Flopack Internal Programs

User Documentation

Second edition
January 1999

Jean-Louis Blanchard

Flomerics Ltd. France

32 rue Jean Rostand
91893 Orsay Cedex - FRANCE
Tel. : (33) 01 69 35 30 90 - Fax : (33) 01 69 41 27 67
Email: `jlblanchard@flomerics.fr`

Table of contents

Table of figures

Table of tables

Table of listings

Typographical conventions

`Typewriter style` is used to represent things typed by the machine and *`slanted typewriter style`* to represent things typed by the user. *Italic style* is used to represent part of a command which needs to be substituted by a value. In that case, the words describing the value are separated by the underscore character (_). *Slanted style* is used to emphasize part of the text. SMALL CAPITAL letters are used for program names.

Designation of files

For convenience, the files and the data written in them are referred to by the notation ∗.*extension* where the extensions used are the following:

`lib`	Library file
`bc`	Boundary condition file
`map`	Boundary condition mapping file
`net`	Network definition file
`bcs`	Solcond internal boundary condition file
`dmo`	Detailed model output file
`res`	Network model file
`fit`	Network backfit file

Notations

n_j	Number of junction nodes
n_e	Number of external nodes
n	Total number of nodes
m_e	Number of environmental conditions
m_j	Number of power dissipation columns in the ∗.map file
m_a	Number of ambient temperature columns in the ∗.map file
m_h	Number of heat transfer coefficient columns in the ∗.map file
T_i	Temperature of node i
q_i	Flux across node i
L_{ij}	Link between nodes i and j
R_{ij}	Resistor between nodes i and j

1 Overview

The generation of a compact model according to the DELPHI methodology involves the main following steps (Fig. 1):

(P1) *Detailed model specification.* This phase requires to specify:
- the internal structure of the detailed model;
- the external walls and the planar heat source which are the support of the compact model nodes.

So, two detailed models have to be considered:
- a *detailed model for archive purpose*, to be used in a FLOTHERM simulation instead of a compact model. Such a model has no external walls;
- a *detailed model for modelling purpose*. Such a model includes external walls, which implicitely means that the nodes of the compact model have already to be known at that stage. Moreover, for the sake of computation time, symmetry considerations may lead to consider only a part of it. For instance, the provided examples for a square PQFP use a quarter model. In addition, some parts of the detailed model may be excluded from the simulation. This is for example the case of the leads for a square PQFP.

(P2) *Detailed model simulation.* A set of environmental conditions is defined to represent the various conditions in which a component may operate. For each environment of this set, the detailed model is simulated. The results obtained are processed to give the average temperature and flux for each external wall, as well as the mean junction temperature.

(P3) *Compact model network generation.* From the thermal maps obtained after the previous phase, and from a given topology of links between the nodes of the compact model, the resistors of each link are computed.

(P4) *Compact model backfit.* The compact model network is solved for each environment of the environmental condition set and the results obtained are compared with those of the detailed model in order to quantify the behaviour of the compact model.

(P5) *Compact model generation.* This phase consists of generating a representation of the compact model which can be implemented in FLOTHERM. At the moment, the procedure consists of creating an object named the *cube and plate model*. This object is made of high-conductive cuboids separated by plates to model the links between the compact model nodes.

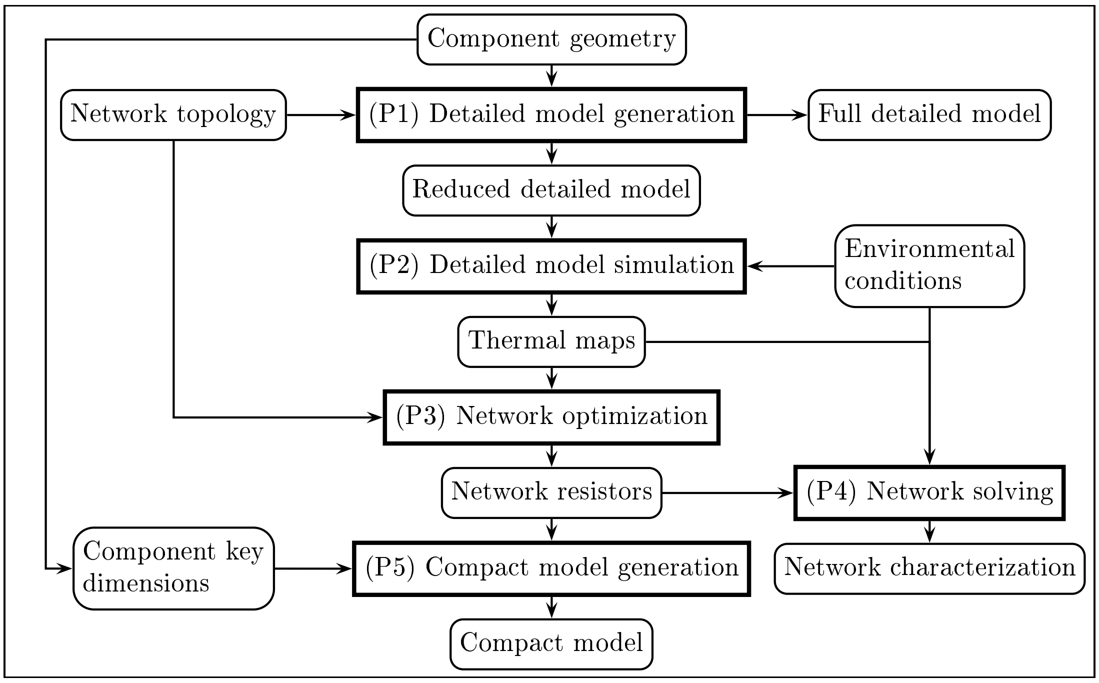

Figure 1: Compact model generation data flow

2 Scope of the Flopack internal programs

The scope of the FLOPACK internal programs named GENBCS, SOLCOND and SOLRES is to cover phases (P2), (P3) and (P4).

SOLCOND is a conduction solver which implements phase (P2) by processing FLOGATE library files in order to provide the thermal maps further processed by SOLRES, which implements phases (P3) and (P4). SOLCOND possesses specific features aimed at facilitating the simulation of detailed model of components.

First, since the FLOGATE library file format only allows to process a single set of boundary conditions, the SOLCOND parser includes a specific mechanism which enables to loop over a set of environmental conditions by updating in the appropriate class (external wall or planar heat source value) part of the data a boundary condition is made of (e.g. the heat transfer coefficient). This mechanism is implemented through a file named a **.bcs file* generated by GENBCS.

Then, as previously emphasized in the description of phase (P1), two detailed models are involved. For that purpose, SOLCOND enables to deal with two library files. The first one, named for instance `detailed.lib`, models the full structure of the component. The second one, named for instance `harness.lib`, defines the external walls for applying the environmental conditions required by the derivation of the thermal maps which make the basis of the compact model generation. The geometrical location of objects has to be consistent between the two files but the `harness.lib` file can draw benefit of the model symmetry. This means for instance that the external walls may only cover a quarter of the detailed model. To this end, SOLCOND enables to define a *clipping cuboid* typically inserted at the head of the `harness.lib` file. With the help of this clipping cuboid some parts of the detailed model can simultaneously be excluded from the simulation.

Finally, SOLCOND also provides a specific mechanism for implementing the one-to-many association between the nodes of the compact model network and the related geometrical objects. This mechanism consists of post-processing the identifiers of the objects referring to the classes `planar_source` and `external_wall` as follows:

– search for the underscore character in the object identifiers;
– compare the stems of those identifiers;
– combine the areas and thermal maps related to the objects sharing the same stem.

For instance, the results attached to two external walls named `TO_1` and `TO_2` can be combined since their identifiers share the same stem `TO`. This mechanism leads to adopt the following rules in the preparation of the library files.

Practical rules for the preparation of the library files. According to the SOLCOND specific features, use the following conventions for naming classes and objects in the `harness.lib` file:

– if the junction node name is `J`, name the `value_psource` class and the planar source object `J_1`; give to the `planar_source` class any name which cannot conflict with `J_1`, like `det_junction`;
– if an external node name is `E` and if two external walls are associated with it, name the corresponding classes and objects `E_1` and `E_2`;
– follow the same order for defining the external nodes in the *.map file, like `TI`, `TO`, `BI`... and the related external wall objects in the `harness.lib` file, like `TI_1`, `TO_1`, `TO_2`, `BI_1`...

It is also worth noting that the default name of the clipping cuboid is `domain`.

Some of these rules may be relaxed. For instance, the underscore character in the previous examples is actually customizable. Similarly, a prefix string can be added at the head of the class names. The clipping cuboid name is customizable as well.

3 Input data file specification

Fig. 2 gives an overview of the file layout. All files are of ASCII type. Those tagged with a plus sign use a fixed format to ease creation and interoperability while those tagged with an asterisk are parsed. Parsed files are computer-generated. The underlined files are those the end user needs to create by hand. Fixed-format files start with a single comment line having a sharp sign (`#`) in the first column. Other lines have data separated by commas.

This section covers the files which have to be created by user, namely the boundary condition mapping file (*.map) and the network definition file (*.net). The boundary condition file (*.bc) can be created or modified by the user, but a default file is provided.

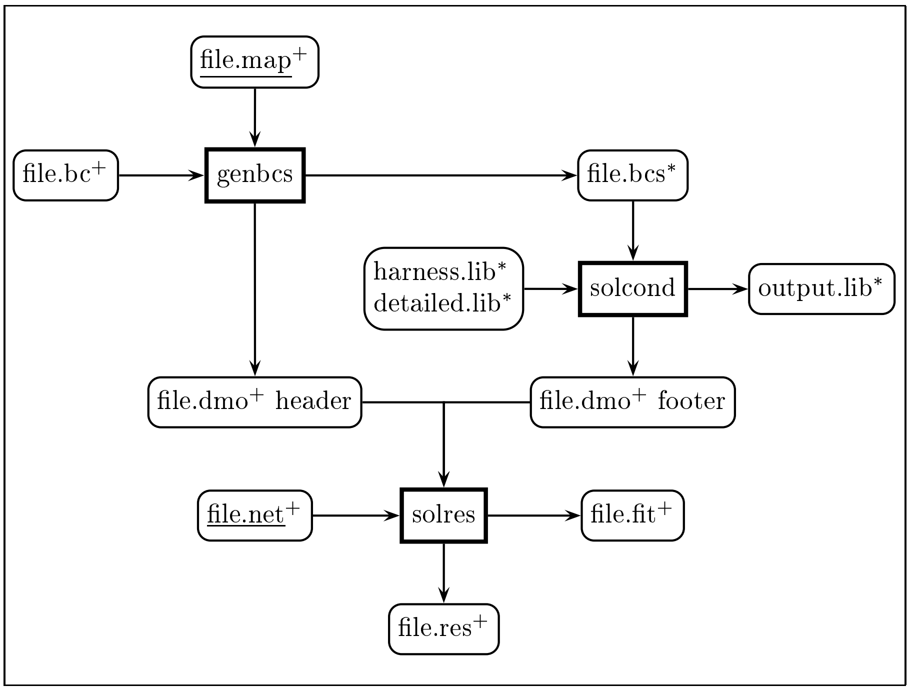

Figure 2: File interoperability between Flopack internal programs

3.1 Boundary condition file (∗.bc)

This file contains the boundary condition information as an array, one column per surface (top, bottom, sides, leads) and one row per environmental condition. The order of columns is irrelevant, as the assignment of the values to particular nodes (external walls) is defined in the ∗.map file. This file being species-independent, a default file named `stdset.bc` is provided.

The file structure is the following:

1 (line)	# *Comment_line*	
2 (line)	*Number_of_power_dissipation_values,*	Initially $n_j = 1$
	number_of_ambient_temperature_values,	Initially $m_a = 1$
	number_of_htc_columns,	m_h
	number_of_environments_in_set	m_e
3 (line)	*Power_dissipation_values*	Initially one value
4 (line)	*Ambient_temperature_values*	Initially one value
5 (block of m_e rows)	Htc_1, htc_2, …, htc_{m_h}	One row per environment

An exceirpt of the default file `stdset.bc` is given in Listing 1.

```
# Definition of the Philips set of 38 environmental conditions
1, 1, 4, 38
1.000000
0.000000
1.000e+04, 1.000e+04, 1.000e+04, 1.000e+04
1.000e+03, 1.000e+03, 1.000e+03, 1.000e+03

1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02
1.000e+02, 1.000e+02, 1.000e+02, 5.000e+02
```

Listing 1: Exceirpt of the default boundary condition file stdset.bc

3.2 Boundary condition mapping file (∗.map)

The ∗.map file has to be created by the user. It ensures the correct mapping between the surface environment conditions specified in the boundary condition file and the geometrical objects (planar sources and external walls) in the relevant library file, typically the `harness.lib` file for the external walls and the `detailed.lib` file for the planar heat source.

The file structure is the following:

1 (line)	# *Comment_line*	
2 (line)	*Number_of_junction_nodes,*	Initially $n_j = 1$
	number_of_external_nodes	n_e
3 (block)	*Planar_source_value_name_prefix,*	One line per junction node
	power_dissipation_column_reference,	
	number_of_planar_source_values	
4 (block of n_e rows)	*External_node_name_prefix,*	One line per external node
	htc_column_reference,	
	ambient_column_reference,	
	number_of_external_wall_classes	

An example is given in Listing 2.

```
# Boundary condition mapping file for the square PQFP
1, 6
J, 1, 1
TI, 1, 1, 1
TO, 1, 1, 2
BI, 2, 1, 1
BO, 2, 1, 2
S, 3, 1, 2
L, 4, 1, 2
```

Listing 2: Boundary condition mapping file 38.map

Junction nodes are defined first. There are three data items for a junction node. External nodes follow the junction ones. There are four data items for external nodes.

The third line is initially a line and not a block, since FLOPACK currently only deals with mono-chip packages which only have one junction. The power dissipation is then set with just one number, so the *power_dissipation_column_reference* is also initially redundant, there being only one column in the ∗.bc file. In future, this line may become a block, since several junction powers may be entered on the first line of the boundary condition file.

The mapping between nodes and the environmental conditions is explicit through the reference to the column number in the environmental condition file. By contrast, the mapping between nodes and geometrical objects is implicit according to the following rules.

From the external node names provided in the ∗.map file GENBCS generates external wall identifiers of the form *external_node_name'_'digit*, with an underscore between the node name and the digit in order to ease the coding and decoding processes. For instance, if the node name is `TO` and if two external walls are attached to it, the identifiers of those are `TO_1` and `TO_2`. These identifiers both refer to the object and class name.

A similar rule is applied to the junction node. If for instance, the junction node identifier is `J` and if the number of planar sources is one, GENBCS generates a planar source identifier of the form `J_1`. This identifier both refer to the object name and the `value_psource` class name.

3.3 Network definition file (∗.net)

The network definition file defines connectivity (resistances) for the network. The topology, i.e. the nodes, are defined as part of the ∗.dmo file.

A link between nodes node i and node j with $1 \le i < j \le n$ is denoted by L_{ij} with $L_{ij} = 1$ if a resistor exists between the two nodes and $L_{ij} = 0$ if no resistor exists. The file structure is the following:

1 (line)	# *Comment_line*	
2 (block of $n-1$ lines)	$L_{12}, L_{13}, L_{14}, \ldots, L_{1n}$	$n-1$ links for node #1
	$L_{23}, L_{24}, \ldots, L_{2n}$	$n-2$ links for node #2
	$\vdots$	
	$L_{(n-1)n}$	$n-(n-1) = 1$ link for node #$(n-1)$

An example is given in Listing 3.

```
# Network definition file for the square PQFP
1, 1, 1, 1, 1, 0
1, 0, 0, 0, 0
0, 1, 1, 1
1, 0, 0
1, 0
0
```

Listing 3: Network definition file 38.net

In order to specify the matrix of links, the following procedure is convenient:

– write horizontally the list of nodes by omitting the first one;
– write vertically the list of nodes by omitting the last one;
– fill in the cells of the upper triangular matrix with 1 if a link exists and 0 if no link exists.

So, for the network depicted in Fig. 3 made of $n = 7$ nodes the matrix has 6 rows and 6 columns and there are $n(n-1)/2 = 21$ cells to fill in as illustrated in Table 1.

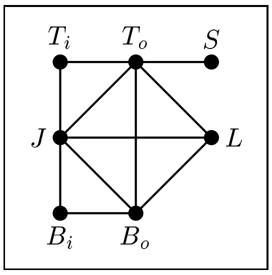

Figure 3: Network example

	TI	TO	BI	BO	L	S
J	1	1	1	1	1	1
TI		1	0	0	0	0
TO			0	1	1	1
BO				1	0	0
L					1	0
S						0

Table 1: Matrix of links

4 Command line format

The FLOPACK internal programs are driven through command line arguments. The rules for formatting the command lines are the following:

– A switch is separated from the previous argument by one or more spaces, e.g. `-ui␣-i` or `-ui 12␣-i`;
– The argument following a switch is considered as the value of it, if it is not itself a switch. The spaces between a switch and its value are mandatory;
– A value without embedded spaces may be specified with or without enclosing quotes, like `-s "abc"` or `-s abc`. The quotes are automatically stripped off by the shell;
– To bound several values to a switch, they must be grouped between quotes, as in `-s "abc def fgi"`;
– When arguments cannot be bound to a switch, they are collected separately. For instance, in the command line `abc -ui 12 def -i 5 ghi`, the three arguments `abc`, `def` and `ghi` are not bound to any switch while `12` is bound to `-ui` and `5` to `-i`.
– Multiple definitions of the same switch are not allowed.

The input format for specifying integer ranges is illustrated by an example dealing with subsets of a 10-element set (Table 2). When a switch meant to specify a range is left undefined, the whole set is assumed to be selected. According to the formatting rules for strings, quotes are optional when the switch value includes no interspersed spaces. Spaces can be freely added provided the range specification is enclosed within quotes, as in `"␣1␣,␣3␣:␣5␣,␣8␣,␣10␣"`.

Input	Subset
`:5`	1, 2, 3, 4, 5
`6:`	6, 7, 8, 9, 10
`2:4`	2, 3, 4
`5:3`	3, 4, 5
`1,10,3:5,8`	1, 3, 4, 5, 8, 10

Table 2: Examples of range specification for a 10-element set

The following conventions are used for documenting a command line:

– The column *switch* gives the character string the switch is made of;
– The column *value* indicates whether the switch takes no value, a compulsory value or an optional value denoted by brackets and roman font ([ ]);
– The column *default* specifies the program behaviour when the switch is undefined or defined without value; for instance, if this column contains 0 [1] for the switch `-s` [*value*], this means that the default is 0 if the switch is undefined or 1 if the switch is defined without value;
– The column *description* summarizes the action performed by the switch.

5 Execution of Genbcs

GENBCS takes a ∗.bc set file and a ∗.map file in order to generate the ∗.bcs file required by SOLCOND and the *header* of the ∗.dmo file required by SOLRES.

The ∗.dmo file is indeed built in two passes. GENBCS writes the header of it and SOLCOND adds the maps of areas, fluxes and temperatures to it. This addition can be made by SOLCOND or externally to it through a UNIX command like *`cat harness.dmo >> 38.dmo`* or an MS-DOS command like *`type harness.dmo >> 38.dmo`* which append the contents of `harness.dmo` to `38.dmo`. To this end, SOLCOND does not generate a leading comment line.

GENBCS is run through the command line *`genbcs` map_file_name*[*`.map`*] [*switches*] where the switches are detailed in Table 3. To get a summary of those or display the GENBCS version number, the command lines to use are *`genbcs -h`* and *`genbcs -v`*.

Switch	Value	Default	Description
`-set`	[*bc_set_file_name*[*.bc*]]	`stdset.bc`	Specify the input file for the ∗.bc data
`-bcs`	[*bcs_file_name*[*.bcs*]]	Standard output	Redirect the output of ∗.bcs data
`-nobcs`			Inhibit the ∗.bcs data generation
`-dmo`	[*dmo_file_name*[*.dmo*]]	Standard output	Redirect the output of ∗.dmo data
`-nodmo`			Inhibit the ∗.dmo data generation
`-log`	[*log_file_name*[*.log*]]	Standard output	Redirect the output of ∗.log data
`-sym`	[*symmetry_factor*]	1 [4]	Specify the symmetry factor
`-nds`	*separator_string*	_ (underscore)	Specify the node digit separator in class names
`-pfx`	*prefix_string*	no prefix	Specify the prefix string of class names
`-opt`	[b][m]		Drive the ∗.log file contents: `b`: log the ∗.bc data `m`: log the ∗.map data

Table 3: Genbcs command line arguments

map_file_name[*`.map`*]

This unbound argument specifies the ∗.map file to process. If missing, the extension `.map` is automatically added. The string used for *map_file_name* is also the default root name for the output files generated by GENBCS.

`-set` [*bc_set_file_name*[*`.bc`*]]

This switch specifies the name of the input file containing the boundary condition data. If missing, the extension `.bc` is automatically added. When this switch is undefined, GENBCS searches for the file `stdset.bc`. When no value is provided, the ∗.bc set file name is defaulted to *map_file_name*[`.bc`].

`-bcs` [*bcs_file_name*[*`.bcs`*]]

This switch specifies the name of the output file used for writing the internal boundary condition data. If missing, the extension `.bcs` is automatically added. When this switch is undefined the ∗.bcs data are written on standard output. When no value is provided, the ∗.bcs file name is defaulted to *map_file_name*[`.bcs`].

-nobcs
: This switch inhibits the generation of ∗.bcs data. The switches `-bcs` and `-nobcs` are mutually exclusive.

-dmo [*dmo_file_name*[*.dmo*]]
: This switch specifies the name of the output file used for writing the header of the ∗.dmo data. If missing, the extension `.dmo` is automatically added. When this switch is undefined the ∗.dmo data are written on standard output. When no value is provided, the ∗.dmo file name is defaulted to *map_file_name*[`.dmo`].

-nodmo
: This switch inhibits the generation of ∗.dmo data. The switches `-dmo` and `-nodmo` are mutually exclusive.

-log [*log_file_name*[*.log*]]
: This switch specifies the name of the output file used for writing the log data. If missing, the extension `.log` is automatically added. When this switch is undefined the log data are written on standard output. When no value is provided, the log file name is defaulted to *map_file_name*[`.log`]. The switch `-opt` enables to drive the contents of the ∗.log file.

-sym [*symmetry_factor*]
: This switch specifies the symmetry factor applied to the power dissipation of junction nodes for the generation of the ∗.bcs file. If the power dissipation is 1 W and the symmetry factor is 4, the power dissipation written in the ∗.bcs file is $1.0/4 = 0.25$ W. If the switch is undefined, the default is 1. If the switch has no value, the default is 4.

-nds separator_string
: This switch specifies the character string used between the node identifiers contained in the ∗.map file and the trailing digit generated by Genbcs. The switch name stands for *node digit separator*. When this switch is undefined, the separator string is the underscore character. This means that if a node identifier is `T0`, the class identifiers generated by Genbcs are `T0_1`, `T0_2`...

-pfx prefix_string
: This switch enables to add the character string *prefix_string* in front of the class identifiers generated by Genbcs. For instance, if this string is made of the `$` character, the identifiers generated from the node name `T0` in the ∗.bcs file are `$T0_1`, `$T0_2`... If this switch is undefined, no prefix is used.

-opt [*b*][*m*]
: This switch drives the contents of the ∗.log file. Each character of the string specifies a category of data. One character at least is required. `bm` or `mb` combines the effect of `b` and `m`. This switch can be used for debugging purpose or for reformatting a ∗.bc file or a ∗.map file generated or modified by hand. The meaning of the character codes is the following:
  - *b* logs the ∗.bc file in the ∗.log file;
  - *m* logs the ∗.map file in the ∗.log file.

Examples.

genbcs 38 -bcs -dmo -sym
: This command generates the files `38.bcs` and `38.dmo` for a quarter model (switch *-sym*) from the ∗.map file `38.map` and the standard set of boundary conditions contained in `stdset.bc`. This is the command used for the generation of the files given in Listing 6 and Listing 7.

genbcs 38 -nobcs -nodmo -opt m
: This command writes the ∗.map data on standard output.

genbcs 38 -nobcs -nodmo -opt m -log 38
: This command writes the ∗.map data in the file `38.log`.

genbcs 38 -nobcs -nodmo -opt b -log 38
: This command writes the ∗.bc data in the file `38.log`.

6 Execution of Solcond

Solcond is aimed specifically at providing a means of running a detailed model of a part through a number of boundary conditions for the purpose of generating the data needed to create a compact model. From the library files describing the model and the ∗.bcs file describing the boundary conditions, Solcond generates the detailed model output file.

Solcond is run through the command line:

solcond lib_file_name_1 [*lib_file_name_2*] [*lib_file_name_3*] [*switches*]

where the switches are detailed in Table 4. To get a summary of those or display the Solcond version number, the command lines to use are *solcond -h* and *solcond -v*.

Switch	Value	Default	Description
`-bcs`	[*bcs_file_name*[*.bcs*]]	No input	Specify the input file for the ∗.bcs data
`-dmo`	[*dmo_file_name*[*.dmo*]]	Standard output	Redirect the output of the ∗.dmo data
`-nodmo`			Inhibit the ∗.dmo data generation
`-log`	[*log_file_name*[*.log*]]	Standard output	Redirect the output of the ∗.log data
`-clp`	[*clipping_cuboid_id*]	[`domain`]	Clip the computational domain
`-nds`	*separator_string*	_ (underscore)	Specify the node digit separator in object names
`-sym`	[*symmetry_factor*]	1 [4]	Specify the model symmetry factor
`-opt`	[`t`][`f`][`r`][`c`][`b`][`d`][`e`]		Drive the ∗.log file contents: `t`: log the temperature of cells `f`: log the thermal maps (areas, fluxes, temperatures) `r`: log the residuals `c`: log the clipping actions `b`: log the flux balance `d`: display the residuals on standard output `e`: display the environment number on standard output
`-nx`	[*grid_line_number*]	0 [5]	Define the number of additional grid lines for Ox
`-ny`	[*grid_line_number*]	0 [5]	Define the number of additional grid lines for Oy
`-nz`	[*grid_line_number*]	0 [5]	Define the number of additional grid lines for Oz
`-r`	[*residual_value*]	10^{-10} [10^{-5}]	Modify the stop criterion for the solver
`-t`	*grid_line_tolerance*	10^{-6}	Modify the grid line tolerance
`-s`			Process the original boundary conditions
`-lib`	[*environment_number*]	0	Write the data structure as a library in the ∗.dmo file
`-dbg`			Dump the internal contents of the data structure

Table 4: Solcond command line arguments

lib_file_name_1[`.lib`]

SOLCOND needs at least one library file provided by the first unbound argument. If missing, the extension is automatically added. The string used for *lib_file_name_1* is also the default root name for the output files generated by GENBCS.

lib_file_name_2[`.lib`] *lib_file_name_3*[`.lib`]

SOLCOND can process several library files (currently three) as if they were pasted together. Typically, the first library file contains the external walls and the clipping cuboid used for the simulation of the detailed model contained in the second library file.

-dmo [*dmo_file_name*[`.dmo`]]

This switch specifies the name of the output file used either for writing the maps of areas, fluxes and temperatures or the library data when the `-lib` switch is enforced. This file is opened in *append mode*. So, if an existing file is attached to the switch, the maps are appended to it. When this switch is undefined the ∗.dmo data are written on the standard output. When no value is provided, the ∗.dmo file name is *lib_file_name_1*[`.dmo`].

-nodmo

This switch inhibits the generation of ∗.dmo data. The switches `-dmo` and `-nodmo` are mutually exclusive.

-clp [*clipping_cuboid_object_identifier*]

This command clips all the objects located outside the clipping cuboid. If no value is attached to the switch, the default identifier `domain` is used.

-nds *separator_string*

This switch specifies the string which enables SOLCOND to extract to node names from the object names during the ∗.dmo file generation. For instance, if the separator string is the default one, i.e. the underscore character, SOLCOND searches for it in the object names and groups the thermal maps according to the stem of these names. For instance, since the object names `TO_1`, `TO_2`... share the same stem, they are associated to the same node, namely `TO`, and the maps of areas, temperatures and fluxes attached to `TO_1`, `TO_2`... are processed accordingly.

-sym [*symmetry_factor*]

This switch specifies the model symmetry factor. The same value than the one used with GENBCS has to be specified. If the switch is undefined, the default is 1. If the switch is defined without value, the default is 4.

`-log` [*lib_file_name_1*[`.log`]]
: This switch specifies the name of the output file used for writing the ∗.log data. If missing, the extension `.log` is automatically added. When this switch is undefined the ∗.log data are written on standard output. When no value is provided, the ∗.log file name is defaulted to *lib_file_name_1*[`.log`].

`-opt` [*t*][*f*][*r*][*c*][*b*][*d*][*e*]
: This command drives the contents of the ∗.log file. Each character of the string specifies a category of data. One character at least is required. The meaning of the character codes is the following:

  - *t* logs the temperatures of the cells;
  - *f* logs the maps of areas, fluxes and temperatures attached to planar sources and external walls;
  - *r* logs the evolution of the solver residuals;
  - *c* logs the clipping actions;
  - *b* logs the overall flux balance;
  - *d* displays the evolution of the solver residuals on standard output;
  - *e* displays the current environment number on standard output.

`-nx` [*grid_line_number*]
`-ny` [*grid_line_number*]
`-nz` [*grid_line_number*]
: These commands specify the number of additional grid lines along each coordinate axis. When the switch is undefined, the default is 0. When the switch has no value, the default is 5.

`-r` [*residual_value*]
: This command specifies the residual value used to drive the conjugate gradient solver. When the switch is undefined, the default is 10^{-10}. When the switch has no value, the default is 10^{-5} in order the speed up the computation.

`-t` *grid_line_tolerance*
: This command enables to modify the grid line tolerance. The default is 10^{-6} m.

`-s`
: When a bcs file is used, Solcond does not solve for the original boundary condition data, i.e. for the ones attached to the objects contained in the library file. This switch enables to specify that the solving must also include the original boundary condition data. When no ∗.bcs file is used, this switch is implicit.

`-lib` [*environment_number*]
: This command writes in the ∗.dmo file the library data reflecting the problem considered by Solcond. This switch automatically disables the generation of the maps normally written in the ∗.dmo file. The boundary data are those specified by the switch value, zero (the default) being by convention the original environment while *environment_number* ≥ 1 refer to the environments described in the ∗.bcs file, if any. Since the generated file has the extension ∗.dmo, it has to be renamed with the extension ∗.lib prior to loading it in Solcond or Flogate. This library file can be used for various purposes:

  - visualise the problem considered by Solcond;
  - check the clipping effect;
  - perform comparisons with Flotherm;
  - simulate a specific environment;
  - reformat a library file…

`-dbg`
: This command dumps the internal contents of the data structure obtained up to the meshing phase.

Examples.

`solcond harness detailed -sym -bcs 38 -dmo 38 -clp -r -log -opt be`
: This command processes the two library files `harness.lib` and `detailed.lib`. A quarter model is simulated (`-sym`) with the boundary conditions contained in `38.bcs` (`-bcs 38`) in order to create the ∗.dmo file `38.dmo` (`-dmo 38`). The computational domain is clipped by the cuboid `domain` (`-clp`). The separator string for the extraction of the node names from the object names is the default one (the underscore). The residual criterion is increased to speed up the execution (`-r`). The balance of fluxes (flag `b` of `-opt`) is written in `harness.log` (`-log`) while the current environment number is displayed on standard output (flag `e` of `-opt`). This command line is used for the generation of the file given in Listing 7. The ∗.log file reproduced in Listing 4 gives the generation order of the thermal maps, which enables to check that this order is consistent with the one provided in the ∗.map file.

solcond harness detailed -sym -bcs 38 -dmo -clp -r -log -opt bde
This command is similar to the previous one, but the trailer of the ∗.dmo file is written in `harness.dmo`. So, the concatenation of the header and trailer of the ∗.dmo file has to be performed manually.

solcond harness detailed -sym -clp -lib -dmo
This command writes in the file `harness.dmo` the library data obtained after clipping the structure defined in `harness.lib` and `detailed.lib`. The boundary conditions are the original ones defined in `harness.lib`.

solcond harness detailed -sym -clp -bcs 38 -lib 11 -dmo 11
This command writes in the file `11.dmo` the library data obtained after clipping the structure defined in `harness.lib` and `detailed.lib`. The boundary conditions correspond to the eleventh environment described in `38.bcs`.

```
# FLOPACK by FLOMERICS - SOLCOND 0.3 Sat Nov 28 18:08:09 1998
# Parsing file harness.lib ...
# ... File harness.lib successfully parsed
# Parsing file detailed.lib ...
# ... File detailed.lib successfully parsed
# Post-processing of object identifiers
# Number of junction objects: 1
# Number of external objects: 6
# List of post-processed identifiers:
# J
# TI
# TO
# BI
# BO
# L
# S
# Size of algebraic system is 150
# Parsing file 38.bcs ...
# ... File 38.bcs successfully parsed
```

Listing 4: Solcond log file harness.log

7 Execution of Solres

Solres creates a network model of a part based on a least squares fit to a set of data for junction temperatures and surface heat fluxes generated by running the detailed model through a number of boundary conditions. Solres fits the network defined in the network definition file, removing negative resistors and resolving the network until one with a set of positive resistances is defined. The current strategy consists of removing the negative resistors one at a time, starting with the largest one.

Solres is run through the command line *solres dmo_file_name -net* [*net_file_name*] [*switches*] where the switches are detailed in Table 5. To get a summary of those or display the Solres version number, the command lines to use are *solres -h* and *solres -v*.

Switch	Value	Default	Description
-net	[*net_file_name*[*.net*]]	No input	Specify the input file for ∗.the net data
-res	[*res_file_name*[*.res*]]	Standard output	Redirect the output of the ∗.res data
-nores			Inhibit the ∗.res data generation
-fit	[*fit_file_name*[*.fit*]]	Standard output	Redirect the output of the ∗.fit data
-nofit			Inhibit the ∗.fit data generation
-log	[*log_file_name*[*.log*]]	Standard output	Redirect the output of the ∗.log data
-ref	*reference_temperature*	0.0	Specify the reference temperature used in backfit
-os	*optimisation_set*	Whole set	Specify the optimisation set
-bs	*backfit_set*	Whole set	Specify the backfit set
-opt	[d][n][r]		Drive the log file contents: d: log the ∗.dmo data n: log the links of the network r: log the conductors and resistors

Table 5: Solcond command line arguments

dmo_file_name[`.dmo`]
: SOLRES needs a ∗.dmo file provided by the first unbound argument. If missing, the extension is automatically added. The string used for *dmo_file_name* is also the default root name for the ∗.net file and the output files generated by GENBCS.

`-net` [*net_file_name*[`.net`]]
: This switch specifies the name of the net file containing the links between nodes. This switch is mandatory. When no value is provided, the ∗.net file name is defaulted to *dmo_file_name*[`.net`].

`-res` [*res_file_name*[`.res`]]
: This switch specifies the name of the output file used for writing the values of the resistors. When this switch is undefined the ∗.res data are written on the standard output. When no value is provided, the ∗.res file name is *dmo_file_name*[`.res`].

`-nores`
: This switch inhibits the generation of ∗.res and ∗.fit data. The switches `-res` and `-nores` are mutually exclusive.

`-fit` [*fit_file_name*[`.fit`]]
: This switch specifies the name of the output file used for writing the results of the backfit. When this switch is undefined the backfit data are written on the standard output. When no value is provided, the ∗.fit file name is *dmo_file_name*[`.fit`]. No backfit is performed when the switch `-nores` is enforced.

`-nofit`
: This switch inhibits the generation of ∗.fit data. The switches `-fit` and `-nofit` are mutually exclusive.

`-ref` *reference_temperature*
: This command enables to specify the reference temperature used for computing the errors during the backfit phase. When this switch is undefined, the reference temperature is 0°C.

`-os` *optimisation_set*
: This command enables to perform the optimisation on a subset of the thermal maps provided in the ∗.dmo file. If for instance the ∗.dmo file is related to the default set made of 38 environmental conditions defined in `stdset.bc`, a range `10:20` means that the optimisation uses only the maps 10 to 20.

`-bs` *backfit_set*
: This command enables to perform the backfit on a subset of the thermal maps provided in the ∗.dmo file. A range `20:30` means for instance that the backfit is only performed for the maps 20 to 30.

`-log` [*log_file_name*[`.log`]]
: This switch specifies the name of the output file used for writing the ∗.log data. If missing, the extension `.log` is automatically added. When this switch is undefined the ∗.log data are written on standard output. When no value is provided, the ∗.log file name is defaulted to *dmo_file_name*[`.log`]. By default, the ∗.log data are the negative conductances encountered during the optimisation process.

`-opt` [*`d`*][*`n`*][*`r`*]
: This command drives the contents of the ∗.log file. Each character of the string specifies a category of data. One character at least is required. The meaning of the character codes is the following:
  - *`d`* logs the details of the input maps provided through the ∗.dmo file;
  - *`n`* logs the links between nodes, including the extra nodes created by SOLRES for conveying the ambient temperature;
  - *`r`* logs all the conductor and resistor values, including the negative conductor removed during the optimisation process.

Examples.

`solres 38 -net -res -fit -opt nr`
: From the ∗.dmo file `38.dmo` and the ∗.net file `38.net` reproduced in Listing 3, this command generates the file `38.res` given in Listing 8 and the backfit file `38.fit` given in Listing 9. The ∗.log file `38.log` reproduced in Listing 5 contains the network (flag `n`) and the values of conductors and resistors (flag `r`). This listing shows that the original network is augmented with a set of nodes the role of which is to convey the ambient temperature related to an external node. These nodes at fixed temperature are named *boundary nodes*. Their names is made after the external node name prefixed with an underscore.

```
# FLOPACK by FLOMERICS - SOLRES 0.0 Tue Dec 22 15:19:38 1998
# Upper triangular matrix of links
      TI  TO  BI  BO   L   S _TI _TO _BI _BO  _L  _S
  J:  TI  TO  BI  BO   L   -   -   -   -   -   -   -
 TI:      TO   -   -   -   - _TI   -   -   -   -   -
 TO:           -  BO   L   S   - _TO   -   -   -   -
 BI:              BO   -   -   -   - _BI   -   -   -
 BO:                   L   -   -   -   - _BO   -   -
  L:                       -   -   -   -   -  _L   -
  S:                           -   -   -   -   -  _S
_TI:                               -   -   -   -   -
_TO:                                   -   -   -   -
_BI:                                       -   -   -
_BO:                                           -   -
 _L:                                               -
# Start of least-square optimization...
# Negative conductor no.4 removed (G = -9.547743e-03)
# ...End of least-square optimization
# Values of conductances and resistances
  J  TI  7.053368e-02  1.417762e+01
  J  TO  1.703852e-02  5.869056e+01
  J  BI  7.637482e-02  1.309332e+01
  J  BO  1.967315e-02  5.083070e+01
  J   L -9.547743e-03  1.000000e+15
 TI  TO  7.244827e-03  1.380295e+02
 TO  BO  1.480812e-01  6.753052e+00
 TO   L  8.130342e-02  1.229961e+01
 TO   S  3.666614e-02  2.727312e+01
 BI  BO  6.175241e-03  1.619370e+02
 BO   L  8.626244e-02  1.159253e+01
# Number of conductors removed from the network: 1
```

Listing 5: Solres log file 38.log

8 Output data file specification

8.1 Solcond internal boundary condition file (∗.bcs)

The ∗.bcs file mechanism is an extension of the FLOGATE grammar enabling to solve for a set of problems sharing the same geometry but differing only in the boundary conditions characterizing a given environment. The ∗.bcs syntax is made of *update blocks* enclosing *class blocks* made of pairs (*attributes*, *values*):

```
update "string" {
  class_name = name {
    attribute = value
  }
}
```

An example is given in Listing 6. An update block groups all the classes the attributes of which need to be updated. A class block groups all the attributes the values of which are updated with respect to the previous update block. For the first block, the considered values are the ones contained in the library file.

```
# FLOPACK by FLOMERICS. BCS data. GENBCS version 0.0 Thu Nov 26 14:10:34 1998
# bcs file '38.bcs'
# Symmetry factor: 4 Prefix: '' Separator: '_' Number of updates: 38
update "bc1" {
  value_psources = J_1 {
    total_source = 2.500e-01
  }
  ext_walls = TI_1 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
  ext_walls = TO_1 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
  ext_walls = TO_2 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
  ext_walls = BI_1 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }

  ext_walls = S_2 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
  ext_walls = L_1 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
  ext_walls = L_2 {
    ext_htc         = 1.000e+04
    ext_temperature = 0.000e+00
  }
}

update "bc38" {
  ext_walls = L_1 {
    ext_htc         = 5.000e+02
  }
  ext_walls = L_2 {
    ext_htc         = 5.000e+02
  }
}
```

Listing 6: Exceirpt of the Solcond internal boundary condition file 38.bcs

8.2 Detailed model output file (∗.dmo)

The ∗.dmo file contains the thermal maps attached to each node. The contents of it is the same, irrespective of whether the user has run a quarter model or a full model, provided the same grid is used in each case.

This file is built in two passes. The header generated by GENBCS contains the node names, power dissipation, heat transfer coefficients and ambient temperatures, which enables the input to SOLRES to be restricted to two input files: the ∗.dmo file and the ∗.net file. The footer generated by SOLCOND contains the nodal areas and thermal maps, i.e. the nodal temperatures and fluxes.

The file structure is the following:

1 (line)	# *Comment_line*	
2 (line)	*Number_of_junction_nodes,*	Initially $n_j = 1$
	number_of_external_nodes,	n_e
	number_of_environmental_conditions,	m_e
	node_identifier_length	
3 (block of n_j lines)	*Junction_node_name, power_dissipation*	One row per junction node
	External_node_name, ambient_temperature	One row per external node
4 (block of m_e lines)	*environment_number,* htc_1, htc_2, …, htc_{n_e}	One row per environment
5 (block of n lines)	*Nodal_areas*	One line per node
6 (block of $2 \times m_e$ lines)	*Environment_number,* $T_1, T_2, \ldots T_n$	One line per environment
	Environment_number, $q_1, q_2, \ldots qn$	One line per environment

An example is given in Listing 7.

```
# FLOPACK by FLOMERICS. DMO data. GENBCS 0.0 Thu Nov 26 14:10:34 1998
1, 6, 38, 2
 J, 1.000e+00
TI, 0.000e+00
TO, 0.000e+00
BI, 0.000e+00
BO, 0.000e+00
 S, 0.000e+00
 L, 0.000e+00
  1, 1.000e+04, 1.000e+04, 1.000e+04, 1.000e+04, 1.000e+04, 1.000e+04
  2, 1.000e+03, 1.000e+03, 1.000e+03, 1.000e+03, 1.000e+03, 1.000e+03

 37, 1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02
 38, 1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02, 1.000e+02, 5.000e+02
9.072563e-05
1.123600e-04
6.643769e-04
1.123600e-04
6.643769e-04
1.336400e-05
1.950900e-04
  1, 5.410499e+00, 3.160685e-01, 1.666010e-02, 3.445124e-01, 2.041770e-02, 4.592711e-02, 2.701813e-03
  1, 1.000000e+00,-3.551346e-01,-1.106858e-01,-3.870941e-01,-1.356505e-01,-6.137698e-03,-5.270967e-03
  2, 7.592822e+00, 2.798530e+00, 2.350704e-01, 2.971017e+00, 2.700731e-01, 2.008964e-01, 6.866628e-02
  2, 1.000000e+00,-3.144429e-01,-1.561753e-01,-3.338234e-01,-1.794304e-01,-2.684779e-03,-1.339610e-02

 37, 1.917262e+01, 1.549957e+01, 4.296456e+00, 1.584419e+01, 4.468572e+00, 3.772904e+00, 3.098692e+00
 37, 1.000000e+00,-1.741531e-01,-2.854466e-01,-1.780253e-01,-2.968816e-01,-5.042108e-03,-6.045238e-02
 38, 1.909938e+01, 1.543290e+01, 4.184316e+00, 1.577734e+01, 4.353959e+00, 3.603590e+00, 2.971988e+00
 38, 1.000000e+00,-1.734041e-01,-2.779963e-01,-1.772741e-01,-2.892669e-01,-2.407919e-02,-5.798051e-02
```

Listing 7: Exceirpt of the detailed model output file 38.dmo

8.3 Network model file (∗.res)

The ∗.res file is similar to the ∗.net one, except that the flags L_{ij} have been replaced by values denoted as R_{ij}. These latter are written to six significant digits in scientific format. The file also contains the nodal names and areas, thus providing a complete thermal (but not geometrical) definition of the compact mdel. Resistances not included in the network are indicated by a value of 10^{15}.

The file structure is the following:

1 (line)	`#` *Comment_line*	
2 (line)	*Number_of_junction_nodes,*	Initially $n_j = 1$
	number_of_external_nodes,	n_e
	number_of_optimisation_conditions,	
	node_identifier_length	
3 (block of n lines)	*Node_names,*	one line per node
	nodal_areas	
4 (block of $n-1$ lines)	$R_{12}, R_{13}, R_{14}, \ldots, R_{1n}$	$n-1$ resistor values for node #1
	$R_{23}, R_{24}, \ldots, R_{2n}$	$n-2$ resistor values for node #2
	$\vdots$	
	$R_{(n-1)n}$	$n-(n-1) = 1$ resistor value for node #$(n-1)$

An example is given in Listing 8.

```
# FLOPACK by FLOMERICS - SOLRES 0.0 Tue Dec 22 15:19:38 1998
1, 6, 38, 2
  J, 9.072563e-05
 TI, 1.123600e-04
 TO, 6.643769e-04
 BI, 1.123600e-04
 BO, 6.643769e-04
  L, 1.336400e-05
  S, 1.950900e-04
1.417762e+01, 5.869056e+01, 1.309332e+01, 5.083070e+01, 1.000000e+15, 1.000000e+15
1.380295e+02, 1.000000e+15, 1.000000e+15, 1.000000e+15, 1.000000e+15
1.000000e+15, 6.753052e+00, 1.229961e+01, 2.727312e+01
1.619370e+02, 1.000000e+15, 1.000000e+15
1.159253e+01, 1.000000e+15
1.000000e+15
```

Listing 8: Network model file 38.res

8.4 Network backfit file (∗.fit)

This file gives the backfit to the junction temperature and surface heat and fluxes for each of the environmental conditions in the boundary condition set. The definition of the absolute and relative errors referred to below are as follows:

$$\begin{aligned} \mathit{absolute\_error} &= (\mathit{compact\_model\_nodal\_value} - \mathit{reference\_value}) \\ &\quad - (\mathit{detailed\_model\_nodal\_value} - \mathit{reference\_value}) \\ &= \mathit{compact\_model\_nodal\_value} - \mathit{detailed\_model\_nodal\_value} \\ \mathit{relative\_error} &= \mathit{absolute\_error}/(\mathit{detailed\_model\_nodal\_value} - \mathit{reference\_value}) \end{aligned}$$

The relative error is expressed as a percentage.

The file structure is as follows. The number of blocks is equal to the cardinality of the backfit subset as specified through the SOLRES command line. The *optimization_flag* is either equal to 1 if the environment condition is used in the network optimization or equal to 0 otherwise.

1 (line)	# *Comment_line*	
2 (line)	*Number_of_junction_nodes,*	Initially $n_j = 1$
	number_of_external_nodes,	n_e
	number_of_backfit_conditions,	
	node_identifier_length,	
	reference_temperature	
3 (block of $n+1$ lines)	*Environment number,*	
	optimization_flag	
	Node_name,	One line per node
	detailed_model_nodal_value,	
	compact_model_nodal_value,	
	absolute_error,	
	relative_error,	

An example is given in Listing 9.

```
# FLOPACK by FLOMERICS - SOLRES 0.0 Tue Dec 22 15:19:38 1998
1, 6, 38, 2,  0.000e+00
  1, 1
  J 5.410e+00 5.729e+00  3.189e-01  5.894e+00
 TI 3.161e-01 3.365e-01  2.040e-02  6.454e+00
 TO 1.666e-02 1.490e-02 -1.761e-03 -1.057e+01
 BI 3.445e-01 3.629e-01  1.837e-02  5.332e+00
 BO 2.042e-02 1.708e-02 -3.337e-03 -1.634e+01
  L 4.596e-02 8.914e-03 -3.704e-02 -8.060e+01
  S 2.703e-03 2.749e-04 -2.428e-03 -8.983e+01
  2, 1
  J 7.593e+00 7.918e+00  3.253e-01  4.284e+00
 TI 2.799e+00 2.946e+00  1.472e-01  5.259e+00
 TO 2.351e-01 2.214e-01 -1.368e-02 -5.820e+00
 BI 2.971e+00 3.110e+00  1.394e-01  4.693e+00
 BO 2.701e-01 2.449e-01 -2.520e-02 -9.331e+00
  L 2.010e-01 2.163e-01  1.527e-02  7.598e+00
  S 6.872e-02 3.503e-02 -3.369e-02 -4.903e+01

 38, 1
  J 1.910e+01 1.841e+01 -6.855e-01 -3.589e+00
 TI 1.543e+01 1.494e+01 -4.949e-01 -3.207e+00
 TO 4.184e+00 4.264e+00  8.015e-02  1.915e+00
 BI 1.578e+01 1.529e+01 -4.898e-01 -3.104e+00
 BO 4.354e+00 4.437e+00  8.313e-02  1.909e+00
  L 3.604e+00 4.186e+00  5.828e-01  1.617e+01
  S 2.972e+00 2.783e+00 -1.885e-01 -6.343e+00
```

Listing 9: Exceirpt of the Backfit file 38.fit