieldj The j-th input field of the input operator. It may be a single entry or multiple entries separated by delimiters. The number of input fields *N* varies with the operator. Input fields that are optional are enclosed in square brackets ([]) throughout the input description.

The sequence of the operators in the input file is important. Any entity inside the current operator fields must have been defined by a preceding operator. If the entity was defined more than once, the last definition before the current operator applies.

The array dimensions are identified by letters, ordered from one to five: *R*, *I*, *E*, *G*, *O*. *R* stands for *region* (it can refer to *faces* also), *I* for *isotope*, *E* for *exposure* (burnup), *G* for *group* and *O* for *originating group* (in matrices).

%OPTIONS(), global options operator

Purpose

This optional operator is processed by ZENITH before any other, regardless its position in the input deck. It is used to define administrative parameter values and listing-file options.

Usage

%OPTIONS (***op=val*****[,*****op=val*****...]]**)

Where:

op	-	option or administrative parameter name.
val	-	option or administrative parameter value.

Rules

1) Listing-file options

Valid names are: LINPAG, LIST and OPENSTAT. LINPAG is used to control the paging on the output listing file. LIST, if set to YES, will print the resulting input file after each processing step and the name of the lists contained under the ZINPUT directory of each specified HERMES file. OPENSTAT is used to change the OPEN status of the output listing file. CCONTROL is used to enable or disable carriage control on the output listing file.

The following table summarizes the listing-file options:

Option (*op*)	Value (*val*)	Description
LINPAG	N > 10	Maximum number of lines per page (default=60)
LIST	NO YES	Verbose output listing (default=NO)
OPENSTAT	OLD NEW UNKNOWN	Status of listing file (default=UNKNOWN)
CCONTROL	ENABLED DISABLED	Carriage control on listing file (default=ENABLED)

val names can be abbreviated (a minimum of three letters is required).

2) Administrative parameters

They are used to dimension administrative arrays whose exact size is not known at the time they are dimensioned. The program has reasonable default values for them, but in certain occasions they could be exceeded. If this happens, ZENITH will stop with an error message indicating which parameter needs to be increased. Then the user can proceed directly to increase its size.

The following table lists the administrative parameters and their default values.

Parameter	Default value	Description
MXLNAM	30 000	Maximum number of names given in the input file
MXAUX	500 000	Maximum number of data in one operator
MXCON	40 000	Maximum number of real constants in the input file
MXKON	5 000	Maximum number of integer constants in the input file
MXTIT	400	Average size of the HELIOS title [bytes]
MXELM	400	Maximum number of elements in any single data field
MXLIST	50 000	Maximum size of a HERMES list in file conversions
MXNRP	20	Minimum reserved number of calculational points. Used only if larger than the number of "rpp" catalogs in STATUX
MXSET	800	Maximum number of lines in a set

Remarks

The administrative parameters should only be modified if an error message indicates that one of them was exceeded.

%IF(), %ELSE, %ENDIF conditional block statements

Purpose

The conditional block statements are used to modify the input file according to the value given to a pre-processor variable. They filter out input lines depending on the instructions.

Usage

%IF (***var.co.const***) THEN

[%ELSE]

%ENDIF

Where:

var	is a pre-processor variable name previously defined as %*var* = integer number.
const	is an integer number.
co	is the conditional clause, one of the following:

LT	- less than
EQ	- equal to
GT	- greater than
LE	- less than or equal to
GE	- greater than or equal to
NE	- not equal

Rules

1) If the logical condition *var*.*co*.*const* is true (false), the pre-processor will:
   a) include (remove) all the input lines between the statements %IF and %ELSE, and
   b) remove (include) all the input lines between the statements %ELSE and %ENDIF.

2) If the statement %ELSE is omitted, only the action in a) is performed (with %ELSE replaced by %ENDIF).

3) Nesting is allowed, provided each %IF statement is closed by the corresponding %ENDIF statement on each level of nesting.

Remarks

The variable occurring in the %IF statement must have been defined as indicated in section "ZENITH pre-processor".

Examples

1)
```
%if (ndet.le.1) then
        --- Block 1 ---
%else
        --- Block 2 ---
%endif
```

2)
```
%if (gam.eq.1) then
        --- Block ---
%endif
```

1) Include the input lines represented by Block 1 if the variable *ndet* is set to 1 or less. Otherwise include the input lines represented by Block 2.

2) Include the input lines represented by Block if the variable *gam* is set to 1.

BEGIN(), the begin-case operator

Purpose

Marks the beginning of a ZENITH case. The title of the case and, optionally, the region numbering directives are given here.

Usage

BEGIN (***ti*[/*bx*[;*by*]][/*d1*;*d2*]]**)

Where:

ti	is the case title. It will be printed on each page of the output listing.
bx by	are the tolerance bands in *x* and *y*. Their units depend on their sign: if +ve: absolute value (cm). if -ve: fraction of the system size.
d1 d2	are the numbering directives for regions and faces.

Rules

1) The BEGIN operator is mandatory and must be the first operator of a case.

2) Gravity centers whose *x* (*y*) coordinates are within the tolerance band *bx* (*by*) are considered to be aligned in the same column (row). If only *bx* is given, it is assumed that *by*=*bx*.

3) Tolerance bands can be given as absolute values in centimeters (positive entry), or relative to the size of the system in each direction (negative entry). If absolute tolerance bands are specified, ZENITH will compare the input value with the geometric tolerance of HELIOS (0.0001 cm) and use the larger of the two values.

   By default *bx* = *by* = -0.005

4) If the bands are larger than the system dimensions, the HELIOS internal region numbering is retained.

5) The numbering directives *d1 d2* are used to define the order in which ZENITH will sweep the gravity center coordinates to assign a number to each region or face. The valid entries for *d1* and *d2* are:

LR	(from left to right)
RL	(from right to left)
TB	(from top to tottom)
BT	(from bottom to top)

The default is *d1*=LR and *d2*=TB. This means that ZENITH starts numbering the uppermost row from left to right, then continuing with the next row below and repeating the procedure until all regions or faces are covered.

Any combination of *d1* and *d2* that covers both the vertical and horizontal axes is valid. There are eight valid combinations.

6) The BEGIN operator cannot be included in a set.

Remarks

After numbering with the specified bands, ZENITH will test the sensitivity of the numbering process by repeating it with two slightly modified band sizes (±1%). If the three results are not identical a warning message will be issued. In this case it is advised to adjust the band values until the sensitivity test is passed without warnings.

Examples

1) BEGIN ('BWR 8x8 3.15 6G2.0')

2) BEGIN ('BWR 8x8 3.15 6G2.0, RL-BT' // RL;BT)

1) The title of the case is defined. Default tolerance bands and region numbering are used.

2) The title of case is defined. Default tolerance bands are used. Regions are numbered from right to left and from bottom to top.

IMPORT(), the import-case operator

Purpose

The IMPORT operator is used to specify a HELIOS case to be processed by ZENITH.

Usage

name = IMPORT(**[*p*[;*p*...];]*na*/*fi***)

Where:

name	is the name of the import operator.
p	is the name of a catalog in the path to reach the case *na*.
na	is the HELIOS case name assigned by the AURORA CASE operator.
fi	is the name of the HERMES file where *na* is.

Rules

1) The case name *na* is the one defined on the left-hand side of the AURORA CASE operator.

2) The catalog path leading to *na* will normally consist of only one catalog, called "HELIOS", because HELIOS always places the case catalogs under it.

3) *fi* is the HERMES data base file defined in the AURORA CASE- operator.

Remarks

File names must comply with the general rules for names. Since they normally contain special characters, such as a period, they must be enclosed in apostrophes.

Examples

1) c1 = IMPORT (HELIOS; 'BWR-4x4' / 'TEST-4x4.HMS')

1) This example defines an import operator "c1" which points to case "BWR-4x4" in the HERMES file "TEST-4x4.HMS".

ARRAY(), the define-array operator

Purpose

The ARRAY operator is used to define a one-dimensional array of real numbers. The array contents is either defined in the ZENITH input file, or carried in the CASE operator of AURORA.

Usage

name = ARRAY(***di*:[*i**]*r*[;[*i**]*r*...]**) or

name = ARRAY(***ca*/*di*:*x***)

Where:

name	is the array name.
di	is the dimension to be loaded with data: *di* = *R*,*I*,*E*,*G*,*O*.
i	is an integer multiplier. The following real number is repeated *i* times.
r	is a real number.
ca	is the name of an IMP operator pointing to the HELIOS case.
x	is the name of a set of real numbers in the text strings of the CASE operator in *ca*.

Rules

1) The array is loaded in the sequence given by the *i***r* entries.

2) In order to be recognized by ZENITH, arrays defined in the text strings of the AURORA CASE operator must comply with the following syntax:

   << *x* = (**[*i**]*r*[;[*i**]*r*...]**) >>

   The "<<" and ">>" are the string delimiters for ZENITH. Several set definitions may be given in the same CASE operator; the first one that matches the name *x* is used.

Remarks

One-dimensional arrays in ZENITH are treated as 5-dimensional arrays with only one dimension defined. The selection of a particular dimension depends on the application.

Examples

```
1)    TIP-resp   = ARRAY(G:1.915;1.862;2.028;2.870;5.284)

2)    voids      = ARRAY(c1/R:vi)
```

1) The array "TIP-resp" contains the 5-group response function of a gamma TIP detector in its G dimension.

2) The array "voids" is defined as one-dimensional in R, with its values taken from the set "vi" of case "c1". The set "vi" is defined as follows in AURORA to indicate, for example, 40% instantaneous void and 70% void history:

```
[AURORA]            c1  = CASE ( xsfil / data.hms /
                    'Voids for ZENITH << vi=(0.40;0.70) >>'
```

BONE(), the B_1 operator

Purpose

The BONE operator acts on an existing neutron macro and defines a new one. Data in the new macro are generated by B_1 calculations followed by a possible group condensation using the B_1 fluxes and currents.

Usage

name = BONE(**[*p*]/[*ca*:]*ma*[/*g*[;*g*...]]**)

Where:

name	is the name of the new macro defined by the operator.
p	controls the output of the B_1 module: 1 - no output (default). 2 - spectrum rebalancing factors. 3 - 2 plus B_1 flux and current spectra.
ca	is the name of an IMPORT operator pointing to the HELIOS case.
ma	is the name of a neutron MACRO catalog in *ca*, or in the internal HERMES file.
g	is the group number of the lower energy limit of a group in the new structure.

Rules

1) If the case name *ca* is omitted, the macro is assumed to be in the internal HERMES file. This means that it was generated by an earlier BONE operator.

2) The input macro *ma* must be a neutron macro containing at least the data types: *ab*, *nf*, *dn*, *ch*, *p0* and *p1*. They must be defined in a single-region area.

3) The group limits *g*, if not given, are assumed the same as those in the input macro. In this case no condensation is performed.

4) Group limits must be specified in increasing order. The last group limit must be equal to the highest group number (lowest-energy group) of the input macro.

Remarks

The BONE operator defines new data types for the subsystem represented in the input macro. The new data types are the effective delayed neutron yields in six groups, *be*. They are generated only when the input macro contains the *dn* data type. The *dn* data type are β's averaged with the fission-neutron production rate per group of the macro output, Eq. (VIII.16) of *HELIOS Methods*. The *be* data type are the effective delayed-neutron yields according to Eq. (VIII.19) of *HELIOS Methods*.

The BONE operator also redefines existing data types that were generated by HELIOS for the entire system. When redefined, however, these data types are for the subsystem represented in the input macro. They are:

dr	- ratio of the B_1 diffusion coefficient to the diffusion coefficient $1/3\Sigma_{tr}$.
kinf	- infinite multiplication factor of the subsystem.
kinfb	- infinite multiplication factor of the subsystem in the critical spectrum.
bsq	- material buckling (cm^{-2}) of the subsystem.
msq	- migration area (cm^2) of the subsystem.

The following three data types are not calculated but set to zero by the BONE operator: *eigv*, *eigvb2* and *b2*. They are related to transport calculations that are not performed by the BONE operator.

The output macro is saved in the internal HERMES file of ZENITH, with the name given in the left-hand side of the B_1 operator. The internal HERMES file is defined only for the actual ZENITH case, and it is deleted at case completion.

Examples

1) b1mac-5g = BONE(/c1:'mac-10g'/3;4;5;7;10)

2) b1mac-2g = BONE(-1/'b1mac-5g'/3;5)

1) Define the macro "b1mac-5g" in five energy groups by condensing the 10-group input macro.

2) Take the output macro from the first example, condense it further to two groups and print the spectrum re-balancing factors.

SELECT(), the select-data operator

Purpose

The SELECT operator defines a ZENITH array with data from a HERMES file.

Usage

name = SELECT(***dt/op/*[*ca*];[*na*][/[[*b*[;*b*…]][/*is*[;*is*…]]]]**)

Where:

name	is the array name defined by SELECT.
dt	is the data type to retrieve. See Table I for valid names.
op	identifies the type of the edit operator referenced by *na*. Valid types are: STATUX, MACRO, MICRO, CUR, GROUP, AREA, FACE
ca	is the name of an IMPORT operator pointing to the HELIOS case.
na	is the name of the AURORA edit operator in *ca*.
b	is a selected reactivity point, given as:

***pa*[:(*st*)*c*[-*c*][,[(*st*)]*c*[-*c*]…]] *c*=*e*[@*n*]**

where:

pa	is the name of a path or tree.
(*st*)	is the name of a state in pa, in parentheses.
e	is a burnup value in MWd/t (metric tons).
n	is the sequence number (1, 2 …) in (*st*) of the calculational point.

is	is the library identifier of an isotope.

Rules

1) *ca* is optional only for macro-type operators. If not given then the catalog *na* must have been defined by a BONE operator in the same case.

2) *na* is the name (left-hand side) of an AURORA edit operator. The AURORA edit operators are:

GROUP, AREA, FACE, MACRO, MICRO and CUR.

na is ignored for the status-type operator because there is no associated edit operator in AURORA (HELIOS always saves this data set).

3) The edit operator types *op* may be abbreviated. The first three letters are required, although more can be given.

4) The list of reactivity points ***b*[;*b*...]** is optional. If not given, all the available reactivity points are selected. If only the path name *pa* is specified, all the points belonging to this path will be selected. If *b* is present and the data type does not load the E-dimension (see Table I), it will be ignored.

5) The isotope list *is*[;*is*...] is an optional field that is used only for micro-type operators (it is ignored for the other operator types). If given, data for only the specified isotopes will be retrieved; if not given, all the isotopes present in the micro-type operator are selected.

Remarks

Table I in the following pages contains a list of the valid data type names *dt*, the edit operator types where they are defined, and the array dimensions they load.

Examples

1)	abso	= SEL(ab/mac/c1;'assy-2g'/depl:(0vh)0@2-50000)
2)	'DB1/D'	= SEL(dr/mac/;'assy-2gB1')
3)	jplus	= SEL(jp/cur/c1;'cassy-2g'/depl:(0vh)0@2-50000)
4)	burnup	= SEL(uburn/status/c1;)
5)	NU235	= SEL(nd/mic/c1;'fuel-2g'//92235)

1) The array "abso" is loaded with the values of the absorption cross section that were saved by HELIOS in the macro-type data set "assy-2g". "c1", defined by an IMPORT operator, points to the HELIOS case and the HERMES file containing the case.
According to Table I, all the dimensions except *I* and *O* are defined for "abso" (*I* is not loaded because "assy-2g" is a macro-type data set, and *O* is not defined for data type ab). In dimension *E*, the calculational points of path "depl" having the state "0vh" and burnup levels between 0 and 50000 MWd/t are included, except the first calculational point at 0 MWd/t (i.e. the xenon-free point).

2) Here the array "DB1/D" contains the ratio of the B_1 diffusion coefficient to the diffusion coefficient, as calculated by the BONE operator that defined the macro-type data set "assy-2gB1" (no reference to an IMPORT operator here). According to Table I, only dimensions *E* and *G* are loaded. Since no reactivity point selection is given, all points are selected (Rule 4).

3) The array "jplus" is loaded with the outcoming partial currents saved in the cur-type data set "cassy-2g". The same reactivity points as in Example 1) are selected. Dimensions loaded are *R*, *E* and *G* (see Table I).

4) The "burnup" array defined here contains the average burnup values for all the reactivity points of the HELIOS case pointed by "c1". According to Table I, *E* is the only dimension loaded.

5) The microscopic number density for U-235 (identifier 92235) is loaded from the micro-type data set "fuel-2g" into the array "NU235". All available reactivity points are selected. The array dimensions that are loaded are *R*, *I* and *E*.

Table I: Data types for use in the SEL operator

Data type name	Time average name	Edit operator types for which data type is defined	Description of data type [units]	Dimensions loaded
ab	abta	MAC, MIC	Absorption XS [cm^{-1} or barns]	R [I] E G .
aw	---	MIC	Atomic weight [amu]	. I . . .
b2	---	all	Buckling given in the AURORA input file [cm^{-2}]	. . E . .
be	beta	MAC defined by BONE()	Effective delayed neutron fractions (6 delayed neutron groups) [-]	1 . E 6 .
bsq	---	all	Material buckling of the homogenized system [cm^{-2}]	. . E . .
bu	buta	MAC	Region burnup [MWd/t]	R . E 1 .
ch	chta	MAC	Fission spectrum [-]	R . E G .
dn	dnta	MAC	$\nu\Sigma_f\phi$-averaged delayed neutron yields (6 delayed neutron groups) [-]	R . E 6 O
dr	---	GROUP, MAC, MIC, CUR	$3{*}\Sigma_{tr}$ times the B1 diffusion coefficient [-]	. . E G .
ed	edta	MAC, MIC	Kinetic energy deposited by scattering (and absorption for γ's) [Watts/cm^3]	R [I] E 1 .
eigv	---	all	Eigenvalue of the transport calculation [-]	. . E . .
eigvb2	---	all	Eigenvalue of the system at input buckling *b*2 [-]	. . E . .
fi	fita	MAC, MIC	Fission XS [cm^{-1} or barns]	R [I] E G .
fx	fxta	MAC, MIC	Neutron or gamma flux [cm$^{-2}{*}$sec^{-1}]	R . E G .
heg	---	GROUP, MAC, MIC, CUR	Higher energy group limits [eV]	. . . G .
heng	---	MAC, MIC	Higher energy limits of neutron groups in (n, γ) matrix [eV]	. . . G .
id	---	MIC	Isotope library identifier [-]	. I . . .
jm	jmta	CUR	Incoming partial current [cm$^{-2}{*}$sec^{-1}]	R . E G .
jp	jpta	CUR	Outgoing partial current [cm$^{-2}{*}$sec^{-1}]	R . E G .

Data type name	Time average name	Edit operator types for which data type is defined	Description of data type [units]	Dimensions loaded
kf	kfta	MAC, MIC	Fission-energy production $\kappa\Sigma_f$ or $\kappa\sigma_f$ [Joule/cm or Joule$*$barn]	R [I] E G .
kinf	---	all	Infinite multiplication factor of the system [-]	. . E . .
kinfb	---	all	Infinite multiplication factor of the system in the critical spec-trum [-]	. . E . .
kod	---	MIC	Isotope code 1...6 [-]	. I . . .
leg	---	GROUP, MAC, MIC, CUR	Lower energy group limits [eV]	. . . G .
leng	---	MAC, MIC	Lower energy limits of neutron groups in (n, γ) matrix [eV]	. . . G .
miso	---	MIC	Number of isotopes in MICRO [-]	1 . . . .
msq	---	all	Migration area of the homogenized system [cm^2]	. . E . .
nd	ndta	MIC	Number densities [atoms/barn-cm]	R I E . .
ndt	---	MAC, MIC	Number of data types stored [-]	1 . . . .
nf	nfta	MAC, MIC	Fission-neutron production $\nu\Sigma_f$ or $\nu\sigma_f$ [cm^{-1} or barns]	R [I] E G .
nfa	---	CUR	Number of faces in CUR [-]	1 . . . .
ng	ngta	MAC, MIC	(n, γ) production matrix [cm^{-1} or barns]	R [I] E G O
ngl	---	GROUP, MAC, MIC, CUR	Number of energy groups [-]	1 . . . .
nng	---	MAC, MIC	Number of neutron groups of the (n, γ) matrix [-]	1 . . . .
nra	---	MAC, MIC	Number of region areas in MACRO/MICRO [-]	1 . . . .
p0	p0ta	MAC, MIC	P0-scattering matrix [cm^{-1} or barns]	R [I] E G O
p1	p1ta	MAC, MIC	P1-scattering matrix (flux-weighted) [cm^{-1} or barns]	R [I] E G O
p2	p2ta	MAC, MIC	P2-scattering matrix [cm^{-1} or barns]	R [I] E G O
p3	p3ta	MAC, MIC	P3-scattering matrix [cm^{-1} or barns]	R [I] E G O
pburn	---	all	Actual path burnup [MWd/t]	. . E . .

Data type name	Time average name	Edit operator types for which data type is defined	Description of data type [units]	Dimensions loaded
pscale	---	all	Scaling factor to power [-]	. . E . .
ptime	---	all	Time along the path until the actual point [sec]	. . E . .
ptimx	---	all	Time along the path until the preceding point [sec]	. . E . .
pwlvl	---	all	Power level [W/g]	. . E . .
rho	---	MIC	Initial heavy-metal mass [g/cm^3]	R . E . .
rpp	---	all	Identifier of calculational point [-]	. . E . .
su	---	FACE, CUR	The faces' surfaces [cm^2]	R . . . .
sx	---	FACE, CUR	The x-coords of the faces' gravity centers [cm]	R . . . .
sy	---	FACE, CUR	The y-coords of the faces' gravity centers [cm]	R . . . .
tburn	---	all	True burnup [MWd/t]	. . E . .
tdays	---	all	Time of a decay step [days]	. . E . .
tr	trta	MAC, MIC	Transport XS (flux-weighted) [cm^{-1} or barns]	R [I] E G .
uburn	---	all	User-specified burnup [MWd/t]	. . E . .
vo	---	AREA, MAC, MIC	The regions' volumes [cm^3]	R . . . .
vx	---	AREA, MAC, MIC	The *x*-coords of the regions' gravity centers [cm]	R . . . .
vy	---	AREA, MAC, MIC	The *y*-coords of the regions' gravity centers [cm]	R . . . .
xrp	---	all	Sequence number of extra calculational point [-]	. . E . .

Notes:

1) Dimension letters enclosed by brackets [] indicate they are loaded depending on the edit operator type.
2) A digit instead of a dimension letter indicates that the dimension is always loaded with the number of values given by the digit (1 or 6 only).

INDEX(), the index operator

Purpose

The INDEX operator defines index arrays to be used by the Get and Put operations in the FORMULA operator. Index arrays permit to select array elements based on their value or position, and on the value of the elements of another array that has the same structure.

Usage

name = INDEX(***c*/*opt*:*od*[/*iv*]**)

Where:

name	is the name of the index array.
c	is the source array on which the index array is based.
opt	is the operation type. Valid types are: SORT, WINDOW, PERMUTE, RANGE
od	is the data field for the operation. It depends on the operation type, as follows:
	for SORT: **[*direction*] [,*d*]** ***direction*=** *ASCENDING, DESCENDING* ***d*=** *R, I, E, G, O*
	for WINDOW: **[*w1*],[*w2*][;[*w1*],[*w2*]...]** ***w1*/*w2*=** lower/upper limit
	for PERMUTE: ***x1 x2 x3 x4 x5*** ***xi*=** *R*, *I*, *E*, *G*, *O*
	for RANGE: **[*r1*];[*r2*];[*r3*];[*r4*];[*r5*]** ***ri*= *lo*[-[*hi*]][,*lo*[-[*hi*]]...]**
iv	is the initial value for arrays generated by the Put operation in a FORMULA operator.

Rules

1) The INDEX operator does not affect the source array *c*.

2) The operation types can be abbreviated by their first three letters: SOR, WIN, PER, RAN.

3) There are two types of SORT operations: absolute and dimensional.

   Absolute sorting consists of sorting all the values regardless of their location in the array. The resulting array has the dimension *R*.

   Dimensional sorting acts within a *d-set*, which is a set of values in the specified dimension *d* that share the same combination of coordinates of the other dimensions. It sorts the values within each of the individual *d-sets*—each combination of the other dimensions—of the source array, without affecting the values of the other coordinates. The resulting array has the same dimensional structure as the source array.

4) The SORT operation creates an index array with the pointers to all the elements of *c* in ascending or descending order, depending on *direction*. The original sequence is preserved for values that are equal. Absolute sorting in descending order is the default. Absolute sorting collapses the array to one dimension whereas dimensional sorting preserves the array structure.

5) The WINDOW operation type creates an index array with pointers to elements of *c* with values greater than or equal to *w1*, and lower than or equal to *w2*.

6) Elements with values within the limits of two or more *w1,w2* sets will be repeated in the index array.

7) If *w1* (*w2*) is given and *w2* (*w1*) is missing, INDEX will select all elements with values greater (lower) than or equal to *w1* (*w2*). If both *w1* and *w2* are missing, all the elements will be selected.

8) The PERMUTE operation type creates an index array that points to all the elements in the order that results from permuting two or more dimensions. The *xi* entries indicate the new dimension order, the old one being always *R I E G O*.

9) The RANGE index type creates an index array with pointers to the elements selected in the *od* field. Each block *ri* selects a set of values for the dimension *i* (*i=1,2,3,4,5 = R,I,E,G,O*). Valid dimension values are integers from 1 through the number of entries the array has in the *i* dimension. If *ri* is omitted all the values are selected.

10) The RANGE specification *lo-hi* selects all the dimension values between *lo* and *hi*. If *hi* is missing but the hyphen (-) is given, *hi* is assumed equal to the number of entries the array has in that dimension. If both the *hi* value and the hyphen are missing, it is assumed that *hi*=*lo*.

11) Elements selected more than once in the RANGE specifications will be repeated in the index array.

Remarks

The INDEX operator allows the user to select elements of arrays based on their value or position. When used in the Get operation of the FORMULA operator, new arrays are generated that contain only the selected elements.

The selected elements can be placed back into the positions they had in the original array by the Put operation of the FORMULA operator. The combination of Get and Put operations using the same index array works as a filter.

Examples of the combined use of the INDEX operator and the Get and Put operations are given later on in the FORMULA operator.

Examples

1) Idesc = INDEX (ppow / SORT:)

This example makes the index array that contains the pointers to the elements of "ppow" sorted in absolute, decreasing order. When applied, Idesc will generate a one-dimensional array with elements in the sorted sequence.

2) Idesc = INDEX (ppow / SORT: ,R)

Assuming ppow is an array in R (regions) and E (burnups), this sentence returns the pointers to the elements of "ppow" sorted in decreasing order for each burnup step. The two-dimensional characteristic of ppow is preserved in Idesc.

3) Iheavy = INDEX (iso / WINDOW: 90000,95000 /-99999)

This index array contains the pointers to the heavy isotope identifiers in "iso", and defines the initial value for the Put operation equal to -99999.

4) Ibur = INDEX (kinf / RANGE: ; ;2,4,8-10; ;)

This index array points to all the elements of the "kinf" whose dimension E is equal to 2,4,8,9 and 10.

5) Iperm = INDEX (p1 / PERMUTE: R I E O G)

This example creates an index array with pointers that permute the G and O dimensions. Let "p1" be the p1 scattering matrix, then a Get operation over "p1" using this index array will create the transposed p1 matrix.

FORMULA(), the formula operator

Purpose

The FORMULA operator is used to define a new array which is the result of operations involving arrays, constants and special functions.

Usage

name = FOR(*ex*)

Where:

name	is the name of the array.
ex	is an expression containing constants, arrays and built-in functions related by the following mathematical operators:

+	addition.
-	substraction, or change of sign.
*	multiplication.
/	division.
**	exponentiation.
()	forces an evaluation sequence.
^	index Get operation.
^^	index Put operation.

Rules

1) The ZENITH built-in functions are single-argument functions whose name starts with the character "at" (@), and have the general format:

@*nnd*(*ex*) or @*nn*(*ex*)

where:	*nn*	is the two-character function name.
	d	is the array dimension the function acts on (*d*=*R*,*I*,*E*,*G* or *O*).
	ex	is an expression as defined above.

The following functions are presently available in ZENITH:

@*SMd*()	returns the sum of values in *d*.
@*SPd*()	returns the partial sum of values in *d*.
@*PRd*()	returns the product of values in *d*.
@*PPd*()	returns the partial product of values in *d*.
@*AVd*()	returns the arithmetic average value in *d*.
@*MNd*()	returns the minimum value in *d*.
@*MXd*()	returns the maximum value in *d*.
@*NBd*()	returns the *d* value of each array element.
@*LN*()	returns the natural logarithm.
@*AB*()	returns the absolute value.

Each function defines a new array. The arrays defined by the functions @*SPd*(), @*PPd*(), @NBd(), @*LN*() and @*AB*() have the same structure of the argument *ex*. The other functions change the structure of the argument by reducing the dimension *d* to only one element.

The partial sum (product) is defined at each array element as the sum (product) of all elements along dimension *d* whose coordinate number is less than or equal to the coordinate number of the current element.

2) The order of evaluation in mathematical expressions is: first exponentiations, from the right to the left; next multiplications and divisions, from the left to the right; and, finally, additions and substractions, from the left to the right.

3) The following rules apply to operations of the type +,-,*,/ or ** between two members A and B:

   a) The operation is performed element by element, and not according to matrix algebra.

   b) If A and B are constants, the result is another constant.

   c) If one is a constant and the other an array, the result is an array with the same structure as the input array.

   d) If both A and B are arrays, then the number of values in any dimension must be equal in both arrays, unless it is equal to 1 or not loaded for one of them. In this case the array with the smaller dimension is expanded to match the structure of the other array. The resulting array may have a larger structure than that of A or B, for example:

```
A = row (1.0, 2.0, 3.0)      then     A*B =     0.1  0.2  0.3
B = col (0.1, 0.5)                              0.5  1.0  1.5
```

4) The index Get (^) and Put (^^) operations relate an array with an index array generated by the INDEX operator. The index array must be at the right of the operator. The array may also be an expression involving at least one array (array expression).

5) The index Get operation generates a new array whose elements are a subset of the elements in the original array or array expression. The elements are selected according to the criteria used to generate the index array, see INDEX operator.

6) The index Put operation is the inverse of Get. It can only be applied to an array resulting from a Get operation. It creates a new array by placing the elements that were selected with Get back into their original positions, and by assigning the initial value (defined by the INDEX operator) to the undefined elements.

7) Let *i* be the index array generated as *i* = INDEX(*c*/...), then:

r = FOR(*a*^*i*)	is possible only if *a* and *c* have the same number of elements in each dimension. The structure of the output array *r* depends on the index operation type. It is: 5-dimensional for PERMUTE, RANGE and dimensional SORT; 1-dimensional in *R*, for absolute SORT and WINDOW.
r = FOR(*b*^^*i)*	is possible only if the total number of elements of *b* is less or equal to that of *c*. The output array *r* has the structure of *c*.

Remarks

The result of a Put operation applied to the array resulting of a Get operation with the same index array is identical to the original array expression, only if the subset of elements selected with the Get operation includes all the array elements. This is always the case when the INDEX array is generated with a SORT or PERMUTE operation type, see INDEX- operator.

Examples

```
1)   relPPW    = FOR ( 100 * PPW / @avR(PPW) )

2)   maxdev    = FOR (@mxR(@mxE(@mxG(@ab((flux1 - flux2)/flux2)))))

3)   Kinf      = FOR ( (nf1+Sr1*nf2/Sa2) / (Sa1+Sr1) )

4)   IcoorR    = FOR   (@nbR(relPPW) )

5)   Isort     = INDEX (relPPW / SORT:DESCENDING)
     SrPPW     = FOR(relPPW ^ Isort)
     Irange    = INDEX (SrPPW / RANGE:1-10; ; ; ; )
     TopPPW    = FOR(SrPPW ^ Irange)
     Topreg    = FOR( (IcoorR ^ Isort) ^ Irange )
     Toprel    = FOR( (TopPPW^^Irange) ^^ Isort )

6)   A         = ARRAY(R: 1, 2, 3, 4, 5)
     PartSum   = FOR(@spR(A))
     PartPro   = FOR(@ppR(A))
```

1) Let "PPW" be the linear power for each fuel pin of the lattice. Then, "relPPW" is the relative linear power of each pin expressed in percent.

2) Let "flux1" and "flux2" be the fluxes of the same system but calculated with different options by HELIOS. Then, "maxdev" is the maximum relative difference in the flux between the two models.

3) Let "nf1, Sr1, ..." be the two-group macroscopic cross sections as a function of the reactivity point. Then, "Kinf" is the infinite-medium multiplication factor as a function of the reactivity point.

4) The array "IcoorR" has the same structure as "relPPW". Each entry in "IcoorR" is defined as the value of dimension R (the region number) for the same entry in "relPPW".

5) "Isort" is an index array containing the pointers to the elements of "relPPW" sorted in decreasing order. "SrPPW" is the array containing the elements of "relPPW" in decreasing order.

   The index array "Irange" points to the first 10 elements in "SrPPW" and, since the elements were sorted, those are the 10 highest relative powers. The array "TopPPW" contains only the 10 highest relative powers.

   The array "Topreg" contains the region numbers where the 10 highest relative powers occurred. It is created by the same double Get operation that created "TopPPW", but applied to the array with the region numbers "IcoorR".

   The double Put operation over the array "TopPPW" produces the array "Toprel", which has the same structure as array "relPPW" but has only ten elements defined, the remaining ones being 0.0 (the default initial value of the INDEX operator). The non-zero elements in "Toprel" are identical to those in "relPPW".

6) This example illustrates the use of the partial sum and product functions. The three arrays are one dimensional in R and their elements are:

A():	(1.0, 2.0, 3.0, 4.0, 5.0)
PartSum:	(1.0, 3.0, 6.0, 10.0, 15.0)
PartPro:	(1.0, 2.0, 6.0, 24.0, 120.0)

7) Given a burnup array *bu* in three regions (*R*) at two exposure points (*E*):

$$bu = \begin{pmatrix} 1 & 5 & 3 \\ 2 & 6 & 7 \end{pmatrix} = \begin{pmatrix} r1e1 & r2e1 & r3e1 \\ r1e2 & r2e2 & r3e2 \end{pmatrix}.$$

The question is: what are the maximum burnups and in which regions do they occur? The solution goes as follows:

*i*1	=	IND(*bu*/RAN:1;;;;)	⇒	points at the first region, e1r1 and e2r1
ip	=	IND(*bu*/PER:RIEGO)	⇒	identity permutation (^^*ip* restores the structure of *bu*)
id	=	IND(*bu*/SORT:)	⇒	points at the array elements $r3e2$, $r2e2$, $r2e1$, $r3e1$, $r1e2$ and $r1e1$
en	=	FOR(@NbE(*bu*)^*id*)	⇒	registers the exposure points of bu and loads them in R: (2 2 1 1 2 1)
rn	=	FOR(@NbR(*bu*)^*id*)	⇒	registers the region numbers of *bu* and loads them in *R*: (3 2 2 3 1 1)
B	=	FOR(*bu*^*id*)	⇒	sorts *bu* in descending order and loads them in *R*: (7 6 5 3 2 1)
ia	=	IND(*en*/SOR:asc)	⇒	points at positions 3, 4, 6, 1, 2 and 5 in *en* (and *rn*); the first three positions are those of the first exposure point ($e1$), the last three those of the second ($e2$)
mb	=	FOR(((*B*^*ia*)^^*ip*)^*i*1)	⇒	(*B*^*ia*) = (5 3 1 7 6 2); ((*B*^*ia*)^^*ip*) = $\begin{pmatrix} 5 & 3 & 1 \\ 7 & 6 & 2 \end{pmatrix}$; (((*B*^*ia*)^^*ip*)^*i*1) = $\begin{pmatrix} 5 \\ 7 \end{pmatrix}$
mp	=	FOR(((*rn*^*ia*)^^*ip*)^*i*1)	⇒	(*rn*^*ia*) = (2 3 1 3 2 1); ((*rn*^*ia*)^^*ip*) = $\begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 1 \end{pmatrix}$; (((*rn*^*ia*)^^*ip*)^*i*1) = $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$

mb shows the two highest region burnups and *mp* gives the corresponding region numbers.

An alternative and easier way to find the solution is using <u>dimensional sorting</u>, as shown in the following lines:

```
i1   = IND(bu/RAN:1;;;;)
id   = IND(bu/SORT:,R)
mb   = FOR((bu^id)^i1)
mp   = FOR((@NbR(bu)^id)^i1)
```

8) Find for isotope $*is* the up, down and self-scattering of its 6×6 P_0 matrix.

Uni = ARR(R: 1; 6*0; 1; 6*0; 1; 6*0; 1; 6*0; 1; 6*0; 1)

⇒ *Uni* contains the elements of the unitary 6×6 matrix

*p*0 = SEL(*p*0/mic/imp;mi/$*is*)) ⇒ selects the P_0 matrix of $*is* from the HELIOS micro *mi*

*ip*0 = IND(*p*0/PER:RIEGO) ⇒ *ip*0 will restore the structure of *p*0 of any array with 36 or less elements

dia = FOR(*Uni* ^^*ip*0) ⇒ *dia* has the structure of *p*0, i.e. a unitary 6×6 matrix in dimensions *G* and *O*

sca = FOR(@SmG(*p*0)) ⇒ scattering cross section per originating group *O*

ig = IND(*sca*/PER:RIEOG) ⇒ interchange pointers to receiving and originating groups, *G* and *O*

sca = FOR(*sca*^*ig*) ⇒ now *sca* is the scattering cross section per group *G*

slf = FOR(@SmO(*p*0**dia*)) ⇒ $$(p0*dia)=\begin{pmatrix} g1o1 & g1o2 & \cdots \\ g2o1 & g2o2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} * \begin{pmatrix} 1 & 0 & \cdots \\ 0 & 1 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} g1o1 & 0 & \cdots \\ 0 & g2o2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

@SmO() reduces the *O* dimension to 1, leaving the self-scattering in *G*

up = FOR(@SmO (@SpG(*p*0)**dia*) - *slf*)

$$(@SpG(p0)*dia)=\begin{pmatrix} g1o1 & 0 & 0 & \cdots \\ 0 & g1o2+g2o2 & 0 & \cdots \\ 0 & 0 & g1o3+g2o3+g3o3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

@SmO() reduces the *O* dimension to 1, leaving the self + up-scattering in *G*
Subtracting *slf*, finally yields the up-scattering in *G*

do = FOR(*sca* - *slf* - *up*) ⇒ the down-scattering per group *G*

9) Arrange the isotopes, their number densities and their absorption rates relative to the total production rate (in units of 10^{-5}) in order of increasing identifiers. The ZENITH import operator is *Case* and the AURORA micro operator is *mi*, which is assumed to be in 1 energy group, for 1 region, but at more than 1 exposure (1-I-E-1-.).

id	= SEL(*id*/MIC/*Case*;*mi*)	selects the identifiers *id* (.-I-.-.-.)
nd	= SEL(*nd*/MIC/*Case*;*mi*)	selects the number densities *nd* (1-I-E-1-.)
fx	= SEL(*fx*/MIC/*Case*;*mi*)	selects the flux *fx* (1-.-E-1-.)
ab	= SEL(*ab*/MIC/*Case*;*mi*)	selects the microscopic absorption cross sections *ab* (1-I-E-1-.)
nf	= SEL(nf/MIC/*Case*;*mi*)	selects the production cross sections *nf* (1-I-E-1-.)
Ptot	= FOR(@SmI(*nd*∗*nf*∗*fx*))	the total fission-production rate (1-1-E-1-.)
ab	= FOR(100 000∗*nd*∗*ab*∗*fx*/*Ptot*)	the isotopic absorption rates (1-I-E-1-.)
x	= FOR(0∗*nd*+*id*)	*x* has the structure of *nd* (and *ab*) with at all exposures the isotope identifiers

The remaining instructions depend on whether absolute or dimensional sorting is used:

----- ABSOLUTE SORTING -----

*i*1	= IND(*id*/PER:RIEGO)	the 'put' operation (^^*i*1) restores the original structure of *id*
*i*2	= IND(*id*/SOR:Asc)	points at the identifiers *id* in order of increasing value
*i*3	= IND(*x*/ PER:RIEGO)	the 'put' operation (^^*i*3) restores the original structures of *x* (and *nd* and *ab*)
*i*4	= IND(*x*/SOR:Asc)	points at the elements of *x* in order of increasing value
C	= FOR((@NbE(*x*))^*i*4)	@NbE(*d*) has the structure of *x*, with at all identifier positions the exposure sequence numbers; in *C* (which is loaded in *R* only) they are ordered according to the identifiers associated with the exposure sequence numbers
*i*5	= IND(*C*/SOR:Asc)	points at the elements of *C* in order of increasing exposure
IdIso	= IND((*id*^*i*2)^^*i*1)	the identifiers in increasing order; (^^*I*1) restores the structure from *R* back to *I*
NdIso	= IND(((*nd*)^*i*4)^*i*5)^^*i*3)	the number densities corresponding to *IdIso*; (^*i*3) restores the structure of *nd*
AbIso	= IND(((*ab*)^*i*4)^*i*5)^^*i*3)	the absorption rates corresponding to *IdIso*; (^*i*3) restores the structure of *ab*

----- DIMENSIONAL SORTING -----

i2 = IND(*id*/SOR:Asc,I)
i3 = IND(*x*/SOR:Asc,I)
IdIso = FOR(*id^i2*)
NdIso = FOR(*nd^i3*)
AbIso = FOR(*ab^i3*)

IdIso contains the sorted identifiers, and *NdIso* and *AbIso* the respective number densities and absorption rates.

LIST(), the list operator

Purpose

The LIST operator is used to define a set of arrays whose values in a selected dimension will be printed columnwise in the output file. A column is printed for each combination of the other four dimensions.

Usage

name = LIST(**[*lb*][;*p*] / [*ti*[;*ti*...]] / *di* / [*f*:]*a*[;[*f*:]*a*...]**)

Where:

name	is the list name.
lb	is an array name defined by a SELECT operator, whose labels will be used.
p	is the layout control, of the form: **[*S*][*P*][;[*nc*][;*cw*]]**

S	if present, suppresses the coordinate strings;
P	if present, suppresses paging so that each list starts right after the previous one without enforcing a page jump;
nc	maximum number of columns in one block of the list;
cw	column width in characters.

ti	is one line of title to be printed.
di	indicates the array dimension that will be printed columnwise, *di*=*R*,*I*,*E*,*G*,*O*.
f	is one of the following format indicators:

En	indicates "E" format with 0≤*n*≤9 decimals.
Fn	indicates "F" format with 0≤*n*≤9 decimals.

a	is the name of an array to list.

Rules

1) If the array *lb* is specified, the list will be labeled with the names of the area or faces, isotope identifiers, path and state names, burnup values and group names corresponding to array *lb*. *lb* can be any array that was defined by a SELECT operator, including one of the arrays to list. Label names longer than 60 characters are truncated.

2) All columns—except the first if it contains the label—have equal width, which is selected automatically (*cw* = 1) or defined manually for all columns (*cw* > 1).

   When automatic width is selected, the column width is set so that it can accommodate figures, array names, coordinate strings and labels.

   If manual width is selected and a field needs a larger column width than specified, it will be truncated. Truncation is indicated with an asterisk (*). The number of truncated fields (if any) is reported at the end of the list.

3) The semicolon (;) separates the arrays to list in the field [*f*:]*a*[;[*f*:]*a*…]. If a double semicolon is given instead, the next array is forced to start in column one.

Remarks

The coordinates string is a text string which, if not suppressed by the option *S*, is printed above each column after the array name. Its format is:

x1-x2-x3-x4-x5

where *x1,x2,...* are the *R*,*I*,*E*,*G*,*O* coordinates of the column. The dimension of the column is indicated with a letter. The undefined dimensions of the array are marked with a dot (.); the other dimensions give the actual sequence number.

If an array is empty, as the result of some INDEX operations, or by selecting a data type from a macro or micro that doesn't exist, a dash (-) is printed where the first element of the array should have been listed.

The following default values apply:

nc	=	40
cw	=	1 (automatic column width)
f	=	E5 (floating point 5 digits)

coordinate string printed
paging active.

Examples

```
1) Ktable     = LIST ( /'Kinf versus Burnup' /E/ f0:BUR; f5:KINF)

2) XStable    = LIST (;S/'Cross section table'/E/
                        f0:BUR; e5: D1; D2; nf1; nf2; Sa1; Sa2; Sr1)
```

1) If "KINF" and "BUR" are arrays with the infinite multiplication factor and the average burnup, then a table of the multiplication factor (fixed format, 5 decimals) as a function of the burnup (fixed format, no decimals) is printed.

2) Similarly to Example 1, a table of cross sections (floating point format, 5 decimals) as function of burnup is produced. Automatic column width is used. No coordinate string is printed.

MAP(), the map operator

Purpose

The MAP operator is used to select a set of arrays to be printed in the output file as X-Y maps. The arrays values in the dimension *R* are placed in the map at the specified coordinates. A map is printed for each combination of the other four dimensions.

Usage

name = MAP(**[*lb*][;±*pn*]/[*nx*;*ny*]/[*ti*;[*ti*...]] / *cx*[:*rx*];*cy*[:*ry*]/*a*[;*a*...])**

Where:

name	is the name of the map.
lb	is an array name defined by a SELECT operator, whose labels will be used.
pn	is 10**p*+*n*, with: *n* - the number of significant digits printed at the map positions; $1 \le n \le 6$. *p* - printout option for extra region map: 1 - only if region overlapping occurs. 2 - only for the first map. 3 - region map printed for every map.
±	is the option to control map layout: + - each map starts on a new page. - - only the first map starts on a new page.
ti	is a line of title.
nx ny	are the sizes in X and Y of the display area.
cx cy	are arrays with the X- and Y-coordinates of the points.
rx ry	are the ranges in X and Y, in the format: *rx* = [*xmin*],[*xmax*] *ry* = [*ymin*],[*ymax*]
a	is the name of an array to display.

Rules

1) If the array *lb* is specified, the list will be labeled with the names of the area or faces, isotope identifiers, path and state names, burnup values and group names corresponding to array *lb*. *lb* must be defined by a SELECT operator; it may be one of the arrays to be printed. Label names longer than 60 characters are truncated.

2) The display area is the rectangle in the output listing where the values of the array are displayed. It is *nx* characters wide and *ny* lines high (by default *nx* = 90 and *ny* = 45). If *nx* > 116, the display area will be split in blocks.

3) The arrays *cx* and *cy* will normally be the coordinates of the gravity centers, defined by the SELECT operator (see there).

4) Only array values with coordinates within the window range (*xmin-xmax, ymin-ymax*) are displayed. This feature allows zooming in a particular area of interest. The limits are set by default to the corresponding extremes of *cx* and *cy*.

5) The regions or faces are numbered according to the specifications given in the BEGIN operator. The default numbering is from left to right and from top to bottom.

Remarks

If the printed values of the array elements for two or more regions overlap in the display area, a letter identifier is written instead. The values of the overlapping elements are then listed with their letter identifier after each map.

The option *pn* is used to set the number of digits printed in the map and to control the printout of the extra map containing the region or face numbers. By default *pn*=+14, which means each map starts on a new page, prints 4 digits at each position and produces the extra map only if overlapping occurs.

The location with the maximum map value is marked with an asterisk appended to the number or to the region identifier.

The centers of gravity of the regions or faces as provided by HELIOS are normally used as the coordinates arrays *cy* and *cy*. However, the map can be rotated, inverted or translated through permutation, change of sign and linear transformations of the coordinate arrays using the FORMULA operator.

Examples

```
1)   PinMap   = MAP (// 'Full assembly pin powers'/ CX; CY/ PINPOW)

2)   '1/4map' = MAP (;+34 / 20;40 / '1/4 assembly pin powers' /
                              CX:0.0,7.5; CY:0.0,7.5 / PINPOW )
```

1) This example produces maps of the array "PINPOW" with the coordinates given in "CX" and "CY". Default values apply for all the options.

2) This prints a partial map of the same array focused on the range (0-7.5, 0-7.5), and an extra region map for every map. The display area is set to 20x40.

EXPORT(), the export-case operator

Purpose

The EXPORT operator saves a data set made up of user-defined arrays into a HERMES file. This data set forms a case in the external file, which has the structure required by TABGEN.

Usage

name = EXPORT(**[+:]*fi*/[*p*[;*p*...]]/*a*[;*a*...] / *b*:*m*[,*m*...][;*b*:*m*[,*m*...]...]**)

Where:

name	is the name of the case to be defined in the HERMES file.
+	This sign preceding *fi* causes ZENITH to overwrite an existing case in the file.
fi	is the name of the HERMES file.
p	is the name of a catalog in the path to reach *name*.
a	is the name of an array saved in the HEADER catalog file.
b	is the name of an array defined only in the E dimension, that contains the burnup values.
m	is the name of an array saved in one of the SET catalogs of the file.

Rules

1) The HERMES file, if not existing, will be created by ZENITH. If the file exists and contains a case with the same name, ZENITH will abort the run unless the + sign precedes the file name.

2) Each data set in the HERMES file consists of one HEADER catalog and one or more SET catalogs. Each catalog contains lists where the data are stored.

3) In the lists of the HEADER catalog, array values are stored as one-dimensional structures. All the array values are saved contiguously following the sequence R, I, E, G, O, with R varying fastest.

   <u>Note</u>: If the HERMES file is to be processed by TABGEN, every HEADER array must be a one-element array. TABGEN expects only one value per array name.

4) In the lists of the SET catalogs, array values are stored as two-dimensional structures. The second dimension corresponds to the E array dimension, and the first to the product of the R, I, G, O array dimensions, with R varying fastest. The burnup values for the E entries are given in the array *b*, which must have only its E-dimension defined.

Remarks

For a description of the file structure, see Chapter 3, "Structure of the Output HERMES File".

Examples

1)
```
'Lattice 1' = EXPORT('REACTOR-DB.HRS'/HELIOS; 'Cycle 1' /
                 PWLVL; AVENR; VOID; VFUEL; AVUDN; NPINS;
                 MODNS; TFUEL; TMODR; VORAT /
                 Uburn:   NU235, NU236, NU238, NPUN9, NP241,
                          NX135, D1, D2, SIGA1, SIGA2, SIGR1, EFISS,
                          SIGN1, SIGN2, SIGF1, SIGF2, XE1, XE2,
                          PPF, KINF, FX2, B1, B2, B3, B4, B5, B6,
                          L1, L2, L3, L4, L5, L6, VTH, VEPI,
                          NP240, NP242, NS149, BOR1, BOR2,
                          DET1, DET2, PWMAP, DISCF)
```

1) This example shows how a set of data is saved with the name "Lattice 1" in the HERMES file "REACTOR-DB.HRS". The catalog path to this data set is: #\HELIOS\Cycle 1\Lattice 1\.

CONVERT(), the convert-file operator

Purpose

The CONVERT operator is used for file format conversions between the HERMES binary and an ASCII format.

Usage

target = CONVERT(***di*:*fo*/*source*[/*p*[;*p*...]]**)

Where:

target	is the name of the target file.
di	specifies the direction of the conversion. Valid entries are: *from* and *to*.
fo	is the format of the file that is not the HERMES binary file.
source	is the name of the source file.
p	is the name of a catalog in the path to the top catalog of the branch to be converted.

Rules

1) The target file is the result of the conversion, which always goes from source to target. The target file will be created if not existing, and overwritten if existing.

2) One of the file formats is always the HERMES binary, the other format is specified by *fo*. Presently this name can only be *TA*, short for TABGEN ASCII.

3) The key *di* specifies the direction of the conversion: *from* indicates conversion from format *fo* to HERMES binary, while *to* indicates conversion from HERMES binary to format *fo*.

4) For conversions from HERMES binary format (*di*=*to*), the last catalog in the path is the top catalog of the branch to be converted. If no path is given, the entire file is converted.

5) For conversions to HERMES binary format (*di*=*from*), the last catalog in the path is the catalog from which the converted file structure will pend. If no path is given, the converted file structure will pend from the root directory (#).

Remarks

The conversion process is general and not based on the catalog structure of the HERMES files. Therefore it works with any HERMES file.

The ASCII file format is that of the program TABGEN. It contains the name, catalog path and contents of all the lists of the binary HERMES file.

For a description of the file format, see Chapter 4. "Format of the Output ASCII File".

Examples

```
1)  'hfile.asc'  = CONV(to:TA/ '[BB.ZEN]hfile.dat')

2)  'fuel1.asc' = CONV(to:TA/ '[BB.ZEN]hfile.dat'/ dbank;fuel1)

3)  'fuel1.dat'  = CONV(from:TA/ 'fuel1.asc')
```

1) This example generates the ASCII format of the file "hfile.dat" located in the directory "[BB.ZEN]".

2) Here only the branch defined by the path "dbank;fuel1" is extracted from the file "hfile.dat" and converted to the ASCII format.

3) The ASCII file from example 2) is here converted to the HERMES binary format. The resulting file contains only the branch below the path "dbank;fuel1" of the source file of the example 1).

PRINT(), the print operator

Purpose

The PRINT operator is used to write arrays into a formatted file without pagination, thus providing a convenient way to transfer output data to other codes.

Usage

file = PRINT(**[+]/[tag]/aname1[;aname2...]**)

Where:

file	is the name of the print file.
+	indicates the arrays will be added to an existing print file.
tag	is an optional character string for identification purposes.
anamei	is the name of a previously defined array, i=1, NARR where NARR is the number of arrays specified.

Rules

1) If *file* exists and is write-protected, the run will be aborted by the system.

2) If the file exists and the plus sign (+) is specified, ZENITH writes the new data at the end of it, thus preserving its previous contents. If the file exists but no plus sign is given, the file will be overwritten and its previous contents will be lost. The print file always is created by ZENITH if not existing, regardless of whether the plus sign is specified or not.

3) Each occurrence of the PRINT operator may use the same or a different file name. Several print files can thus be created in a single run.

4) Each occurrence of the PRINT operator creates a new block of data in the print file. The structure of this block is uniform and explained in Remarks. Blocks of data are written in the sequence of the PRINT operators without any divider or pagination.

5) The string *tag* (80 character maximum) can be any text used to identify the block. If not given, a blank text will be assumed.

6) Any number of arrays with different structures and dimensions can be specified.

Remarks

The following FORTRAN sentences can be used to read a block of data written by the PRINT operator on a file opened on the LU unit.

```
READ(LU,'(A80)') TAG
READ(LU,'(I7)') NARR

DO N=1,NARR
  READ(LU,'(A80)') ANAME(N)
  READ(LU,'(5I7)') NR(N), NI(N), NE(N), NG(N), NO(N)
  KMAX = NR(N)*NI(N)*NE(N)*NG(N)*NO(N)
  READ(LU,'(5E14.6)') (A(K,N), K=1,KMAX)
END DO
```

Upon execution ANAME() will contain the array names; NR(), NI(), NE(), NG() and NO() the sizes of the dimensions R,I,E,G,O of each array; and A() the array values.

Examples

1) 'FluxandXS.prt' = PRINT(// fx; ab; fi; nf)

   This example writes the contents of the arrays “fx”, “ab”, “fi” and “nf” to the file “FluxandXS.prt”. One block of data will be written and, since the tag is omitted, the block will be headed by a blank line.

2) 'FluxandXS.prt' = PRINT(/‘Fluxes’ / fx)
   ‘FluxandXS.prt' = PRINT(+/‘XS’ / ab; fi; nf)

   These two PRINT operators write the same arrays as in example 1, except that now two blocks will be written. The first block is headed by the string “Fluxes” and contains the array “fx”. The second block has the other three arrays and is headed by the string “XS”.

   The first sentence has no plus sign, so the file "FluxandXS.prt" will be created if it did not exist. If it existed, its contents will be destroyed and replaced by the block "Fluxes". The plus sign in the second PRINT sentence instructs ZENITH to add the block "XS" to the file "FluxandXS.prt" (behind the block "Flux").

PARAMETER(), the define-parameter operator

Purpose

The PARAMETER operator is used to define the contents of a parameter. After being defined, a parameter can be used in other input operators as a symbolic reference to its contents.

Usage

$*name* = PARAMETER(***co***)

Where:

name	is the parameter name.
co	is a text string with the contents.

Rules

1) A parameter name must always be preceded by the dollar sign ($). It must comply with the ZENITH rules for names.

2) The contents of a parameter can include any kind of data. The only limitations are that strings in apostrophes must be entirely contained, and that parenthesis must be balanced within the contents of the parameter.

3) The contents can include reference to any other previously defined parameter.

4) After being defined, a parameter can occur anywhere in the input and it will be replaced by its contents. However the following restrictions apply:

   - A parameter cannot be part of a pre-processor instruction.
   - A parameter cannot be part of a SET operator.
   - The general form of an operator: name = oper() cannot be broken down with the use of parameters.

5) Parameter substitution takes place only after the SET operators have been resolved. Therefore, there is no limitation concerning the use of parameters in operators contained in sets.

6) The seven parameters $*P1*, $*P2*, … $*P7* are the command-line parameters. Unless redefined in the input file, they contain the arguments 1 through 7 given in the command line that activates the ZENITH program.

7) The command $*P1* contains the name of the input file which is the first (mandatory) argument on the command line, following the name that activates the pro-

gram. Arguments following the input file name are optional and, if present, can be used to transfer data from the command line into the input file.

Remarks

Systematic use of parameters to define data fields required by many operators can simplify the input and make it easier to change. One example is the selection of reactivity points in the SELECT operator.

The combined use of parameters and SET operators is a powerful tool to build large input files based on input "subroutines" that perform a defined task.

Examples

1)
```
$case      = PAR (  C1; allpins  )
$rpoints   = PAR (  Path1:(void2)0@1-1000,1000@2-30000;
                    Path2:(void2)35000-50000  )
$HIso       = PAR ( $rpoints / 92235, 92236, 92238, 92237,
                    93239, 94239, 94240, 94241, 94242  )
nhi        = SEL (  nd / micro / $case / $Hiso  )
nufhi      = SEL (   nf / micro / $case / $Hiso  )
```

"nhi" and "nufhi" are respectively the number densities and fission neutron production of the heavy isotopes present in the micro-type data set "allpins" (defined in AURORA).

The parameter "$case" contains the references to the IMPORT operator and to the name "allpins". The parameter "$rpoints" contains the reactivity points for the SELECT operator and is referred to by "$Hiso" which, finally, contains the reactivity points and the list of heavy isotopes.

SET(), the include operator

Purpose

The SET operator is used to retrieve sets of input operators stored by AURORA in a HERMES file, and merge them with the user's input file.

Usage

&*name* = SET(***fi/[p[;p...]]***)

Where:

&*name*	is the name of the input set to be included.
fi	is the name of the HERMES file.
p	is the name of a catalog in the path to the input set *name*.

Rules

1) The first character of the left-hand side must always be an ampersand (&). The name following the ampersand must comply with the rules for names in the ZENITH input.

2) *fi* is the file where the set of operators has been stored using AURORA.

3) SET is the only input operator of ZENITH where parameters cannot be used.

4) The set of operators can contain references to other SET operators. The only limitation is that a SET cannot contain a direct or indirect reference to itself.

5) There is no limitation concerning the use of parameters in input operators within a set, but the restriction imposed by Rule 3 on SET operators still apply.

6) A set cannot contain any pre-processing instruction.

7) A set can include any ZENITH input operator except BEGIN and END.

Remarks

The SET operator allows the use of general-purpose sets of input operators for ZENITH, where case-specific information is passed via parameters.

Examples

Assume that the set "&ListArea" is defined as:

```
vo    = SELECT ( vo / area / $case )
vx    = SELECT ( vx / area / $case )
vy    = SELECT ( vy / area / $case )
Li    = LIST( ;S / 'Area data types' / R / F3: vo; vx; vy )
```

and is stored in the data base file "sets.hms" under the catalog "List".

1)
```
$case       = PARAMETER ( C1; allpins )
&ListArea   = SET ( 'sets.hms' / List )
```

The parameter "$case" contains the reference to an IMPORT- operator (C1) and the name of the AURORA AREA operator "allpins".

The line containing the SET operator in the input file will be expanded as follows:

```
vo    = SELECT ( vo / area / C1; allpins )
vx    = SELECT ( vx / area / C1; allpins )
vy    = SELECT ( vy / area / C1; allpins )
Li    = LIST( ;S / 'Area data types' / R / F3: vo; vx; vy )
```

The expanded input produces a list with the volumes and the coordinates of the gravity centers of the regions.

END(), the end-case operator

Purpose

The END operator marks the end of the input instructions for one ZENITH case.

Usage

```
END()
```

Rules

1) The END operator is mandatory and must be the last operator of any input case.

2) It has no arguments but the parentheses must be given.

3) No left-hand side name is defined by the END operator.

4) It cannot be included in a set.

Remarks

None.

Examples

Not needed.

CHAPTER 3

STRUCTURE OF THE HERMES OUTPUT FILE

The binary output file is a hierarchical HERMES database file with catalogs and data lists. The convention used to describe its structure is that of the HERMES package (see technical note T1/41.15.08, Rev 1: "HERMES, a FORTRAN-77 Package for Data Management", 29 Nov 1991).

Catalog Tree

```
# --------------------- path ----------------------- case -------------------- HEADER
                                                          |
                                                          |---------- SET001
                                                          |
                                                          |---------- …
                                                          |
                                                          |---------- SETxxx
```

"path" and "case" represent the catalog path and the left-hand side name in the EXPORT operator.

HEADER Catalog

The case data that are not burnup-dependent are contained in this catalog. It contains five data lists, described below.

H-SIZES Integer type (4 elements, 1 word per element):

NH	number of array names.
NWPE	number of words per element in list H-NAMES.
NSETS	number of SETmmm catalogs contained in "case".
NHDAT	number of data in list H-DATA.

H-NAMES Text type (NH elements, NWPE words per element):

HNAM()	array with the array names.

H-NAMLEN Integer type (NH elements, 1 word per element):

HLEN()	number of characters in each HNAM() name.

H-DATSIZ	Integer type (NH elements, 1 word per element):
HDSZ()	number of data in each array.

H-DATA	Real type (NHDAT elements, 1 word per element):
HDAT()	the array values, arranged as: (HDAT(1:HDSZ(N)),1:N), N=1,NH)

SETxxx Catalogs

The burnup-dependent case arrays are contained here. There are NSETS catalogs named SET001, SET002 … , one for each array. The following six lists are defined in each SETxxx catalog.

S-SIZES	Integer type (4 elements, 1 word per element):
NB	number of burnup points.
NP	number of arrays stored (always 1).
NVPP	number of values per burnup point.
NWES	number of words in the S-NAMES list.

S-BURNUP	Real type (NB elements, 1 word per element):
SBUR()	array with the burnup points.

S-NAMES	Text type (NP elements, NWES words per element):
SNAM()	name of the burnup-dependent array.

S-NAMLEN	Integer type (NP elements, 1 word per element):
SLEN()	number of characters in the array name.

S-PARAM	Real type (NVPP*NB*NP elements, 1 word per element):
SDAT()	the array values, arranged as: SDAT(1:NVPP,1:NB,1:NP).

S-RIGO	Integer type (4*NP elements, 1 word per element):
SRIG()	the number of entries in dimension R, I, G, and O. (the absolute value of their product = NVPP)

CHAPTER 4
FORMAT OF THE ASCII OUTPUT FILE

The ASCII file is generated by writing the information of every list to the HERMES output file in three records. The lines of the ASCII file are 132-character long.

Each of the three records is identified by a four-letter keyword as the first word of each line. The keywords are HEAD, PATH and CONT. HEAD takes only one line, but PATH and CONT may extend through more lines. To improve readability, a blank line separates the blocks of lines that belong to different lists. When reading from an ASCII file, lines starting on a name other than HEAD, PATH or CONT are ignored. The reading uses a free-format subroutine, so no special data alignment is required.

The HEAD Record

HEAD LTYP NOFELS NEPL NWPE NPATH \

where:

LTYP	list type identifier: 1, 2, -K: 1/2 means integer/real -K indicates a text list starting in column K on the CONT record (column 1 is that of the letter C in CONT).
NOFELS	number of elements in the list.
NEPL	number of elements in any CONT line except the last one, that may contain less than NEPL elements.
NWPE	number of words per element.
NPATH	number of catalogs in the PATH record (not including the HERMES root catalog #).
\	delimiter of catalog names in the PATH record.

The PATH Record

PATH \cat-1\cat-2\ … \cat-NPATH\

where:

cat-i	the i'th name in the catalog path.

The CONT Record

CONT data-1 data-2 … data-NMAX

where:

data-i	the i'th data (integer, real or character name) on the list.
NMAX	NOFELS*NWPE for integer and real list types.
NOFELS	for text types, because the NWPE words in one element make one data-i entry without blanks in between for better readability of names.

CHAPTER 5

SAMPLE PROBLEM

Short Input

```
BEGIN ('ZENITH sample case 01')
%OPTIONS(LINPAG=60)

! ====================== Parameters for output sets ======================= !
$Hcase  = PAR('BWR-4x4')
$Hfile  = PAR('AR931A-02.HRF')
$Path   = PAR(burn40)
$State  = PAR(v40u)

$Burnup = PAR(0,1000,2000-30000)
$Burgam = PAR(0,1000,2000-30000)

! -------------------------------- Options --------------------------------- !
%'basic_output'       = 1         !1/0: y/n basic output
%'xenon_and_samarium' = 1         !1/0: y/n Xe and Sm yields and concentration
%'maximum_burn_power' = 1         !1/0: y/n maximums of burnup and power maps
%'delayed_neutrons'   = 1         !1/0: y/n delayed neutron fraction & velocity
%'neut_det_response'  = 1         !1/0: y/n neutron detector response factors
%'gamm_det_response'  = 1         !1/0: y/n gamma detector response factors
%'burnup_map'         = 1         !1/0: y/n burnup distr map
%'power_maps'         = 1         !  0: no power maps produced
                                  !  1: power peaking map
                                  !  2: 1 + linear power map [W/cm]
%'gamma_smearing'     = 3         !  0: no gamma smeared results
                                  !  1: gamma smeared power peaking factors
                                  !  2: 1 + gamma smeared power peaking maps
                                  !  3: 2 + gamma smeared power maps [W/cm]
%'fast_fluence'       = 1         !1/0: y/n fast fluence box & clad
%'discont_factors'    = 1         !1/0: y/n discontinuity factors
%'av_fuel_isotopics'  = 1         !1/0: y/n average fuel isotopics
%'fuel_pin_isotopics' = 1         !1/0: y/n number densities per fuel pin, only
                                  !     for the isotopes given in $Isos
                                  !     (if none, all isotopes are selected)
$Isos = PAR(92238; 94239)
! ======================================================================== !
!     <<<<<<<<<<<<<<<<<<<<<< End of Users Input >>>>>>>>>>>>>>>>>>>>>>>>      !
! ======================================================================== !
$calp        = PAR($Path:($State)$Burnup)
$calpg       = PAR($Path:($State)$Burgam)
&Common      = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
&BuPow       = SET('AR931A-01.HRF' / SAMPLE; ZENITH)

%if ('basic_output'.eq.1)        then
&Basic       = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('maximum_burn_power'.eq.1)  then
&MaxBuPow    = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('gamma_smearing'.ne.0)      then
&GammaSmear  = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
ppfGam       = LIST(Bugam;S/ 'Gamma smeared power peaking factor' /E/
                          f4:'Gamma smeared ppf')
%end if
```

```
%if ('xenon_and_samarium'.eq.1)  then
&XenSam      = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('delayed_neutrons'.eq.1)    then
&DelayNeut   = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('neut_det_response'.eq.1)   then
&NeutronDetect = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('gamm_det_response'.eq.1)   then
&GammaDetect = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('fast_fluence'.eq.1)        then
&FastFluence = SET('AR931A-01.HRF' /SAMPLE; ZENITH)
%end if

%if ('discont_factors'.eq.1)     then
&DiscFactors = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('av_fuel_isotopics'.eq.1)   then
&AvFuel      = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('fuel_pin_isotopics'.eq.1)  then
&PinFuelIso  = SET('AR931A-01.HRF' / SAMPLE; ZENITH)
%end if

%if ('burnup_map'.eq.1)          then
BuMap        = MAP(burnup; 25 /$mapsize/ 'Burnup distribution [Mwd/t]'
                                            / cxfuel; cyfuel/ burnup)
%end if

%if ('power_maps'.gt.0)          then
PPFMap       = MAP(fx; 15 /$mapsize/ 'Power peaking distribution'
                              / cxfuel; cyfuel/ 'power peaking')
%end if

%if ('power_maps'.gt.1)          then
PowMap       = MAP(fx; 15 /$mapsize/ 'Linear power distribution [W/cm]'
                                              / cxfuel; cyfuel/ power)
%end if

%if ('gamma_smearing'.ge.2)      then
PeakGam      = MAP(fxpin; 15 /$mapsize/
                  'Gamma smeared power peaking distribution'
                 / cxfuel; cyfuel/ 'Gamma smeared pow peak')
%end if

%if ('gamma_smearing'.eq.3)      then
PowGam       = MAP(fxpin; 15 /$mapsize/
                  'Gamma smeared linear power distribution [W/cm]'
                            / cxfuel; cyfuel/ 'Gamma smeared pow')
%end if

END()
```

Expert Input

```
! ------------------------------------------------------- Common definitions !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &Common)
! -------------------------------------------------------------------------- !
C1       = IMP(HELIOS;$Hcase/ $Hfile)
Burnup   = SEL(uburn / sta / C1;   / $calp)
$mapsize = PAR(58;13)
                                             ! Compute the power distribution !
fx       = SEL(fx    / mac / C1;Maps/ $calp)
kf       = SEL(kf    / mac / C1;Maps/ $calp)
vo       = SEL(vo    / mac / C1;Maps)
power    = FOR(fx*kf*vo)


! ------------------------------------------------- Macroscopic neutron data !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &Basic)
! -------------------------------------------------------------------------- !
'K-inf'  = SEL(kinf  / mac / C1;XSset/ $calp)
'K-eff'  = SEL(eigv  / mac / C1;XSset/ $calp)
'Mig-area' = SEL(msq / mac / C1;XSset/ $calp)
'Crit-B2'  = SEL(bsq / mac / C1;XSset/ $calp)

dr       = SEL(dr    / mac / C1;XSset/ $calp)
tr       = SEL(tr    / mac / C1;XSset/ $calp)
ab       = SEL(ab    / mac / C1;XSset/ $calp)
fi       = SEL(fi    / mac / C1;XSset/ $calp)
nf       = SEL(nf    / mac / C1;XSset/ $calp)
kf       = SEL(kf    / mac / C1;XSset/ $calp)
p0       = SEL(p0    / mac / C1;XSset/ $calp)
fx       = SEL(fx    / mac / C1;XSset/ $calp)

i1       = IND(ab    / RAN:;;;1;)
i2       = IND(ab    / RAN:;;;2;)
i12      = IND(p0    / RAN:;;;1;2)
i21      = IND(p0    / RAN:;;;2;1)

D1       = FOR((dr/(3*tr))^i1)
D2       = FOR((dr/(3*tr))^i2)
Abso1    = FOR(ab^i1)
Abso2    = FOR(ab^i2)
Fiss1    = FOR(fi^i1)
Fiss2    = FOR(fi^i2)
Nufi1    = FOR(nf^i1)
Nufi2    = FOR(nf^i2)
Remo1    = FOR((p0^i21)*Abso2/(Abso2+(p0^i12)))
Flux1    = FOR(fx^i1)
Flux2    = FOR(fx^i2)
'Reso esc prob' = FOR(Remo1/(Remo1+Abso1))

Basic    = LIST(Burnup;S/
               'Path/tree name, state name, burnup, k-inf, k-eff';
               'migration area [cm2], critical buckling [cm-2] and';
               'resonance escape probability'
                /E/
                f5:'K-inf'; 'K-eff'; e5:'Mig-area'; 'Crit-B2';
                f5:'Reso esc prob')
XSdata   = LIST(Burnup;S/
               'Average macroscopic cross sections and fluxes (2 groups)';
               'Cutoff energy = 1.8 ev'
                /E/
                f5:D1; D2; e5:Abso1; Abso2; Remo1; Nufi1; Nufi2;
                              Fiss1; Fiss2; Flux1; Flux2 )
```

```
! ------------------------------------------------------------ Xe and Sm data !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &XenSam)
! --------------------------------------------------------------------------- !
! fimac is found from a 1-group macro for the individual pins !
! Xe and Sm yields are cumulative, including all predecessors !

fx       = SEL(fx    / mac / C1;Maps/ $calp)
fi       = SEL(fi    / mac / C1;Maps/ $calp)
fimac    = FOR(@smG(fx*fi)/@smG(fx))

nxe      = SEL(nd    / mic / C1;XeSm/ $calp/ 54635)
nsm      = SEL(nd    / mic / C1;XeSm/ $calp/ 62649)
axe      = SEL(ab    / mic / C1;XeSm/ $calp/ 54635)
asm      = SEL(ab    / mic / C1;XeSm/ $calp/ 62649)
fx       = SEL(fx    / mic / C1;XeSm/ $calp)
vo       = SEL(vo    / mic / C1;XeSm/ $calp)

avfxfi   = FOR(@smR(fx*fimac*vo)/@smR(vo))
avfxaxe  = FOR(@smR(fx*axe*vo)  /@smR(vo))
avfxasm  = FOR(@smR(fx*asm*vo)  /@smR(vo))
'N(Xe)'  = FOR(@smR(nxe*vo)     /@smR(vo))
'N(Sm)'  = FOR(@smR(nsm*vo)     /@smR(vo))
'Yield(Xe)'  = FOR(1E24*'N(Xe)'* (avfxaxe*1E-24 +2.1E-5) /avfxfi)
'Yield(Sm)'  = FOR(     'N(Sm)'*  avfxasm                /avfxfi)

XeSm     = LIST(Burnup;S/
                'Average nr densities and fission yields for Xe-135 and Sm-149';
                'Number densities (N) in [at/barn-cm]';
                'Effective yield of Xe-135 evaluated with lambda=2.1E-5 [sec-1]'
                 /E/
                 e5:'N(Xe)'; 'N(Sm)';'Yield(Xe)'; 'Yield(Sm)' )


! ------------------------------------- Max burnup, power and power density !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &MaxBuPow)
! --------------------------------------------------------------------------- !
bu       = SEL(bu    / mac / C1;Maps    / $calp)
rho      = SEL(rho   / mic / C1;FuelIsos/ $calp)

ip       = IND(bu / PER:RIEGO)
i1       = IND(bu / RAN:1;;;;)
is       = IND(bu / SOR:)
b1       = FOR(bu^is)
e1       = FOR(@nbE(bu)^is)
r1       = FOR(@nbR(bu)^is)
ia       = IND(e1    / SOR:ascen)
'Max burn' = FOR(((b1^ia)^^ip)^i1)
'at pin'   = FOR(((r1^ia)^^ip)^i1)

ip       = IND(power / PER:RIEGO)
i1       = IND(power / RAN:1;;;;)
is       = IND(power / SOR:)
p1       = FOR(power^is)
e1       = FOR(@nbE(power)^is)
r1       = FOR(@nbR(power)^is)
ia       = IND(e1    / SOR:ascen)
'Max W/cm' = FOR(((p1^ia)^^ip)^i1)
' at pin'  = FOR(((r1^ia)^^ip)^i1)

powdens  = FOR(power/(rho*vo+1E-20)) ! Avoid division by zero !
ip       = IND(powdens/ PER:RIEGO)
is       = IND(powdens/ SOR:)
p1       = FOR(powdens^is)
e1       = FOR(@nbE(powdens)^is)
r1       = FOR(@nbR(powdens)^is)
ia       = IND(e1    / SOR:ascen)
'Max W/gU' = FOR(((p1^ia)^^ip)^i1)
'  at pin' = FOR(((r1^ia)^^ip)^i1)
```

```
MaxVals  = LIST(Burnup;S/
               'Maximum burnup, power density [W/gU] and linear power [W/cm]';
               'Power peaking factor (ppf) without gamma smearing'
                /E/
                f0:'Max burn';    'at pin'  ; f3:'Max W/gU'; f0:'  at pin';
                f3:'Max W/cm'; f0:' at pin' ; f4: ppf                     )


! ------------------------- Delayed neutron fractions and neutron velocities !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &DelayNeut)
! ------------------------------------------------------------------------- !
B1macro  = BONE(/C1:XSset)
sig1v    = SEL(ab    / mic / C1;'1/v'   / $calp)
be       = SEL(be    / mac /   ;B1macro / $calp)

'1/v'    = FOR(sig1v/ 220000)
i1       = IND(be /RAN:;;;1;)
i2       = IND(be /RAN:;;;2;)
i3       = IND(be /RAN:;;;3;)
i4       = IND(be /RAN:;;;4;)
i5       = IND(be /RAN:;;;5;)
i6       = IND(be /RAN:;;;6;)
'Beta(1)'  = FOR(be^i1)
'Beta(2)'  = FOR(be^i2)
'Beta(3)'  = FOR(be^i3)
'Beta(4)'  = FOR(be^i4)
'Beta(5)'  = FOR(be^i5)
'Beta(6)'  = FOR(be^i6)
'Beta-tot' = FOR(@smG(be))

DelNeut  = LIST(Burnup/
               'Effective delayed neutr fractions & inverse neutr velocities';
               'Beta(i):  delayed neutr fraction in delayed neutr group "i"';
               'Beta-tot: integrated value over the 6 groups';
               '1/v:      inverse neutron velocities in sec/cm'
                /E/
                e5:'Beta(1)'; 'Beta(2)'; 'Beta(3)'; 'Beta(4)';
                   'Beta(5)'; 'Beta(6)'; 'Beta-tot';  '1/v'  )


! --------------------------------------------------- Average fuel composition !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &AvFuel )
! ------------------------------------------------------------------------- !
$hiso    = PAR(92235; 92236; 92238; 94238; 94239; 94240; 94241; 94242)
$avo     = PAR(0.6023)
ndens    = SEL(nd    / mic / C1;FuelIsos/ $calp/ $hiso)
aw       = SEL(aw    / mic / C1;FuelIsos/ $calp/ $hiso)
rho      = SEL(rho   / mic / C1;FuelIsos/ $calp)
vol      = SEL(vo    / mic / C1;FuelIsos)

nd       = FOR(@smR(ndens*vol)/@smR(vol))
'g/cm'   = FOR(@smR(vol)*nd*aw/$avo)
'wt%'    = FOR(100 *'g/cm'/@smR(rho*vol))

iu5      = IND(nd /RAN:;1;;;)
iu6      = IND(nd /RAN:;2;;;)
iu8      = IND(nd /RAN:;3;;;)
ip8      = IND(nd /RAN:;4;;;)
ip9      = IND(nd /RAN:;5;;;)
ip0      = IND(nd /RAN:;6;;;)
ip1      = IND(nd /RAN:;7;;;)
ip2      = IND(nd /RAN:;8;;;)

U235     = FOR(nd^iu5)
U236     = FOR(nd^iu6)
U238     = FOR(nd^iu8)
Pu238    = FOR(nd^ip8)
Pu239    = FOR(nd^ip9)
```

```
Pu240    = FOR(nd^ip0)
Pu241    = FOR(nd^ip1)
Pu242    = FOR(nd^ip2)

InvNd    = LIST(Burnup;S/
               'Average fuel composition in at/barn-cm'
                /E/
                e5:U235; U236; U238; Pu238; Pu239; Pu240; Pu241; Pu242)

U235     = FOR('g/cm'^iu5)
U236     = FOR('g/cm'^iu6)
U238     = FOR('g/cm'^iu8)
Pu238    = FOR('g/cm'^ip8)
Pu239    = FOR('g/cm'^ip9)
Pu240    = FOR('g/cm'^ip0)
Pu241    = FOR('g/cm'^ip1)
Pu242    = FOR('g/cm'^ip2)

InvG     = LIST(Burnup;S/
               'Average fuel composition in g/cm'
                /E/
                f4:U235; U236; U238; Pu238; Pu239; Pu240; Pu241; Pu242)

U235     = FOR('wt%'^iu5)
U236     = FOR('wt%'^iu6)
U238     = FOR('wt%'^iu8)
Pu238    = FOR('wt%'^ip8)
Pu239    = FOR('wt%'^ip9)
Pu240    = FOR('wt%'^ip0)
Pu241    = FOR('wt%'^ip1)
Pu242    = FOR('wt%'^ip2)

InvRel   = LIST(Burnup;S/
               'Average fuel composition in weight % of initial heavy mass'
                /E/
                f4:U235; U236; U238; Pu238; Pu239; Pu240; Pu241; Pu242)


! --------------------------------------------------- Burnup and Power arrays !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &BuPow)
! ------------------------------------------------------------------------- !
burnup   = SEL(bu    / mac / C1;Maps/ $calp)
cxfuel   = SEL(vx    / mac / C1;Maps)
cyfuel   = SEL(vy    / mac / C1;Maps)
nopins   = SEL(nra   / mac / C1;Maps)

avpow    = FOR(@smR(power)/nopins)
'power peaking' = FOR(power/avpow)
ppf             = FOR(@mxR('power peaking'))


! ------------------------------------------------------- Gamma-smeared power !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &GammaSmear)
! ------------------------------------------------------------------------- !
Bugam    = SEL(uburn / mac / C1;EgAll / $calpg)
TotGamPow= SEL(ed    / mac / C1;EgAll / $calpg)
pingam   = SEL(ed    / mac / C1;EgFuel/ $calpg)
fxaver   = SEL(fx    / mac / C1;XSset / $calpg)
kfaver   = SEL(kf    / mac / C1;XSset / $calpg)
fxpin    = SEL(fx    / mac / C1;Maps  / $calpg)
kfpin    = SEL(kf    / mac / C1;Maps  / $calpg)
cxfuel   = SEL(vx    / mac / C1;Maps)
cyfuel   = SEL(vy    / mac / C1;Maps)
nopins   = SEL(nra   / mac / C1;Maps)

Factor   = FOR(1-TotGamPow/@smG(fxaver*kfaver))
'Gamma smeared pow' = FOR((fxpin*kfpin*Factor+pingam)*vo)
```

```
avpow    = FOR(@smR('Gamma smeared pow')/nopins)
'Gamma smeared pow peak' = FOR('Gamma smeared pow'/avpow)
'Gamma smeared ppf'      = FOR(@mxR('Gamma smeared pow peak'))


! ------------------------------------------------------------ Fuel isotopics !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &PinFuelIso)
! ---------------------------------------------------------------------------- !
cx       = SEL(vx    / mic / C1;FuelIsos)
cy       = SEL(vy    / mic / C1;FuelIsos)
Volumes  = SEL(vo    / mic / C1;FuelIsos)
Isotopes = SEL(id    / mic / C1;FuelIsos//$Isos)
'N(fuelpin)' = SEL(nd/ mic / C1;FuelIsos/ $calp / $Isos)

PinMap   = MAP(;24 /$mapsize /
              'Output for fuel isotopics per pin';
              'Map showing the fuel pin numbers and volumes'
               / cx; cy/ Volumes)

IsoList  = LIST(;S/
                'Fuel isotopics per pin';
                'List of isotope identifiers'  /I/
                 f0:Isotopes)

NdList   = LIST(Burnup /
               'Fuel isotopics per pin';
               'Number densities in atoms/barn-cm per fuel pin and isotope';
               'R-I-E-.-. indicates fuel pin R and isotope I'            /E/
               'N(fuelpin)')


! ------------------------------------------------- Fast fluence in box & clad !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &FastFluence)
! ---------------------------------------------------------------------------- !
$ecut    = PAR(1.0E6)
tsec     = SEL(ptime / mac / C1;FlBox / $calp)
fxbox    = SEL(fx    / mac / C1;FlBox / $calp)
fxclad   = SEL(fx    / mac / C1;FlClad/ $calp)
eh       = SEL(heg   / mac / C1;FlClad)
el       = SEL(leg   / mac / C1;FlClad)

f1       = FOR(@ln(eh/$ecut)/@ln(eh/(el+1E-20)))
i1       = IND(f1   /WIN:0.0,/ 0.0)
f2       = FOR((f1^i1)^^i1)
i2       = IND(f2   /WIN:-1.0,1.0/ 1.0)
f        = FOR((f2^i2)^^i2)
Box      = FOR(@spE(@smG(fxbox *f*tsec)))
Clad     = FOR(@spE(@smG(fxclad*f*tsec)))

Fluence  = LIST(Burnup;S/
               'Fast fluence (>1 Mev) in channel box and clad'
                /E/
                Box; Clad)


! -------------------------------------------------- Neutron detector response !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &NeutronDetect)
! ---------------------------------------------------------------------------- !
fxdet    = SEL(fx    / mic / C1;DetNeu/ $calp)
fidet    = SEL(fi    / mic / C1;DetNeu/ $calp /92235)

NeuDet   = FOR(fxdet*fidet*1E-24)

NeutronD = LIST(Burnup;S/
               'Neutron detector response factors:';
               'NeuDet     : absolute fission rate in fissions/sec'
                /E/
                NeuDet)
```

```
! --------------------------------------------------- Gamma detector response !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &GammaDetect)
! --------------------------------------------------------------------------- !
DetResp  = ARRAY(G: 1.915; 1.862; 2.028; 2.870; 5.284)
Burgam   = SEL(uburn / mac / C1;DetGam/ $calpg)
fxdet    = SEL(fx    / mac / C1;DetGam/ $calpg)

GamDet   = FOR(@smG(fxdet*DetResp))

GammaD   = LIST(Burgam;S/
               'Gamma detector response factors:';
               'GamDet    : absolute ionization rate in ion-pairs/sec'
                /E/
                GamDet)


! --------------------------------------------------- Gamma detector response !
&ADD     = SET('AR931A-01.HRF' / SAMPLE; ZENITH; &DiscFactors)
! --------------------------------------------------------------------------- !
fx       = SEL(fx    / mac / C1;XSset    / $calp)
jpcor    = SEL(jp    / cur  / C1;CurCorner/ $calp)
jmcor    = SEL(jm    / cur  / C1;CurCorner/ $calp)
jpsid    = SEL(jp    / cur  / C1;CurSide  / $calp)
jmsid    = SEL(jm    / cur  / C1;CurSide  / $calp)

Corners  = FOR(2*(jpcor+jmcor)/fx)
Sides    = FOR(2*(jpsid+jmsid)/fx)

DisFact  = LIST(Burnup;;5/
               'Flux discontinuity factors';
               'f-.-E-g-. denotes face "f" and group "g"';
               'f=1 for NW corner and N side';
               'f=2 for NE corner and W side';
               'f=3 for SW corner and E side';
               'f=4 for SE corner and S side'
                /E/
                Corners ;; Sides)

! --------------------------------------------------------------------------- !
! <<<<<<<<<<<<<<<<<<  E n d   o f   Z e n i t h   s e t s  >>>>>>>>>>>>>>>>>> !
! --------------------------------------------------------------------------- !
```

Studsvik® Scandpower

ORION

USER'S GUIDE

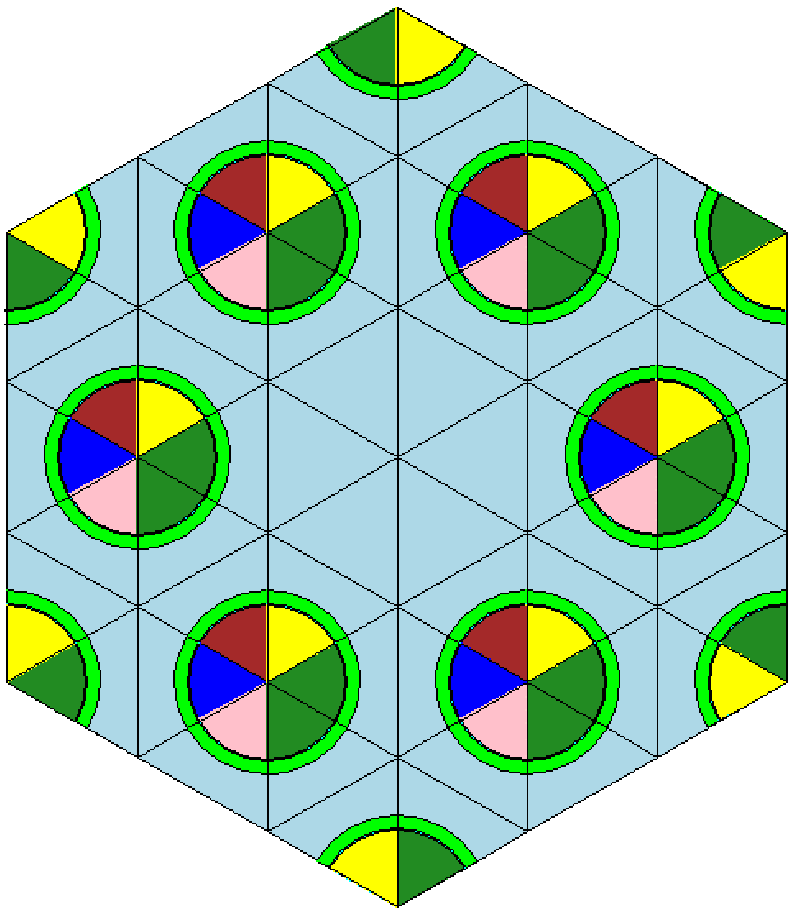


TABLE OF CONTENTS

1 INTRODUCTION

ORION is a display tool for viewing the geometry and the material properties of the system which is about to be calculated by HELIOS.

ORION is very useful in the verification as well as in the built up process of the geometry, material and temperature overlays assignments.

ORION is greatly efficient and helpful for documentation purposes of HELIOS input data.

The ORION program consists of two parts:

- A "orion code" written in Fortran 90 which extracts the necessary data from the HERMES file and saves those data in temporary ASCII files.
- A "orion script" is a script written for TCL/TK interactive tool which uses the output from the "orion code" to visualise the requested parts of Helios input.

This User's guide covers the Unix and Windows versions of ORION.

The pictures used in this manual are produced from a Hermes file "helios-24.hrf" which is a part of ORION sample cases.

2 INSTALLATION NOTES

The installation of ORION consists of two steps:

1. Copying all files and directories from the distribution CD to an installation area. The installation path has to be defined as variable called SCM.(see next point)

2. Define environment variables
   - Unix. (shell dependent!)

     ```
     >setenv SCM installation_path
     >setenv SYS operating_system
     ```

     where:
     operating_system must be one of the following: osf1, hpux, aix, sun5 or linux

   - Windows

     ```
     >set SCM=installation_path
     >set SYS=winnt
     ```

The definition of SCM and SYS is important for proper running of ORION.

3 GETTING STARTED

ORION starts by execution of a start-up script placed in:

Unix:
`$SCM/orion-x.x/tclsource/start_orion`

Windows:
`%SCM%\orion-x.x\tclsource\start_orion.bat`

where:
x.x is the version of ORION fex 1.8.

NB! *The user must have write permissions in the current working directory.*

ORION writes scratch files locally in the current working directory. The source of ORION and the HERMES file may lay anywhere provided the user has execute and read permissions respectively.

4 MAIN WINDOW

When ORION is started properly ORION's main window appears on the screen.

The ORION main window is shown in Figure 4.1.

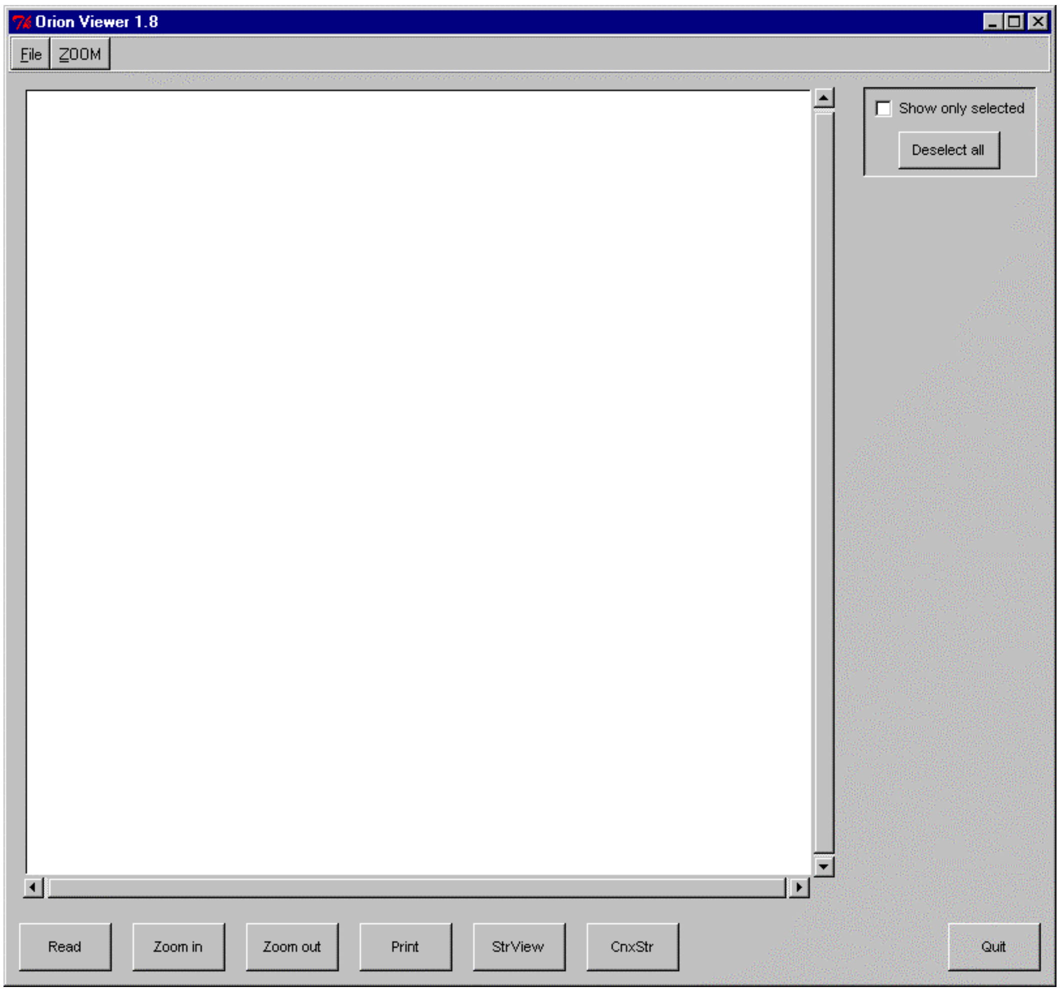


Figure 4.1. ORION main window.

The Orion main window consists of the following logical parts:

- The <u>canvas</u> is where the lattice is displayed.
- The attached <u>scrollbars</u> will change the view of the lattice.
- The lower <u>button section</u> and <u>menu bar</u> control the basic functions.
- The <u>palette</u> shows the correspondence between materials and colours, and offers options for selecting which materials to be displayed.

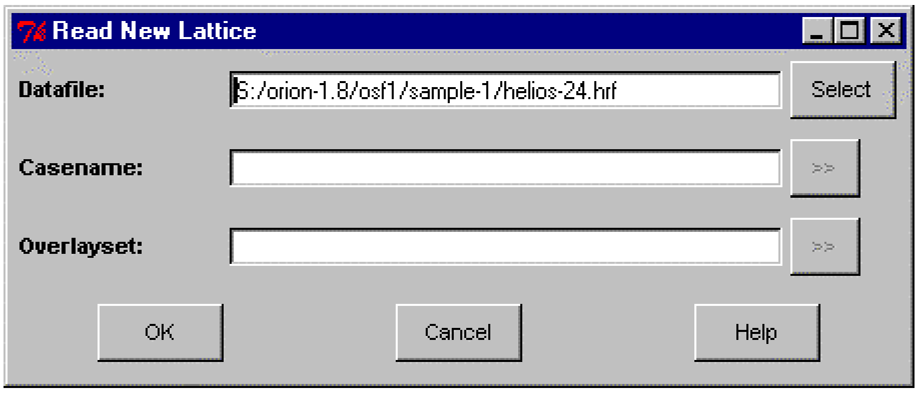
Remark:
It may happened that the main window does not fit into the size of the current display. Refer to chapter "Orion settings and Customisation" of this manual for details of how to resize the ORION's main window.

5 READING INPUT

The steps of supplying the input are followed up in this chapter.

To give the input parameters, click the "Read" button. (An alternative is to use the menu bar "File" and then "Open". A hot key "Ctrl-o" may also be used.)

The dialog box shown in Figure 5.1 will appear:


Read New Lattice

Datafile: S:/orion-1.8/osf1/sample-1/helios-24.hrf

Casename:

Overlayset:

Figure 5.1. Dialog box for entering parameters of a new file.

5.1 Enter a Hermes Data File

The user must supply the path and the name of the Hermes file either by direct typing in the field "Datafile" or by clicking on "Select" button and browsing the file-system to select the file name.

5.2 Case name

The case name is optional.

If the field "Casename:" is left empty ORION will find and use the first HELIOS case saved in the HERMES file.

If the field "Casename" is not empty the CASE which name is written there must exist in the HERMES file. Otherwise an error message will appear suggesting missing or wrong CASE name.

5.3 Overlay set name

The overlay set name is optional.

If the field "Overlayset:" is left empty ORION will find and use the first material overlay saved in the HERMES file.

If the field "Overlayset" is not empty the Overlay set, either material or temperature, which name is written there must exist in the HERMES file. Otherwise an error message will appear suggesting missing or wrong name.

5.4 Initialise the system

After supplying at least the name of the Hermes file, by a click on the button "OK" in Figure 5.1 ORION will initialise the system:

The result is shown in Figure 5.4

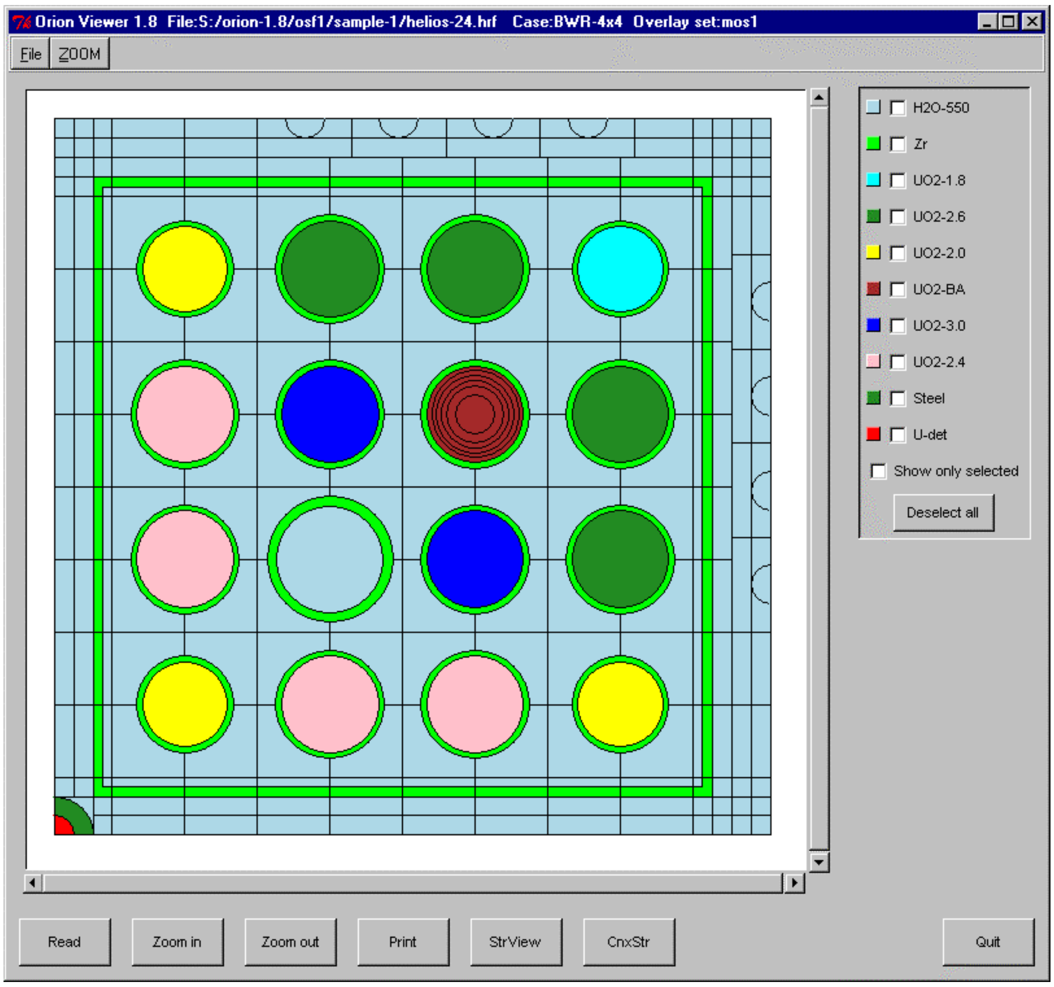


Figure 5.4 Main window

On the title bar ORION notifies which file is used and which CASE and OVERLAY are activated.

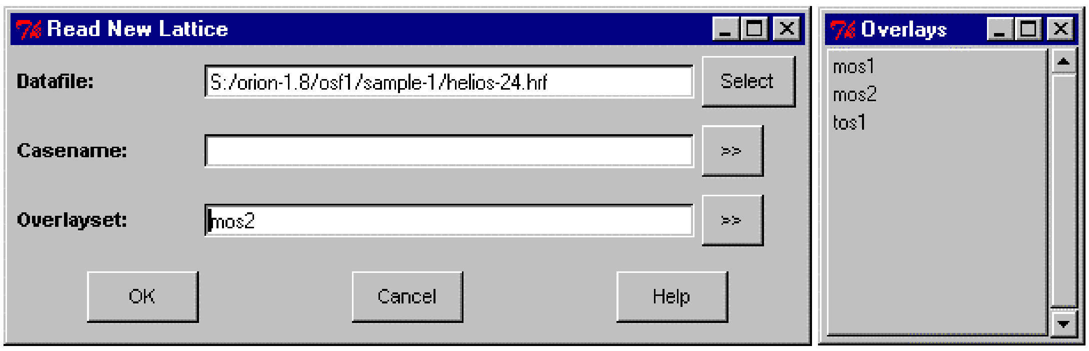
5.5 Selecting CASE and OVERLAY from a list

Once the system is initialised the user can select from the list of Helios cases and Overlays.

By click on the "Read" button the dialog box from Figure 5.1 will reappear.

As the system has been initialised the arrows left from the fields "Casename:" and "Overlayset:" are activated.

The result from a click on the arrow left from "Overlayset" is shown on Figure 5.5.



Read New Lattice

Datafile: S:/orion-1.8/osf1/sample-1/helios-24.hrf

Casename:

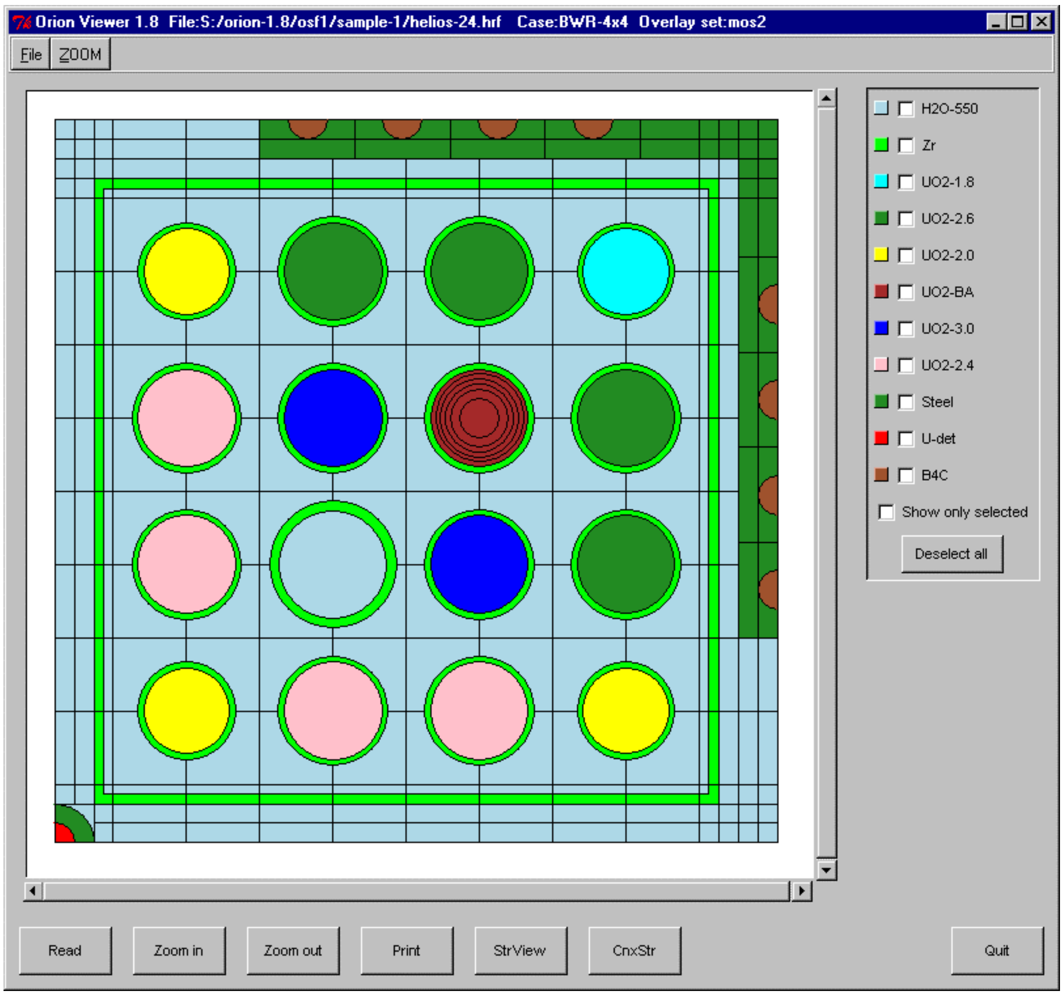
Overlayset: mos2

Overlays

mos1
mos2
tos1

Figure 5.5.1 Selection of an Overlay set.

The selection is done by a "double click" on the selected overlay name.
The system filled with the materials of the overlay set "mos2" is shown on Figure 5.5.2



Orion Viewer 1.8 File:S:/orion-1.8/osf1/sample-1/helios-24.hrf Case:BWR-4x4 Overlay set:mos2

Figure 5.5.2 Case: BWR-4x4, Overlay set: mos2

ORION can select between material and temperature overlay sets.
In the example of Figure 5.5.1 the overlay set called "tos1" is a temperature overlay set. Filling the system with it will result as shown in Figure 5.5.3:

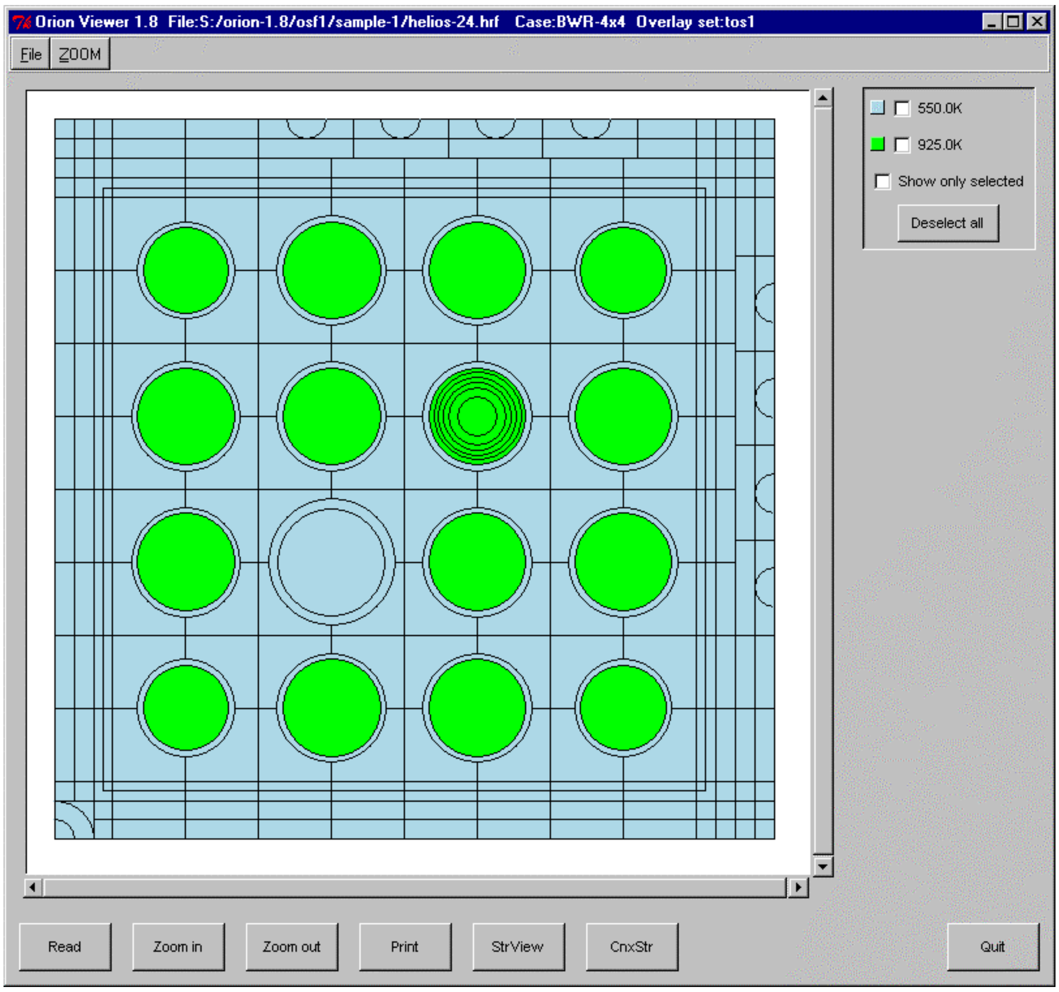

Orion Viewer 1.8 File:S:/orion-1.8/osf1/sample-1/helios-24.hrf Case:BWR-4x4 Overlay set:tos1
File
ZOOM
550.0K
925.0K
Show only selected
Deselect all
Read
Zoom in
Zoom out
Print
StrView
CnxStr
Quit

Figure 5.5.3. Temperature overlay view.

Remark:

If more than one HELIOS CASE exists in the HERMES file, the selection must be done on three steps:

1. First select the CASE name.
2. Initialise the system.
3. Select from the Overlays which correspond to the selected CASE name

5.6 Undefined material

If a part of the system is without "filling" material/temperature, ORION will give a message notifying that fact. The corresponding parts of the system will be filled with a separate colour with corresponding name "Undefined"

6 MANIPULATING THE MAIN WINDOW

Some manipulations that can be done on the main window are described in this chapter.

6.1 "Zoom in" and "Zoom out"

Zooming in and out can be done by clicking the corresponding buttons.
The replacing "hotkeys" are "PageUp" and "PageDown".

6.2 Scroll the window

Scrolling can be done by dragging the scroll-bars with mouse or using of arrow keys of the keyboard. The gravity centre of the initial picture is preserved during the scrolling.

A picture result of consecutive zooming in and scrolling to the South-West corner is shown in the Figure 6.2

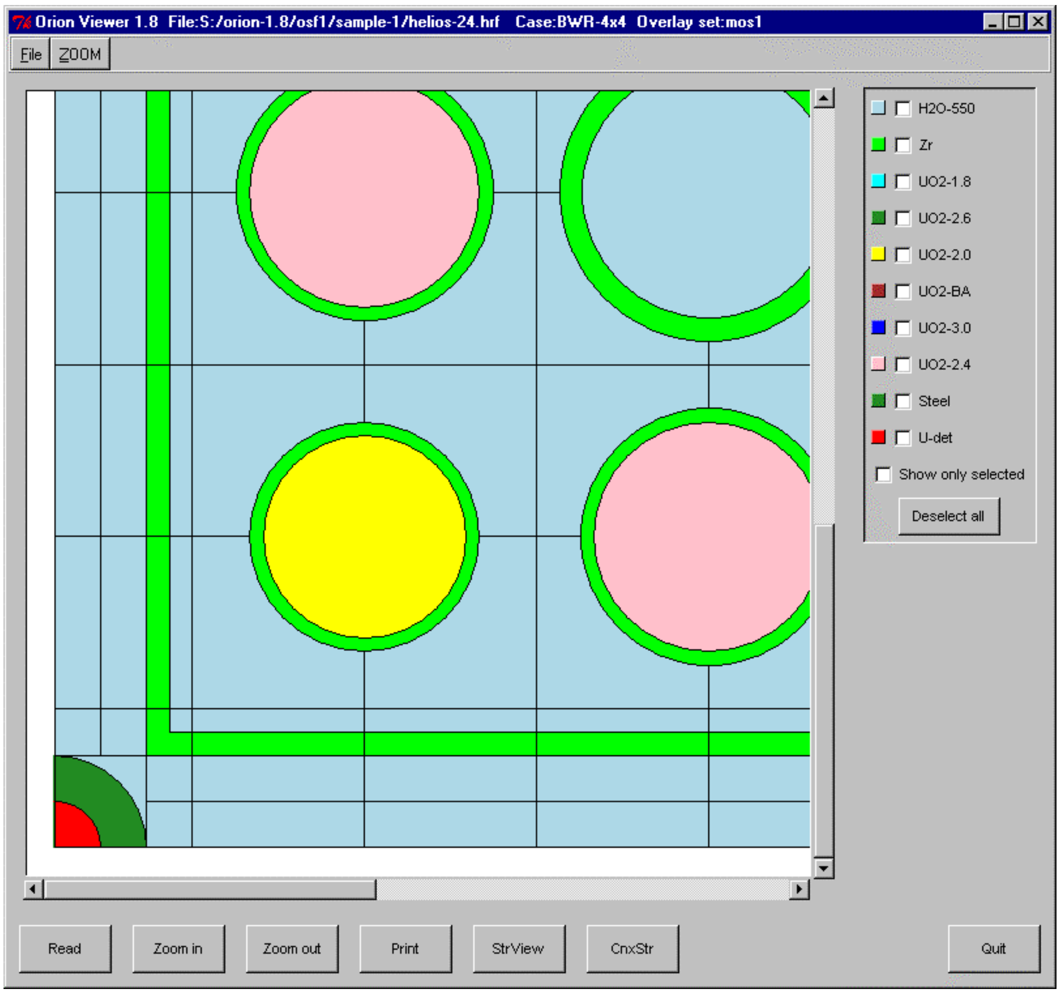


Figure 6.2 ZOOM and SCROLL

6.3 Select and deselect materials/temperatures.

Individual materials/temperatures can be selected by the help of the functionalities of the pellet.

A click in the check-box "Show only selected." will result in removing all colours from the system. Then selecting only the check-box of the materials, wished to show up the result will be as shown in the Figure 6.3.

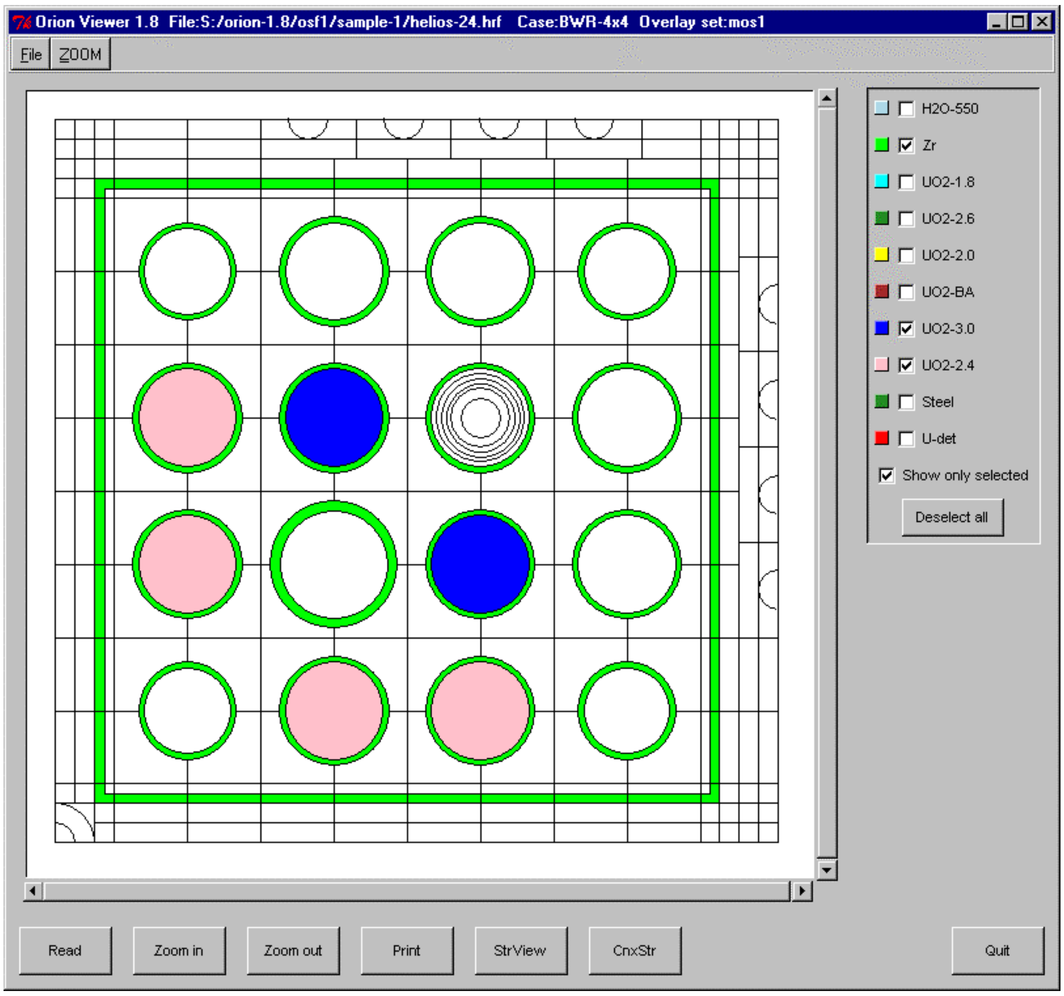
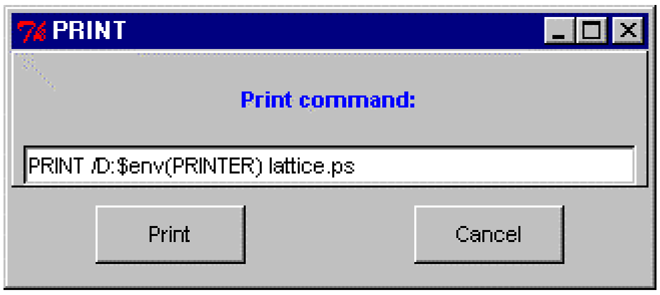


Figure 6.3 Select and deselect region's properties.

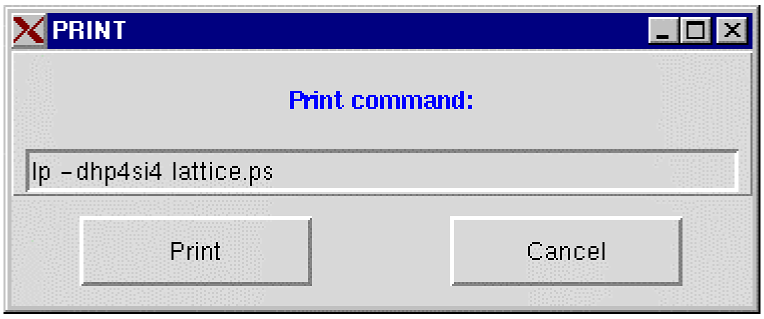
6.4 PRINT the main window

The picture of the main window will be printed to the connected printer port by click on the button "Print" or by pressing "Ctrl-p" on the keyboard.
ORION expects that a variable PRINTER is defined in the user environment.
A temporary PostScript file will be created in the current working directory and send to the printer.
Alternatively the user can supply the print command as shown in the examples of Figures 6.4.1 and 6.4.2



```
PRINT /D:$env(PRINTER) lattice.ps
```

Figure 6.4.1 Windows Print command example



```
lp -dhp4si4 lattice.ps
```

Figure 6.4.2 Unix Print command example

6.5 Save the picture to a PostScript file

The picture of the main window can be saved in a post script file.

Click on the menu bar "File" and choose "Save as". (hotkey "Ctrl-s")
Then brows to the directory where the file should be saved.

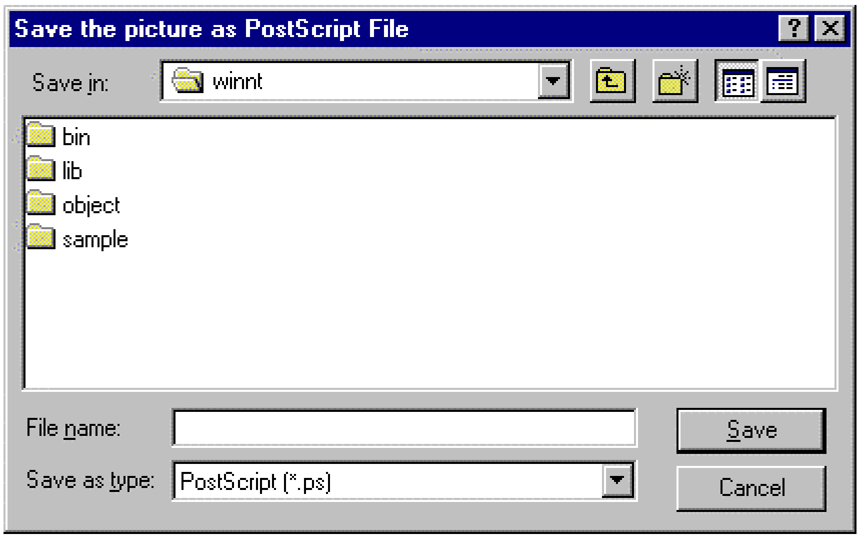


Figure 6.5. Save as a PostScript file.

7 STRUCTURE VIEW

Click on the button "StrView"(hotkey "ctrl-t") and a window titled "View Structures" will appear as shown on Figure 7.1.

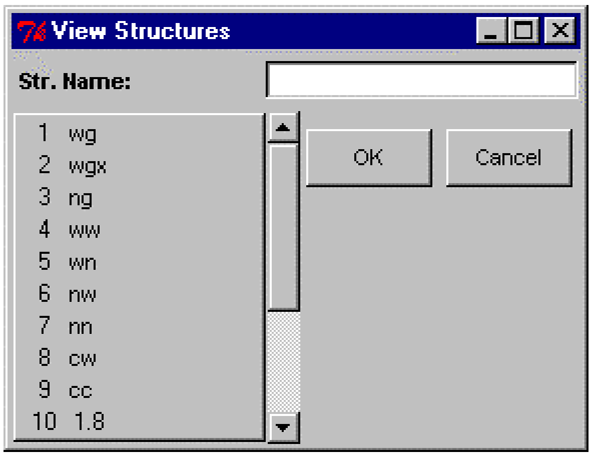


Figure 7.1. Structure view selecting window.

The list on the left side of Figure 7.1 contains all structures defined(activated) by STR operator in the exercising system.

There are three ways of getting the desired structure selected(Fugure 7.2):

- double click of the left mouse button
- selecting by "left + right" mouse click and push "OK"
- by typing the structure sequence number and push "OK"

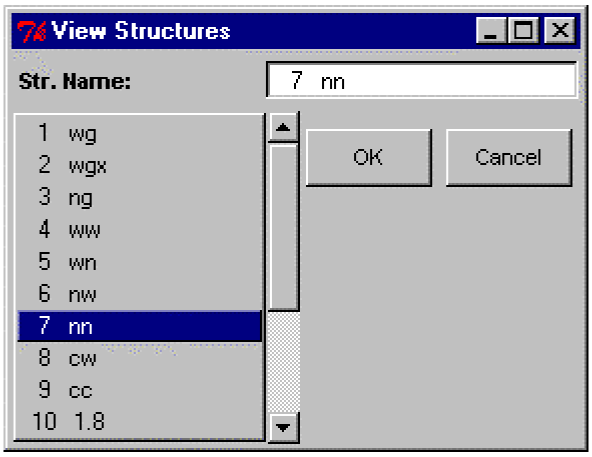


Figure 7.2. Select a structure.

Remarks:

1. Some of the structures may not be in use(connected) in the current system. However they will be listed as activated by an STR operator.
2. The number leading the structure name is the sequence number of STR operators as they appear in the input.

Once the structure is selected by means of any of the three ways mentioned above the picture on the main window will be replaced by a picture of the just selected individual structure as shown in Figure 7.3

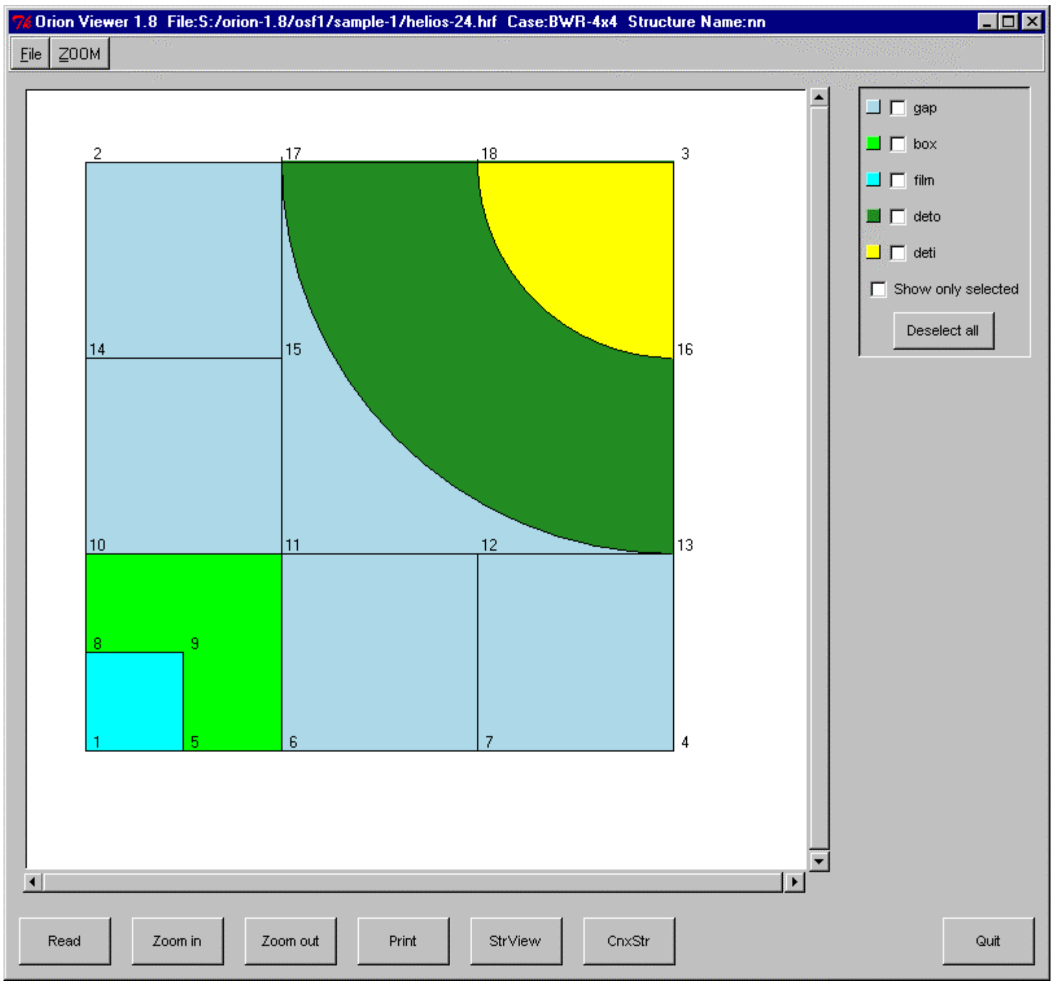


Figure 7.3 Structure view. Main window

Remarks:

1. The structure is drawn as defined in the input (local coordinate system)
2. The node numbers are placed always North-West of the exact point of corresponding node.
3. It may happened that the nodes are overlapping each other. Zooming will help to separate them.
4. The names in the pallet are the names of the regions given in the structure definition. Individual regions can be exercised by checking "Show only selected" and check on the individual region name buttons.
5. The names in the pallet are limited to 10 characters.
6. The title bar of the main window shows the file, the HELIOS case and the structure name currently displayed on the main window.

8 CONNECTIONS VIEW

The final system which is should be calculated by HELIOS consists of one or more structures, or, one or more subsystems connected together by a CNX operator.
ORION displays the connections by a click on "CnxView"(hotkey "Ctrl-x") as shown in Figure 8.1

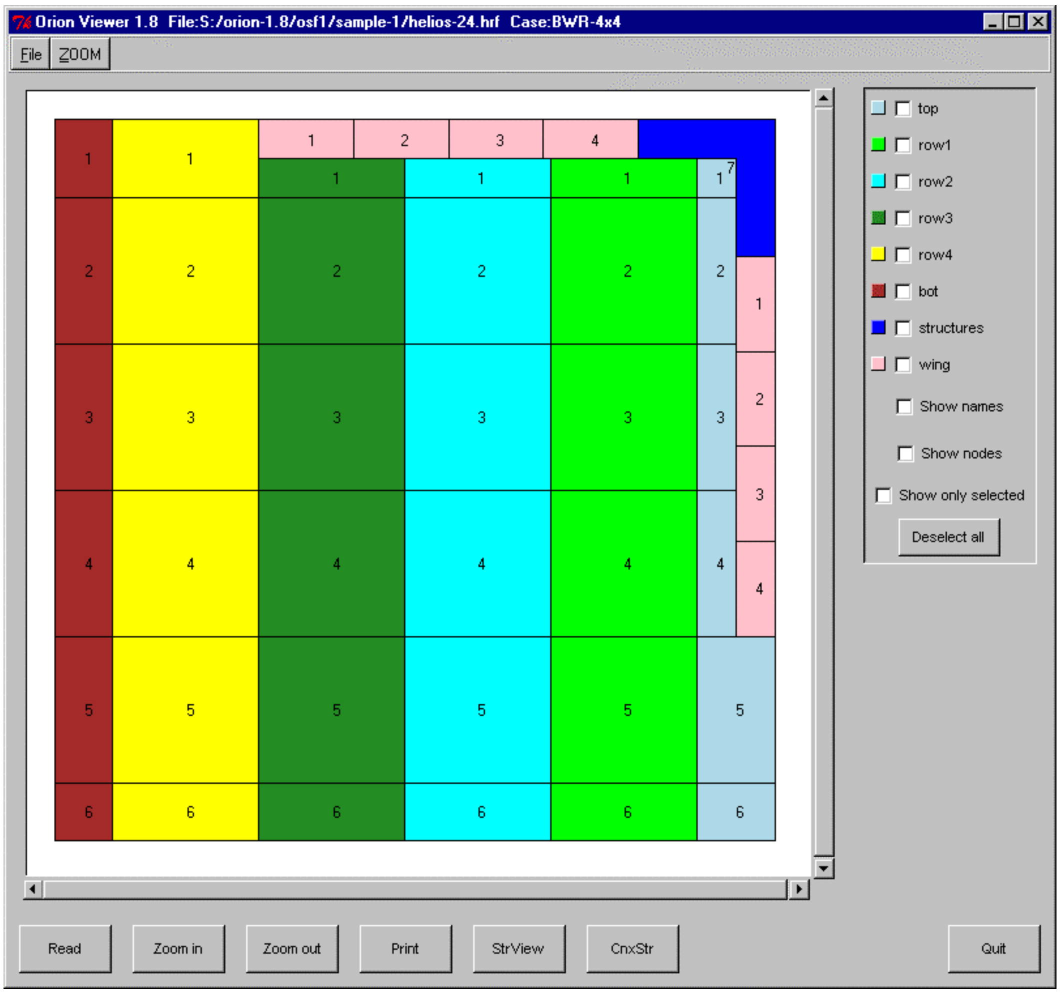


Figure 8.1 Connections view.

Remarks:

1. The pallet shows the names of the subsystems included in the final CNX
2. The subsystems view is limited to the first level. For example if the final system consists of two subsystems and each of those two subsystems consist of other two subsystems(second level deep connections) ORION will display the four subsystems. In other words ORION extracts only the first level subsystems and cannot combine them in higher level connections.
3. The names in the pallet are limited to 10 characters.
4. If individual structures are connected to the final system together with CNX defined subsystems they will be "combined" in the picture under the name "structures". See also chapter 8.1
5. One or more subsystems can be displayed by check on the "Show only selected" button and select their names from the pallet.

8.1 Structure names in the connections.

By a click on check button "Show names" ORION displays the names of the structures connected in the final system, as parts of its subsystems. Figure 8.1.1.

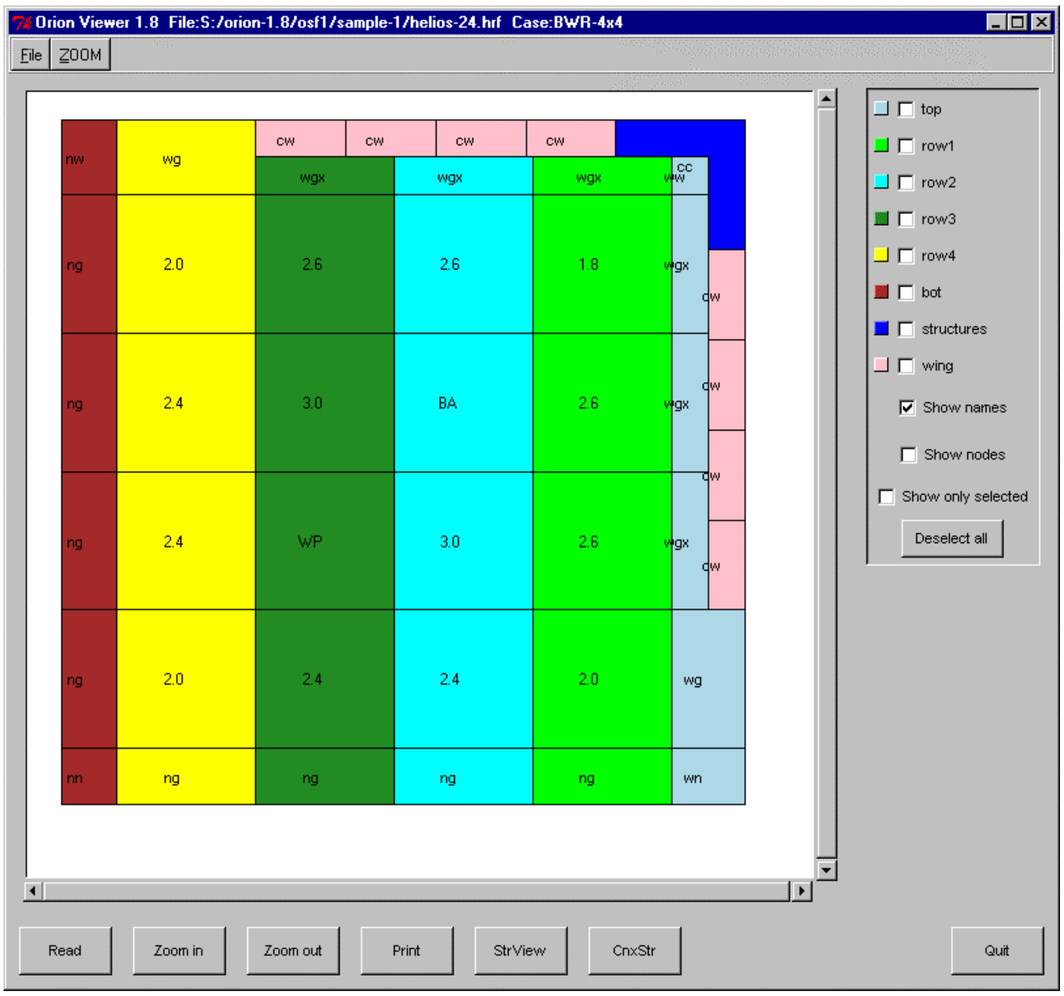


Figure 8.1.1 Structure names in the final connection

Remarks:

1. The names are placed in the gravity centre of the structure. It may happened that the gravity centre lays out of the structure drawing area.
2. The names are limited to 10 character. If a name consists of more than 10 characters only the first 10 will be displayed.
3. The pallet window keeps the names of the subsystems.

8.2 Structure connection nodes and sequence numbers

By a click on the check-box “Show nodes” ORION displays the sequence number of the structures in the CNX operator and their periphery nodes. Figure 8.2.1

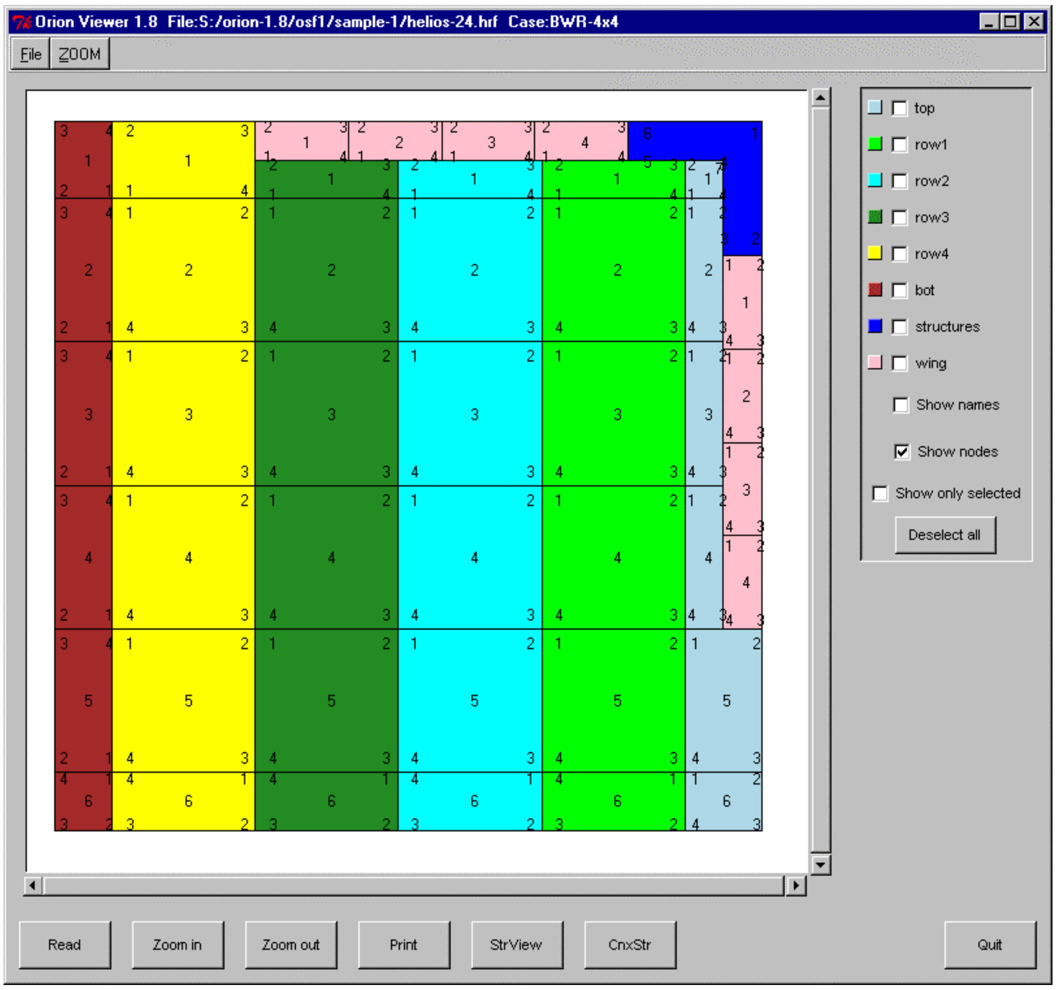


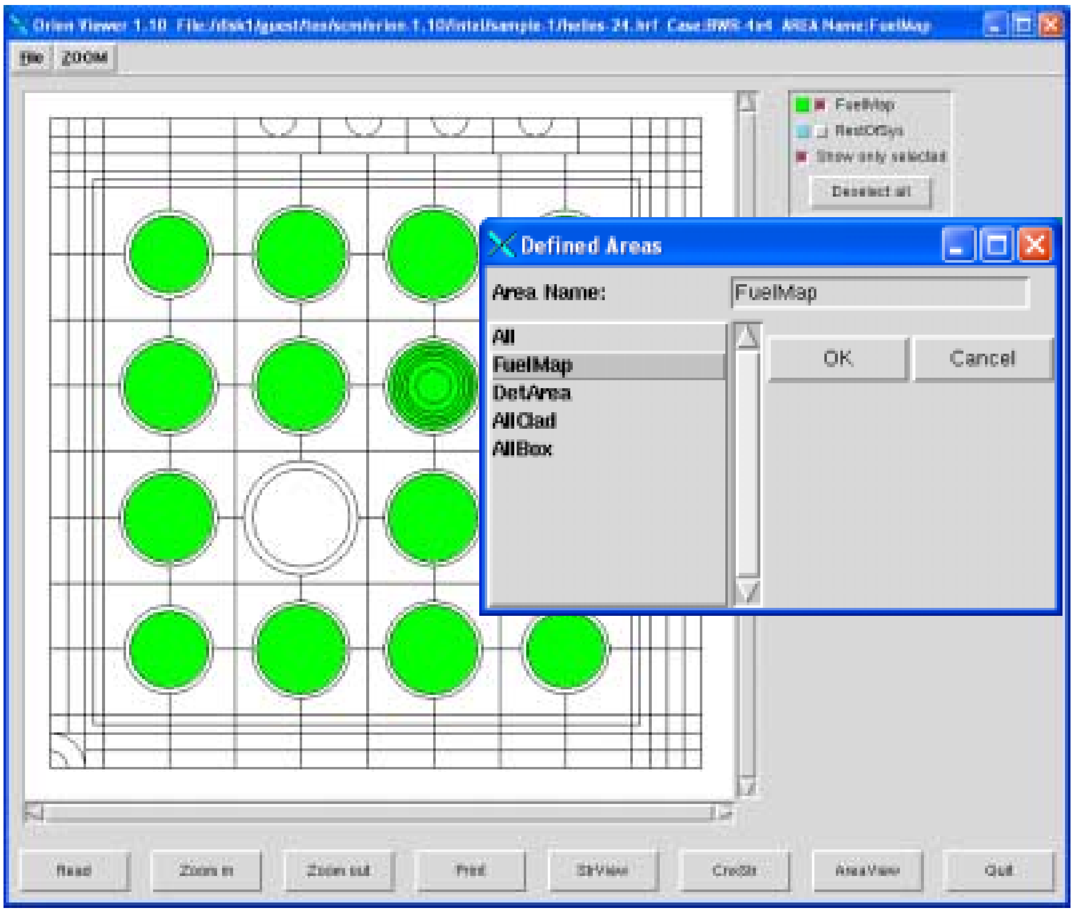
Figure 8.2.1 Nodes and sequence numbers in the connections.

Remarks:

1. ORION shows only the nodes on the periphery of each structure. They are not necessarily the ones used in the connections in a CNX operator. If other nodes have been used in the connection they are not displayed. Use Structure view to exercise all nodes of one particular structure.
2. The nodes used in the connection of subsystems are not notified.

9 AREA VIEW

AREA view provides a possibility to check and review the areas defined in the current system. By pressing “AreaView” a window with a list of defined areas shell appear as shown on Figure 9.1 below:



There are three ways of getting the desired structure selected(Fugure 9.1):

- double click of the left mouse button
- selecting by “left + right” mouse click and push “OK”
- by typing the area name and push “OK”

Remarks:

1. All regions included in an area will be shown. However ORION will not show the “homogenisation” properties of the area. For example ORION will show the same picture for the following AREAs:

```
Area1     = AREA(<  *-*- (fuel1,fuel2)>)
Area2     = AREA(   *-*-<(fuel1,fuel2)>)
```

Area1 homogenizes all regions fuel1 and fuel2 all over the system- “all fuel area”. Area2 homogenizes fuel1 and fuel2 in the structure- “fuel map area”.

10 ORION SETTINGS AND CUSTOMISATION

The settings of ORION are set up in a few files which the customer can use to fit his/hers own preferences.

/orion-x.x/tclsource/orion	- main start file and settings
/orion-x.x/tclsource/.orion_color	- colour sequence
/orion-x.x/tclsource/.orion_font	- fonts
/orion-x.x/tclsource/.orion_printer	- default print command

10.1 Set up the MAIN window canvas

The size of the MAIN window is set up through the following variables in file `orion`:

```
set adm(canv_height) 600
set adm(canv_width)  600
```

The figures in bold may be changed to fit the screen size.

10.2 Set up default fonts

ORION uses only three fonts. They can be customised in the file `.orion_font`.
Below is an example of needed variables to set up the default fonts on Unix:

```
set appOption(message_font) -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-*-*
set appOption(normal_font)  -*-helvetica-medium-r-normal-*-12-*-*-*-*-*-*-*
set appOption(bold_font)    -*-helvetica-bold-r-normal-*-12-*-*-*-*-*-*-*
set prfsz 18
```

Example on Windows:

```
set appOption(message_font) {helvetica 10 bold}
set appOption(normal_font)  {helvetica 8 normal}
set appOption(bold_font)    {helvetica 8 bold}
set prfsz 12
```

Remark: ORION uses only non scalable fonts.

10.3 Set up the colour sequence

The colours are set up in the file `.orion_color`.
Below is an example:

```
set color(0) cyan
set color(1) lightblue
set color(2) green
set color(.) cyan
set color(n) forestgreen
```

where: the maximum of "n" is defined by the variable `colorN`

Remark:

1. The colours must be given in ascending sequence

NB! The sequence of the colours corresponds to the sequence in the color pellet which on its side corresponds to the sequence of MAT operators in the AURORA input.