Flair Advanced Features

Advanced FLUKA Course

About

/fleə(r)/ n [U,C] natural or instinctive ability (to do something well, to select or recognize what is best, more useful, etc.

[*Oxford Advanced Dictionary of Current English*]

What is flair [1/2]

FLUKA Advanced Interface [http://www.fluka.org/flair]

- **All-in-one** User friendly graphical Interface;
- Minimum requirements on additional software;
- Working in an intermediate level
  Not hiding the inner functionality of FLUKA

Front-End interface:

- Fully featured **Input file Editor**
  - Mini-dialogs for each card, allows easy and almost error free editing
  - Uniform treatment of all FLUKA cards
  - Card grouping in categories and card filtering
  - Error checking and validation of the input file during editing
- **Geometry:** interactive visualization editing, transformation, optimizations and debugging (tomorrows talk);
- **Compilation** of the FLUKA Executable;
- **Running** and **monitoring** of the status of a/many run(s)

What is flair [2/2]

Back-End interface:

- Inspection of the output files (core dumps and directories)
- Output file(s) viewer dividing into sections
- Post processing (merging) the output data files
- Plot generation through an interface with gnuplot;

Other Goodies:

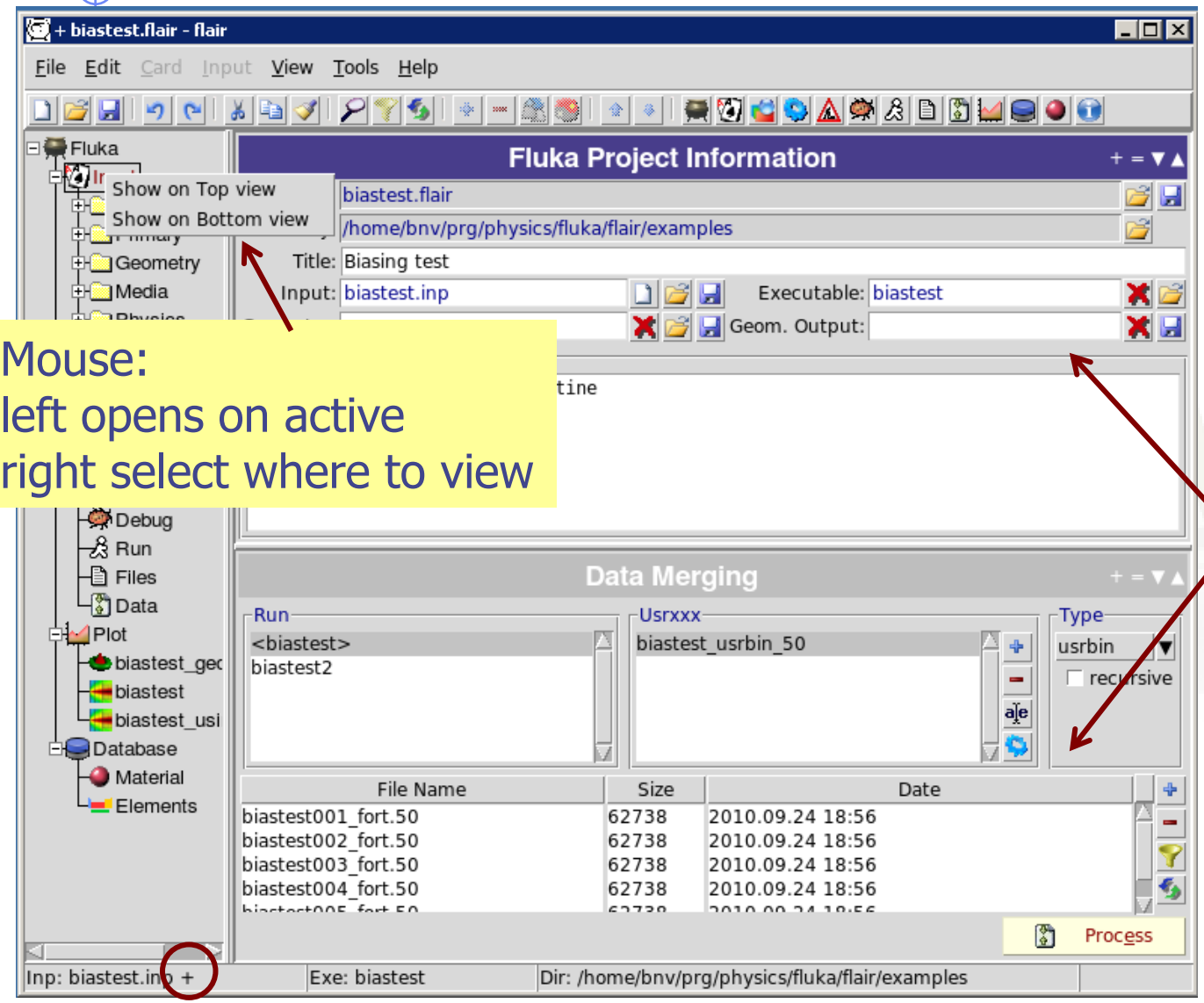
- Access to FLUKA manual as hyper text
- Checking for release updates of FLUKA and flair
- Nuclear wallet cards
- Library of materials
- Database of geometrical objects (Not yet completed)
- Programming python **API**
- Everything is accessible with keyboard shortcuts

Concepts: Flair Project

- Store in a single file all relevant information:
  - Project notes
  - Links to needed files: input file, source routines, output files …
  - Multiple runs from the same input file, as well running status
  - Procedures on how to run the code
  - Rules on how to perform data merging
  - Information on how to post process and create plots of the results
- You can consider Flair as an editor for the project files.
- Can handle any FLUKA input format (reading & writing), but internally it works using the names format for the input, free with names for the geometry (Recommended way of working)
- The format is plain ASCII file with extension: .flair

Note: If you want to copy a project you need to copy also all linked files especially the input and source routines!

Interface



Mouse:
left opens on active
right select where to view

active

+ vertical/horizontal
= equalize
▼ minimize
▲ maximize

2 working frames

inactive
click to activate

input modified and not saved

Command line options

Usage: flair [options] <filename | filename.flair | filename.inp>

Options:

- -d/D Activate/Deactivate the beta-development features
- -e exe Use exe as fluka executable
- **-g** Open geometry editor window
- -i inputfile Fluka input file (w/o the .inp extension)
- **-r** Load most recent project
- -R # Load recent project (number 1..10 or filename)
- -l List recent projects
- -x Run through an xterm (default)
- **-X** Run without an xterm (useful in case of start failure)
- -1 Load the first flair file in the folder

Interface

Keyboard:

Almost everything is possible with the keyboard see manual for shortcuts

Ctrl-Enter: Execute most important action

Ins/Del: Add or Delete

Mouse:

right-click anywhere to get a popup menu

Listboxes:

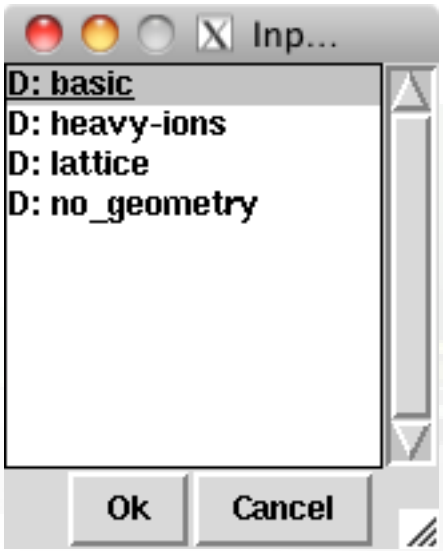
all listboxes are searchable. Typing only the characters (A-Z) and numbers (0-9) all other are ignored

LabelFrames:

can be collapsed/expanded by clicking on the label

Input Templates

- When requesting a new input or a new project flair will prompt to select an input template:



Default template: basic.inp

```
TITLE

GLOBAL                                              1.0       1.0
DEFAULTS                                                            NEW-DEFA
BEAM
BEAMPOS
GEOBEGIN                                                            COMBNAME
    0    0
* Black body
SPH blkbody    0.0 0.0 0.0 10000000.0
* Void sphere
SPH void       0.0 0.0 0.0 1000000.0
* Cylindrical target
RCC target     0.0 0.0 0.0 0.0 0.0 10.0 5.0
END
* Black hole
BLKBODY      5  +blkbody -void
* Void around
VOID         5  +void -target
* Target
TARGET       5  +target
END
GEOEND
* ..+....1....+....2....+....3....+....4....+....5....+....6....+....7..
ASSIGNMA    BLCKHOLE    BLKBODY
ASSIGNMA      VACUUM       VOID
ASSIGNMA      COPPER     TARGET
RANDOMIZ         1.0
START
STOP
```

- Flair default templates are prefixed with "D:"
- User templates will be prefixed with "U:"

The user can create his own set of input templates. They are normal FLUKA input files and they have to be placed in the directory ~/.flair/templates (create the directory if not existing)

Card Categories

For easier access, cards are groups in the following categories:

- **General** General purpose (TITLE, DEFAULTS, GLOBAL…);
- **Primary** Definition of the primary starting particles;
- **Geometry** Cards related to the definition of the geometry bodies/regions/lattices plotting and rotations/translations;
  - **Bodies** Subcategory containing only the bodies definition;
  - **Transformations** Subcategory containing only the geometrical directives;
- **Media** Definition and assignment of materials;
- **Physics** Setting physics properties of the simulation;
- **Transport** Modify the way particles are transported in FLUKA;
- **Biasing** Cards for importance biasing definition;
- **Scoring** Cards related to scoring;
- **Flair** flair special cards;
- **Preprocessor** Definitions for creating conditional input files.

Concepts: Extended Cards [1/2]

- Flair is treating the input file as a list of extended cards;
- Each extended card contains:
  - Comment: All commented lines preceding the card(s) as well the inline comments;
  - Tag: The 8 character word identifying the card. All tags not recognized by flair will be converted to #error;
  - WHATs: Multiple number of WHATs (0=sdum, 1-6 first line, 7-12 continuation line...)
  - Extra: multi line string of extra information for special cards like REGION, TITLE, PLOTGEOM etc.
  - State (Enable/Disable);
- Flair recognize automatically (and separates them from the comments) all the disabled valid FLUKA cards;

Concepts: Extended Cards [2/2]

- The region definition in the in geometry is emphasized by the presence of a card named "REGION";
- All the COMPOUND cards related to one material are joined in one card;
- Cards are edited with the flair editor through the use of the mini-dialogs, forcing the user to enter the *correct* information (default);
- The user can nevertheless gain full control of the card using the Edit dialog (*Ctrl-E);*
- Flair will try to find the best floating point representation of each number, to ensure the maximum accuracy; number of digits that fits in the specific width (10 for the fixed format, 22 for the free format).
- Function evaluation: a field value starting with **=** will force flair to evaluate its content as a function e.g.

  BEAMPOS x: =2*10+length

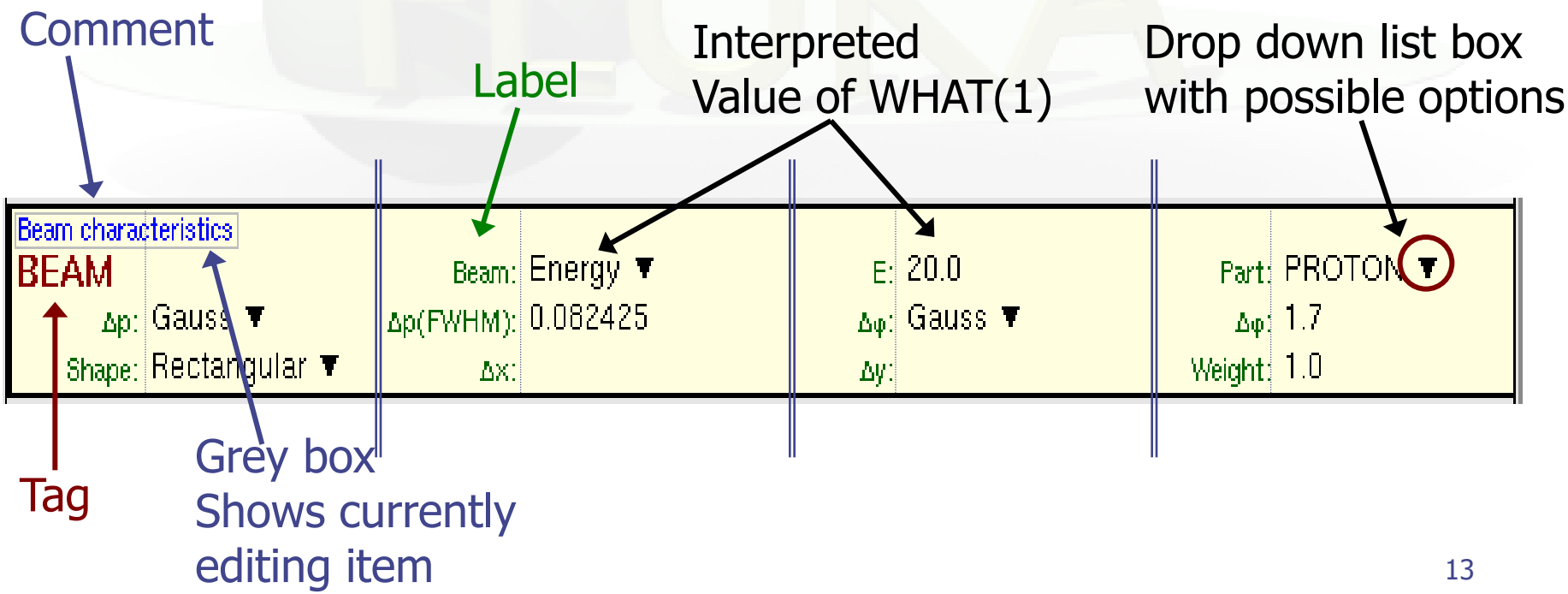
  Flair will create a valid fluka input containing the evaluation of the formula and keep the formula inside the comments as

  *@what.1 =2*10+length

Anatomy of a card mini-dialog [1/2]

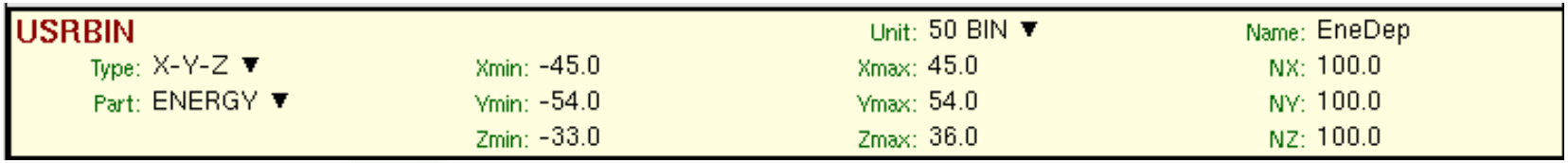
- For each extended card flair has a mini dialog (currently in 4 columns), interpreting all information stored in the card

```
* Beam characteristics
BEAM              -20.0 -0.082425      -1.7                          1.0PROTON
```

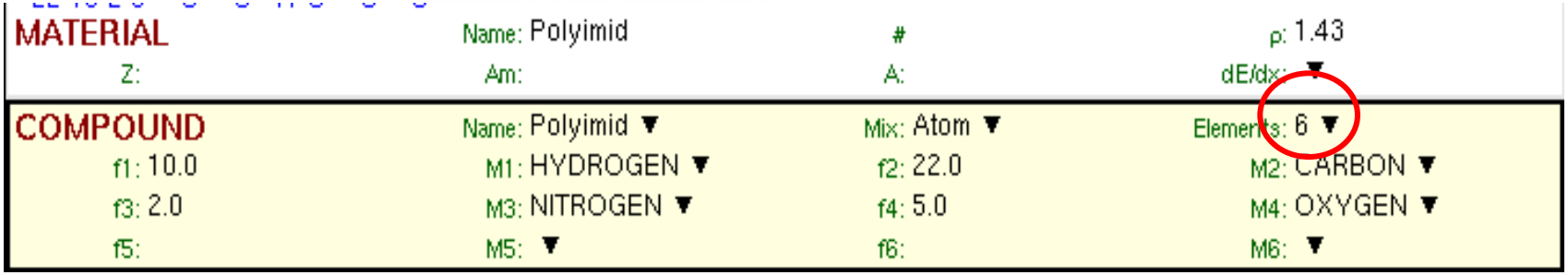


Anatomy of a card mini-dialog [2/2]

```
* Energy deposition in 3D binning
USRBIN         10.0     ENERGY      -50.0       45.0       54.0       36.0EneDep
USRBIN        -45.0      -54.0      -33.0      100.0      100.0      100.0&
```




```
* Polypyromellitimide Polyimide, Kapton
* Chemical        O = C   H-C   C = O
* Formula           / \  // \ / \      H-C - C-H        H-C - C-H
*                  /   \C    C   \    //    \\       //     \\
*            ---- N     |    ||   N - C       C - O - C        C ----
*                  \   /C    C   /     \     /         \     /
*  C  H  N O        \ /  \\ / \ /       H-C = C-H        H-C = C-H
*   22 10 2 5     O = C   H-C    C = O
MATERIAL                                 1.43                        Polyimid
COMPOUND        10.0  HYDROGEN        22.0     CARBON        2.0  NITROGENPolyimid
COMPOUND         5.0    OXYGEN                                       Polyimid
```



Input Editor - 1

highlight differences during editing

Input Editor - 2

- Drag'n'drop from the TAG of the cards
- Double click on card TAG to select all similar cards
- Editing multiple cards: select cards and modifying the value in one card will propagate the change to all similar selected cards
- Ctrl-Double-Click Show/Hide selected cards
- #if..#endif, $transform, $translat or $expand flair will enclose the selected cards with the #if #endif, or $start_xxx, $end_xxx transformation cards
- Popup Balloon tooltip displays short help:
  - for every option on every card
  - body description in the REGION expression
- Right-click: shows popup-menu
  - Quick filtering by REGION, MATERIAL, scoring etc…
- Easter Eggs: AWARI by Double-Right-Click on dialog showing the card representation as text at the bottom of the screen

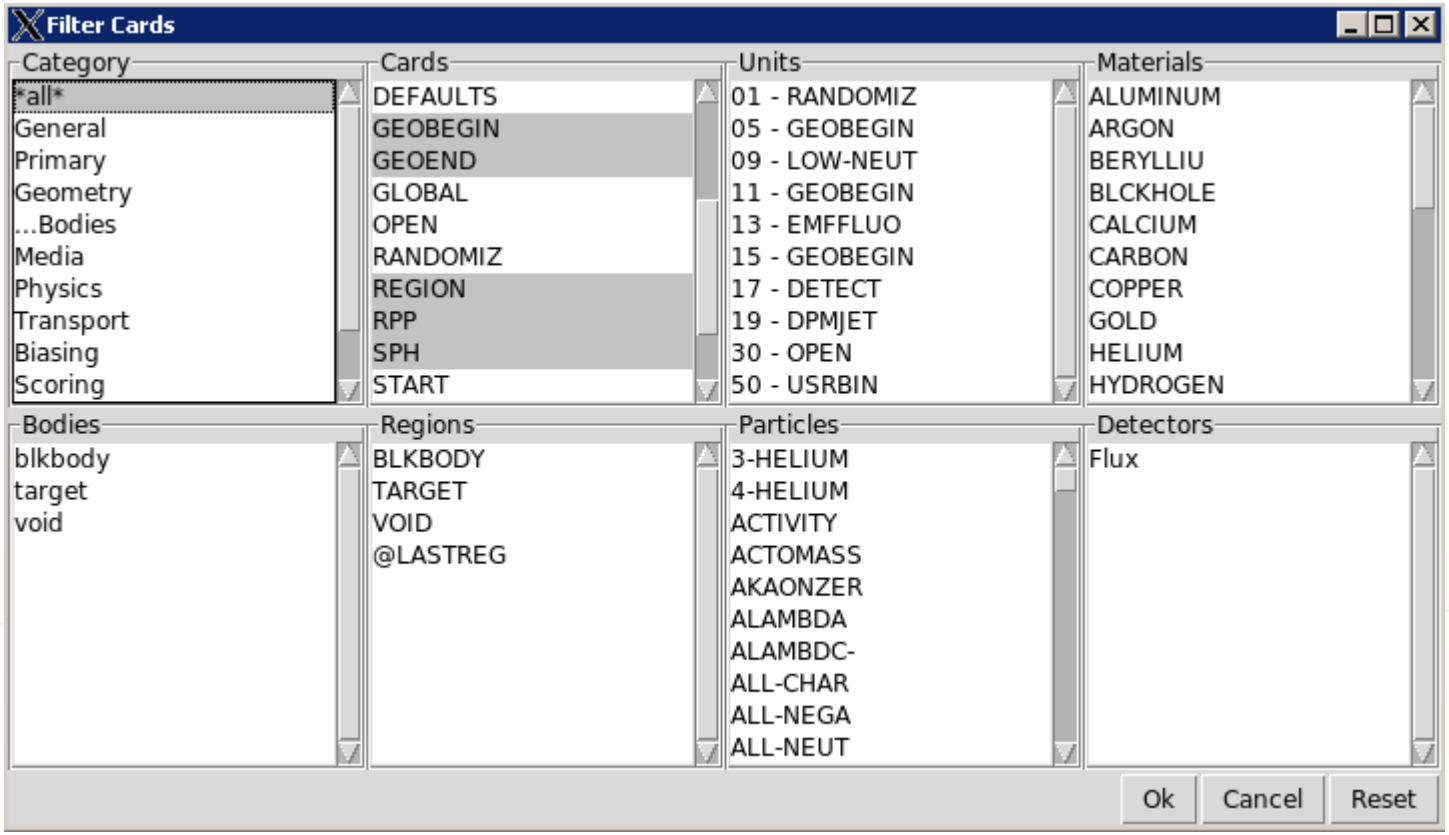
Input Editor - 3

- Automatic indentation of nested #if..#endif and $start..$end directives.
- To refresh the display press Ctrl-R

- Each REGION can be split into many cards if needed to be used with preprocessor commands.
- Use as a name "&"

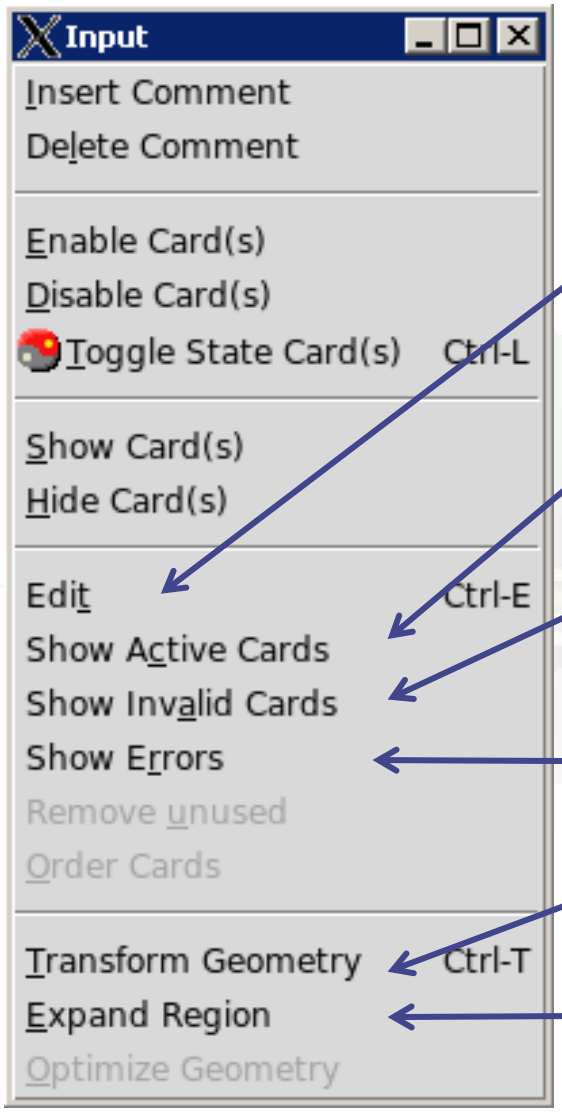
```
Void around
REGION VOID                                   Neigh: 5          Volume:
     Expr: +void -target
#if        BIAS ▼
   REGION &                  cont: -bias
#endif
```

Input Card Filtering



- Filter Cards dialog allows a more advanced selection of cards to be displaced, by showing only the cards that match the selected options

Input Menu



Manual editing of the card

Scan input and display only active cards (not excluded by the preprocessor)

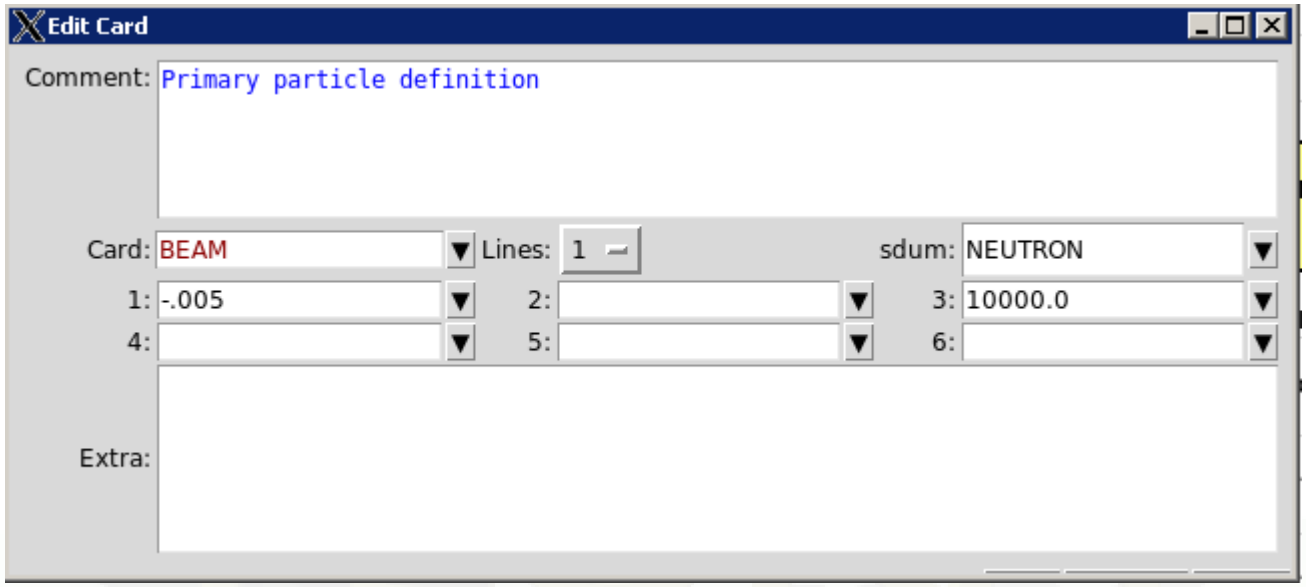
Show cards containing problems/errors

Display a message with the errors identified in the cards

Open the geometry transformation dialog

Expand parenthesis in the region (only logical optimization will performed)

Manual Card Editing



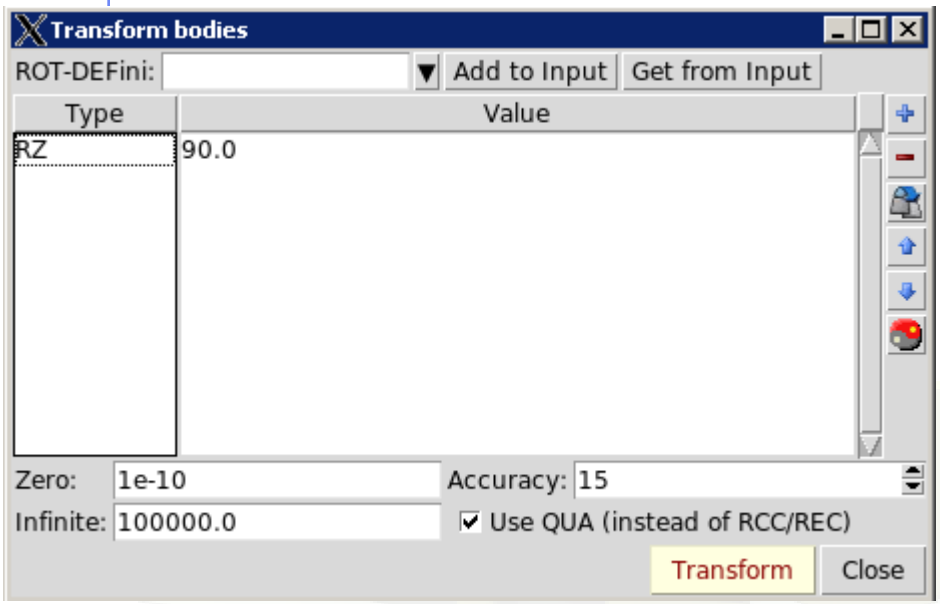
Accessible: Ctrl-E or right-click → Edit, Menu → Input → Edit

Lines: Number of lines the card extends

Extra: additional information for a card like title string for TITLE, or region expression for REGION

Dropdown box: shows with categories all items defined in the input (bodies, regions, materials, particles…)

Bodies Transformation



Transformation Types:

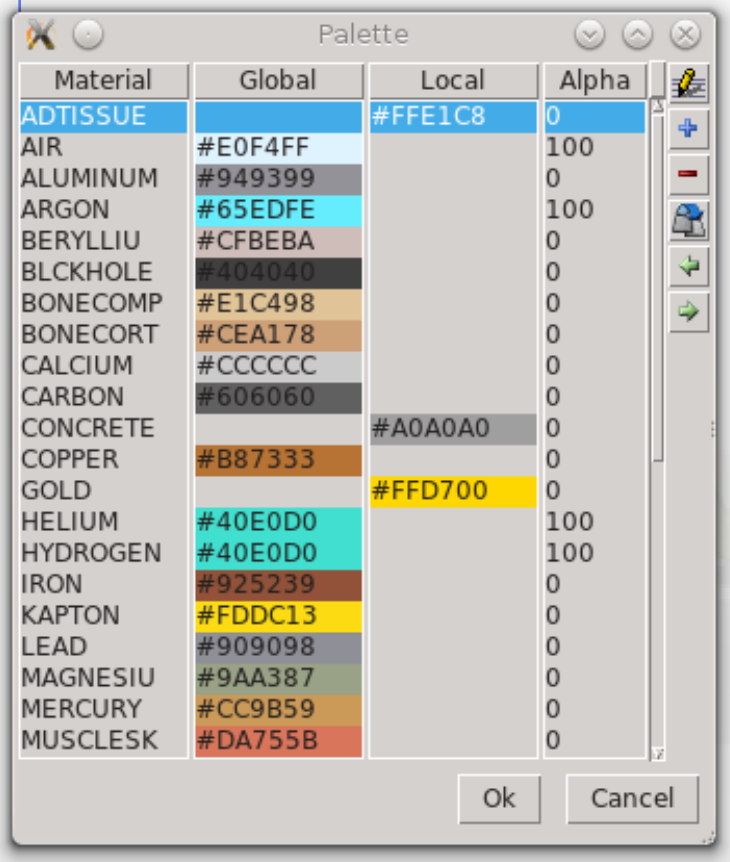
T	translate along a vector
TX TY TZ	translate along axis
RX RY RZ	axis rotation (degrees)
S	scaling

- Applies a user transformation to the selected bodies on the input editor.
- Convert transformations to/from ROT-DEFini cards
- Zero: limit below which to be considered as zero
- Accuracy: Numeric digits
- Infinite: infinite bodies when converted to which size to use
- Use QUA: convert infinite cylinders to infinite QUAdrics

Remember:

When transforming bodies for use with LATTICE card, use the maximum precision

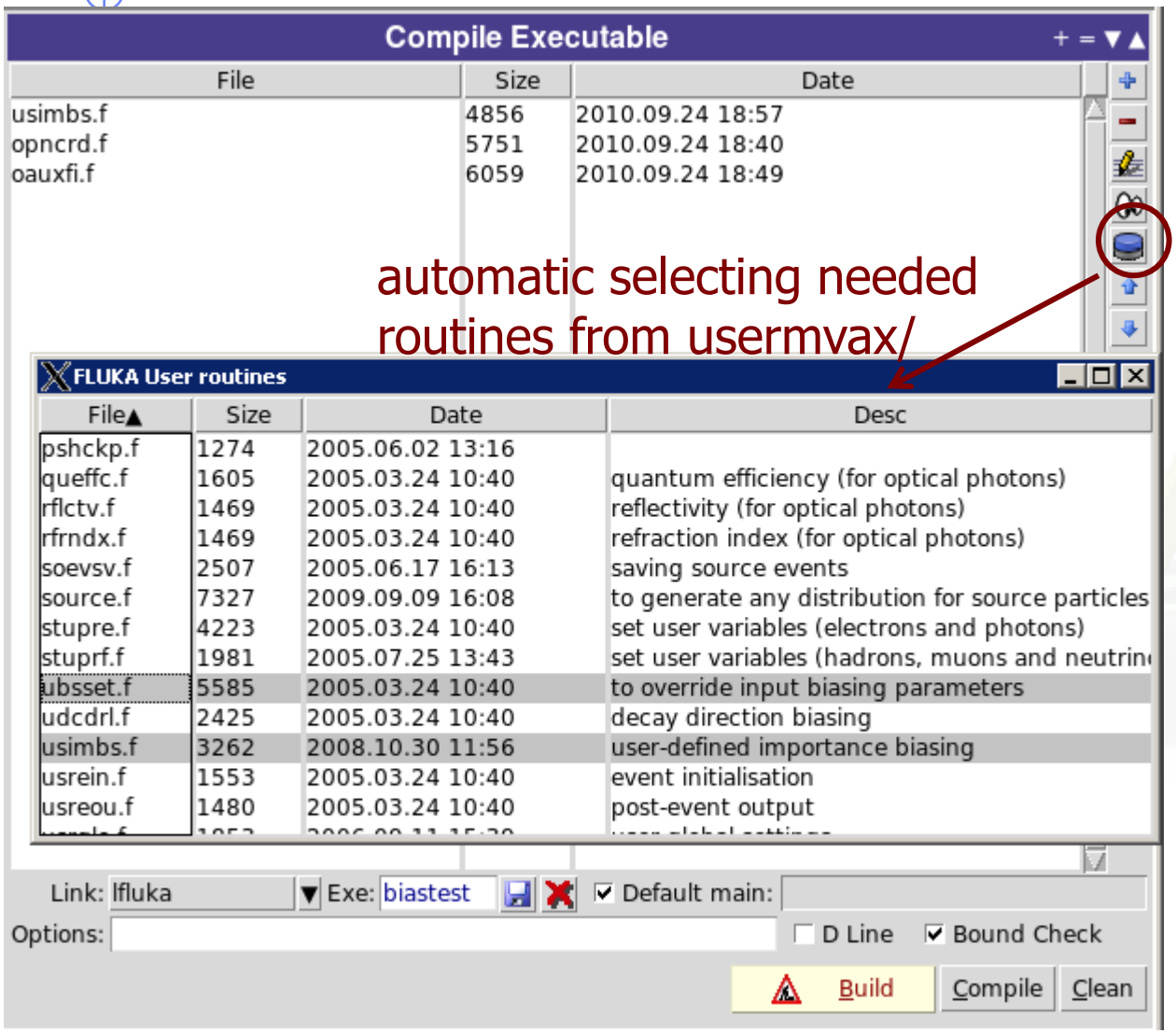
Color Palette



Accessible: Menu → View → Palette

- Edit colors used for material display in Geometry plots and GeometryEditor
- Global colors are saved inside flair.ini and are shared between all projects
- Local colors are initially randomly assigned and saved inside the project file
- Alpha channel for setting transparency on materials. Used on the 3D raytracing plotting.
  0 = opaque
  100 = transparent

Compiling



Filetypes accepted:

- Fortran: .f, .F, .for, .FOR
- C/C++: .c, cpp, .cxx, .cc
- Libraries: .a, .so

Automatic scanning of necessary user routines and copying them to project folder.

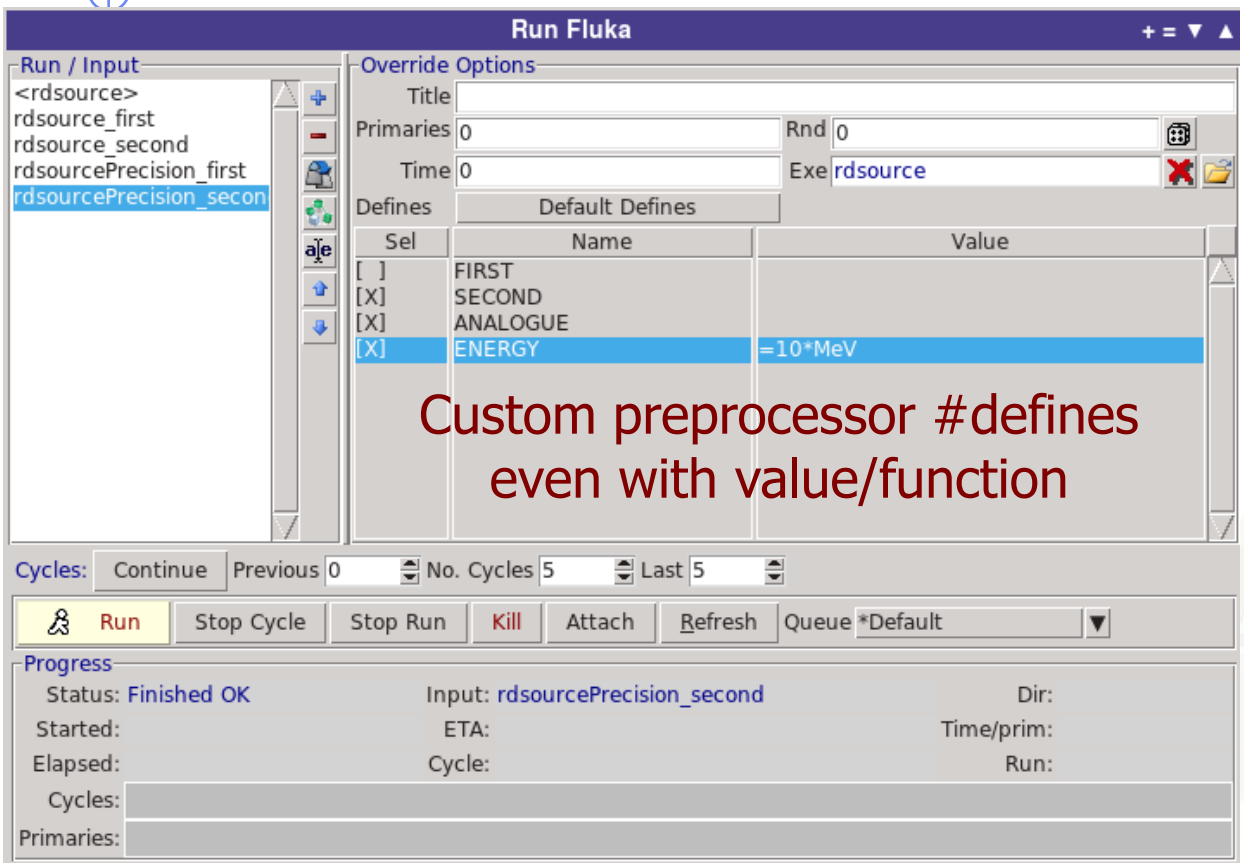
Build: behaves like a "makefile" compiles based on files timestamp when are newer

Compile: Forces compile of the selected files

Clean: cleanup of all produced files

When you are unsure, click on "Clean" before "Build"

Running



<inputname> refers to the input file AS IT IS in the input editor.

Create additional runs based on the same input file by overriding:

- Title
- Preprocessor definitions
- Random number seed
- Starting particles
- Execution timeout
- Executable

- Monitors the status of the run by inspecting the FLUKA output files. If timeout occurs try to re-Attach to the running process.
- The timeout is user-definable in the Preferences dialog

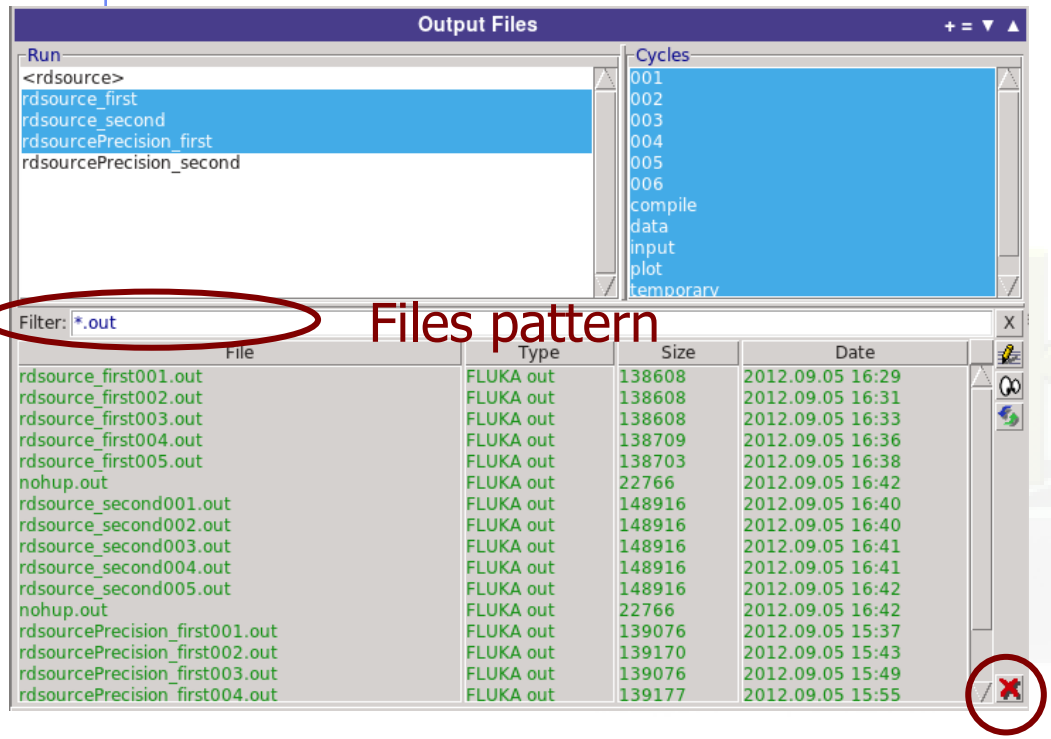
Running: How to use multicore CPU's

- Create clones of the current input e.g. test.inp named: test1.inp, test2.inp, test3.inp …
- Assign a different random number seed on each run (Rnd entry)
- Select all in the listbox and click Run

Multiple Selection:

- To modify many runs at the same time, select them in the listbox
- The options will be "*disabled*"
- Right-click on the options you want to enable and modify them
- Modify the filters in Data processing for summing up all cycles from all runs (see later)

Output Files



Delete selected files

Inspect Output files generated by FLUKA classified per:

Run/Cycle

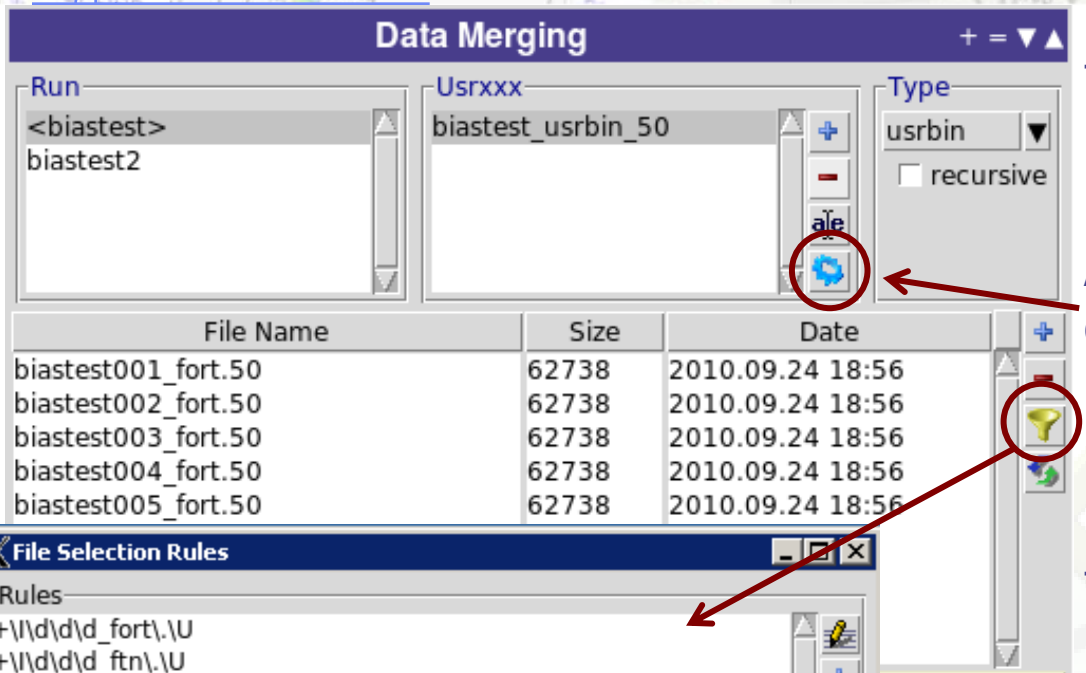
As well special output files from compilation data processing plotting and temporary

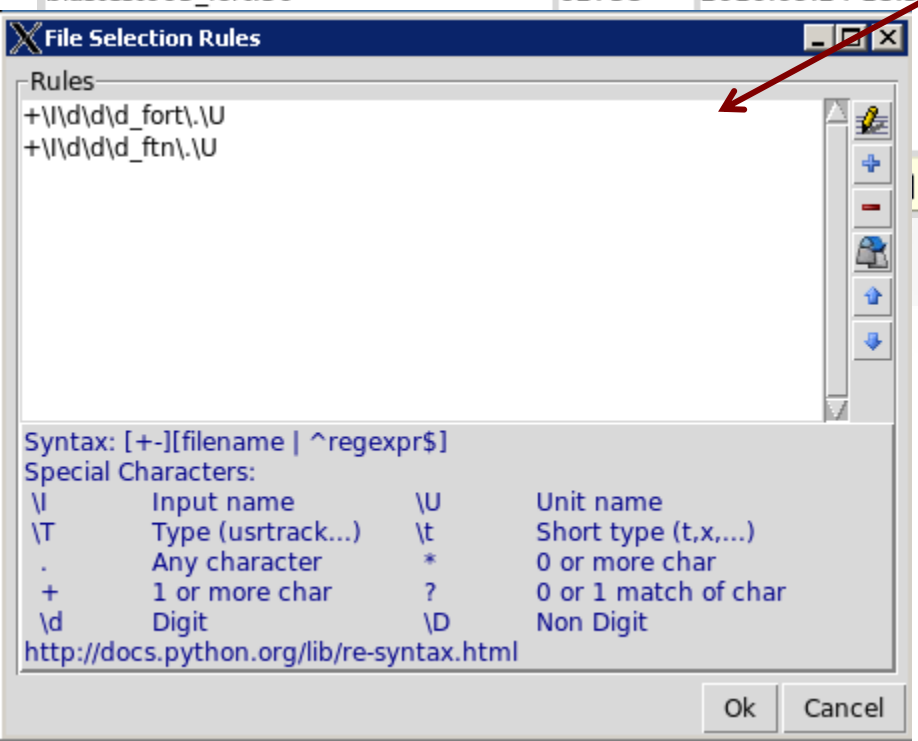
Double clicking opens:

- Files in the file Viewer
- coredumps in debugger

Right click can convert USRBIN's from formatted to unformatted

Data Processing

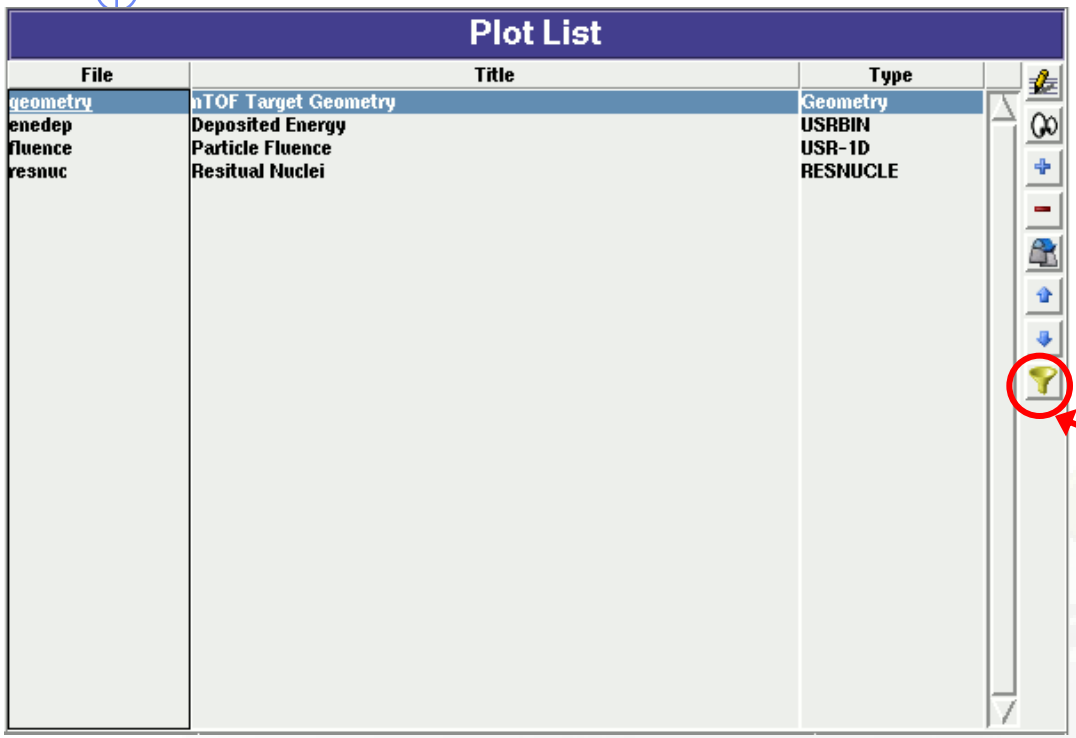




- Process all scoring BINARY output files for each Run.
- Name rules are defined in Preferences
- Automatically scan input for scoring cards
- +/- Modify file list by adding / removing items
- Dialog for editing scanning rules for files.

- Use the rules to merge from multiple runs. e.g. add a \d in the target like +\I\d\d\d\d_fort.\U
- To modify the rules for multiple scoring cards, select all Usrxxx before
- The default rules can be modified in the Preferences Dialog

Plot List

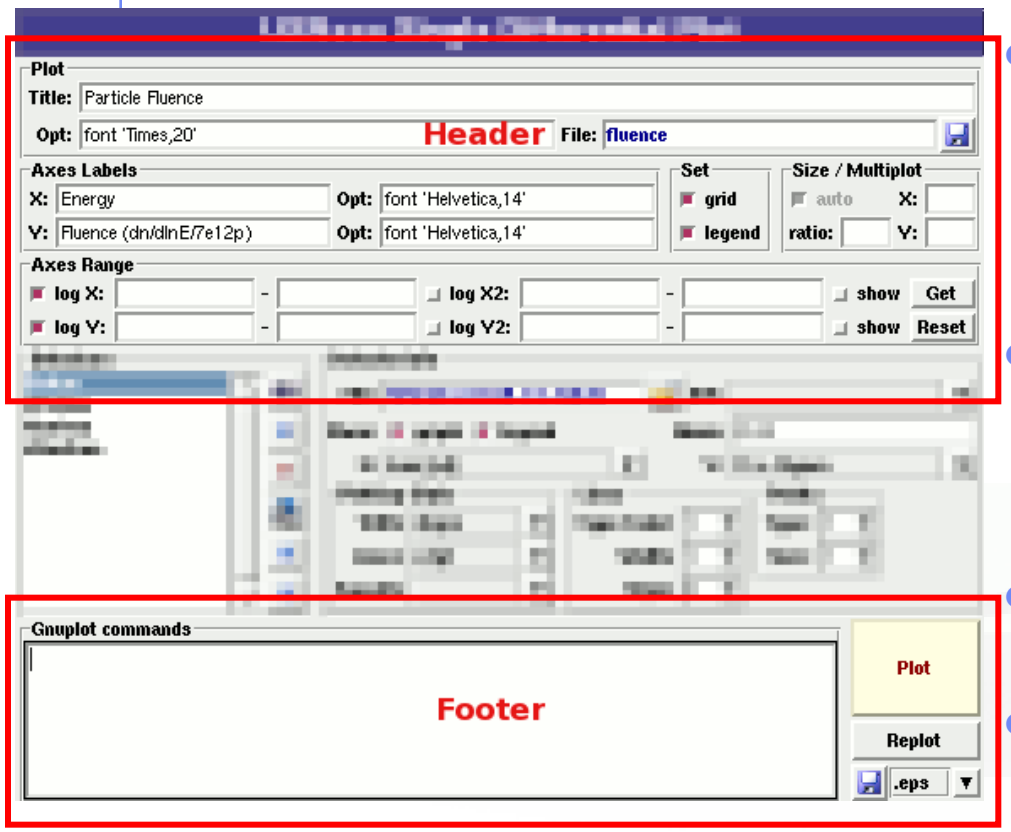


- Plots can be created in the "Plot" list frame. Either Add new plots or Clone from existing ones.
- It is important to set a unique filename for each plot. This filename will be used for every auxiliary file that the plot needs (the extension will change)
- The Filter button creates automatically one plot for each processed unit (From the default input file)
- Hit Enter or click the Edit icon to display the plotting dialog
- Fast Double click on item to open the corresponding dialog
- Slow Double click to modify the value

Plot Types

- Geometry — For geometry plots
- USRBIN — For plotting the output of USRBIN
- USR-1D — To plot single differential quantities from cards USRBDX, USRTRACK, USRCOLL, USRYIELD
- USR-2D — To plot double differential from USRBDX
- RESNUCLE — To plot 1d or 2d distributions of RESNUCLEi
- USERDUMP — To plot the output of USERDUMP. Useful for visualizing the source distribution (ToDo)

Plotting Frames



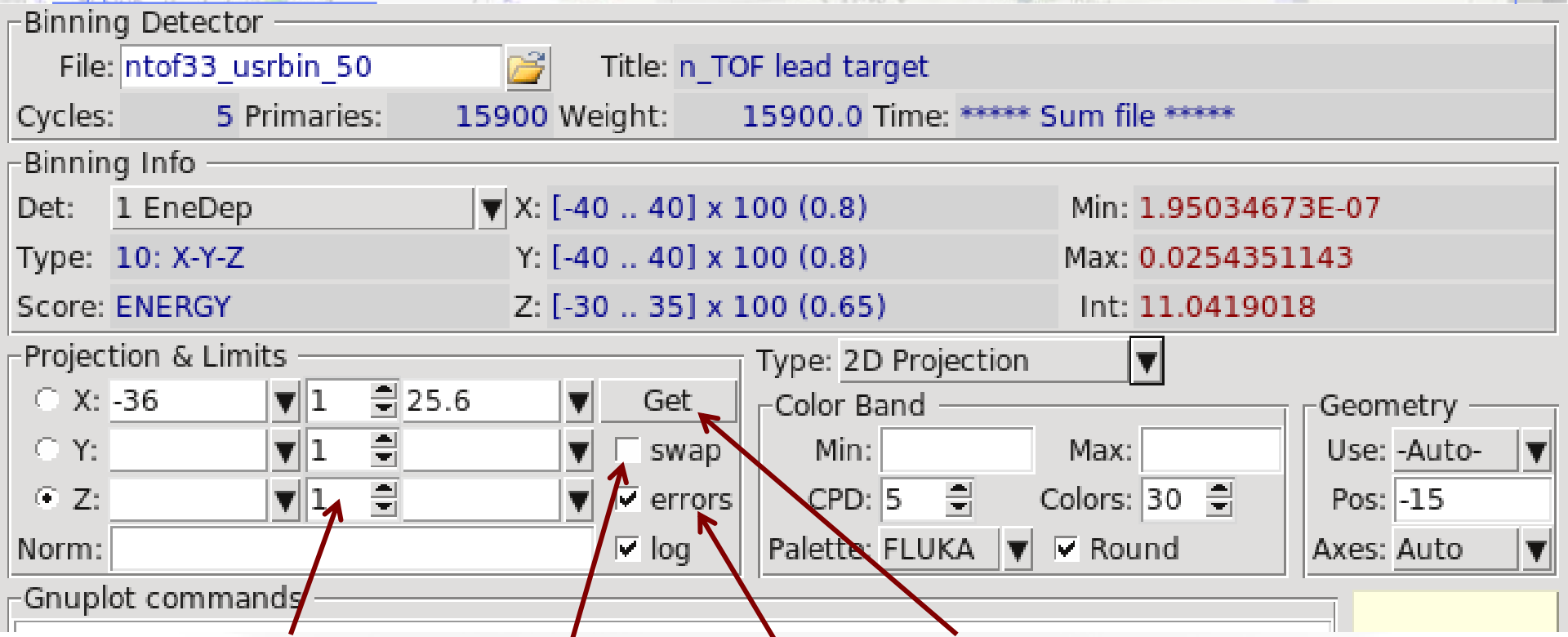
- All plot types share some common fields:
  Title + options, Filename, Axis Labels, Legends (Keys) and Gnuplot Commands.
- Plot button (Ctrl-Enter) will generate all the necessary files to display the plot, ONLY if they do not exist.
- Re-Plot will force the creation of all files regardless their state
- Check the gnuplot manual to provide additional customization commands: e.g. To change the title font to Times size=20, add in the Opt: field the command:
  font 'Times,20'

Look in the flair manual for a short reference of gnuplot commands

General Tips

- In the Configuration Dialog you can set global commands to execute before or after any plot
- The output window displays all the commands that are sent to gnuplot. As well as the errors. In case of problem always consult the output window!
- In the Gnuplot commands you can fully customize the plot by adding manually gnuplot commands:
- Special commands:
  - plot, splot — with no options, defines the order where flair should insert the plot or splot command.
  - replot <plot-cmd> — append extra plots to the one generated by flair

USRBIN Plots -1



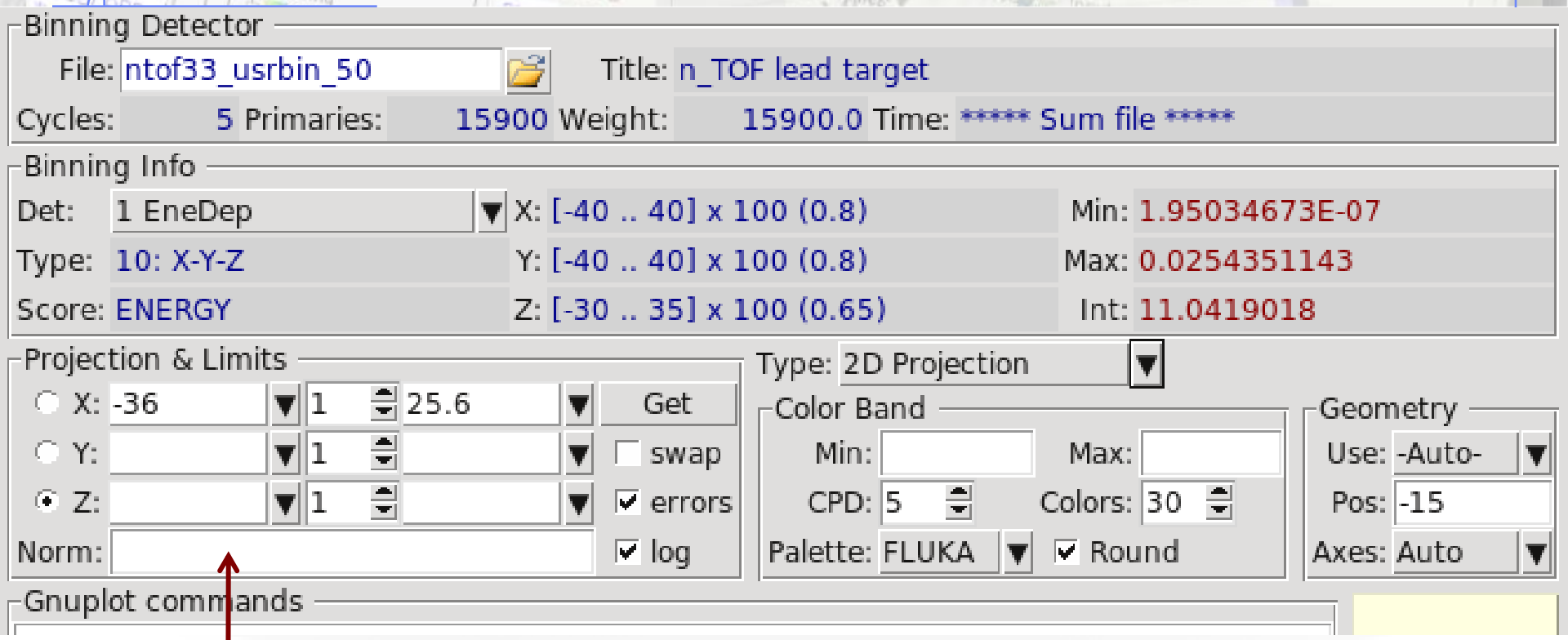
Rebinning

Swap axes

Get limits from gnuplot using right-mouse

Draw errors. (combined with log)
Correct only if one slice is used

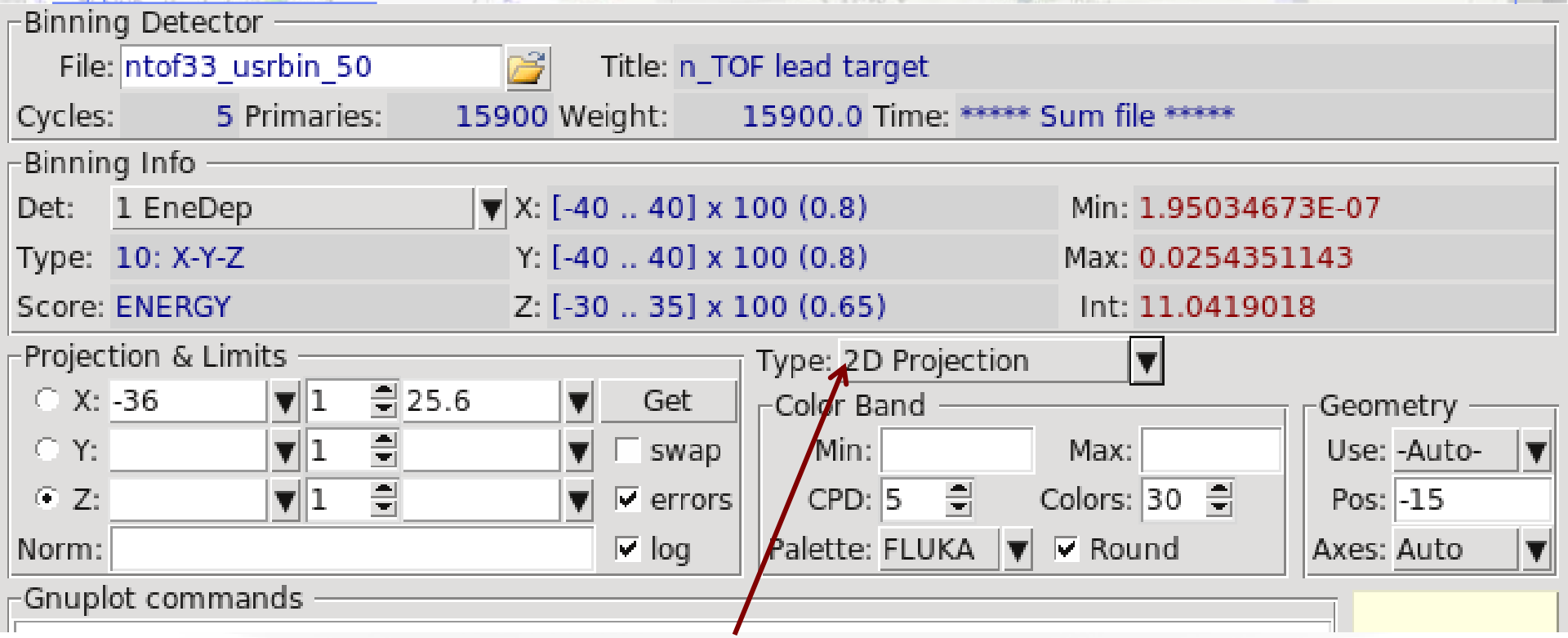
USRBIN Plots - 2



Normalization could be used as:

- number or expression evaluating in a number 65e-3/2.7
- function with x as variable. e.g E2T(x*65e-3/2.7)-293
  with the function defined in the Gnuplot commands
  E2T(x) = ((3.00629e-08*x-0.000108436)*x+1.01097)*x+311.839

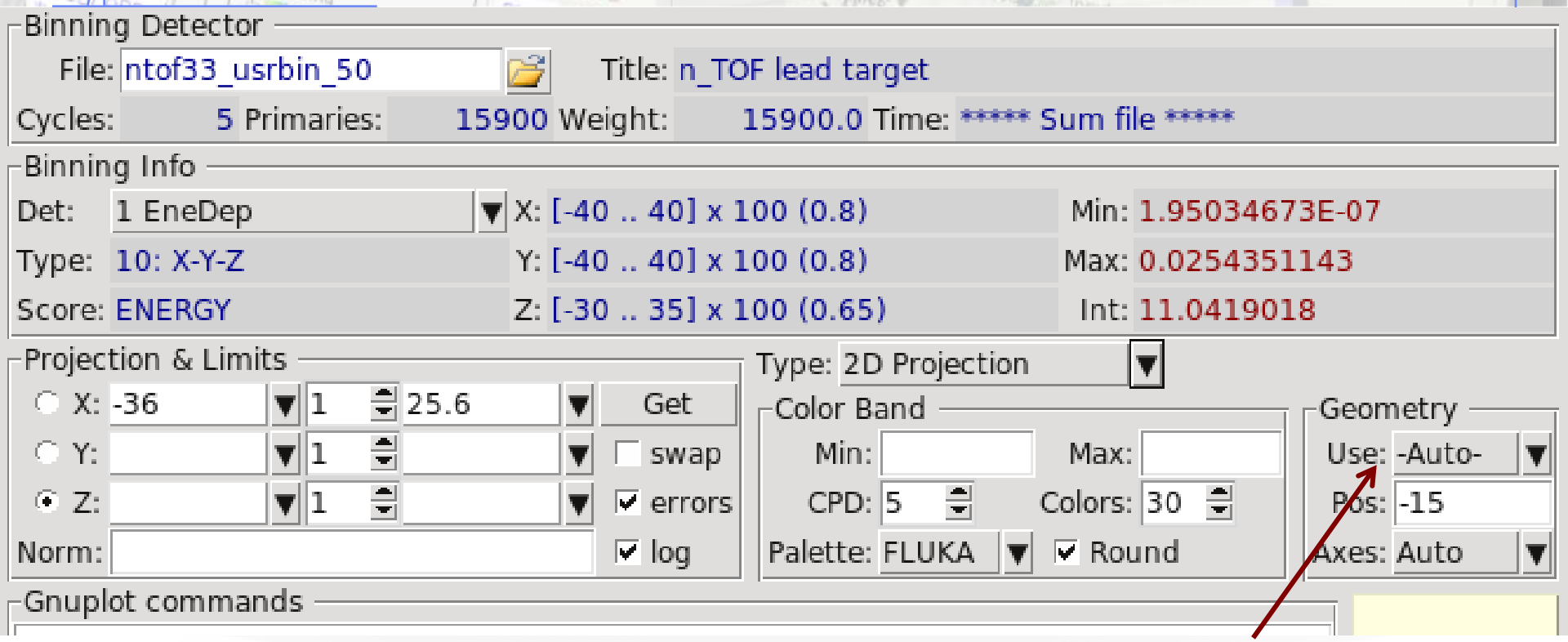
USRBIN Plots - 3



Normalization could be plotted:

- 2D projection, 1D projection
- Trace of the maximum
- Full width at half maximum

USRBIN Plots - 4



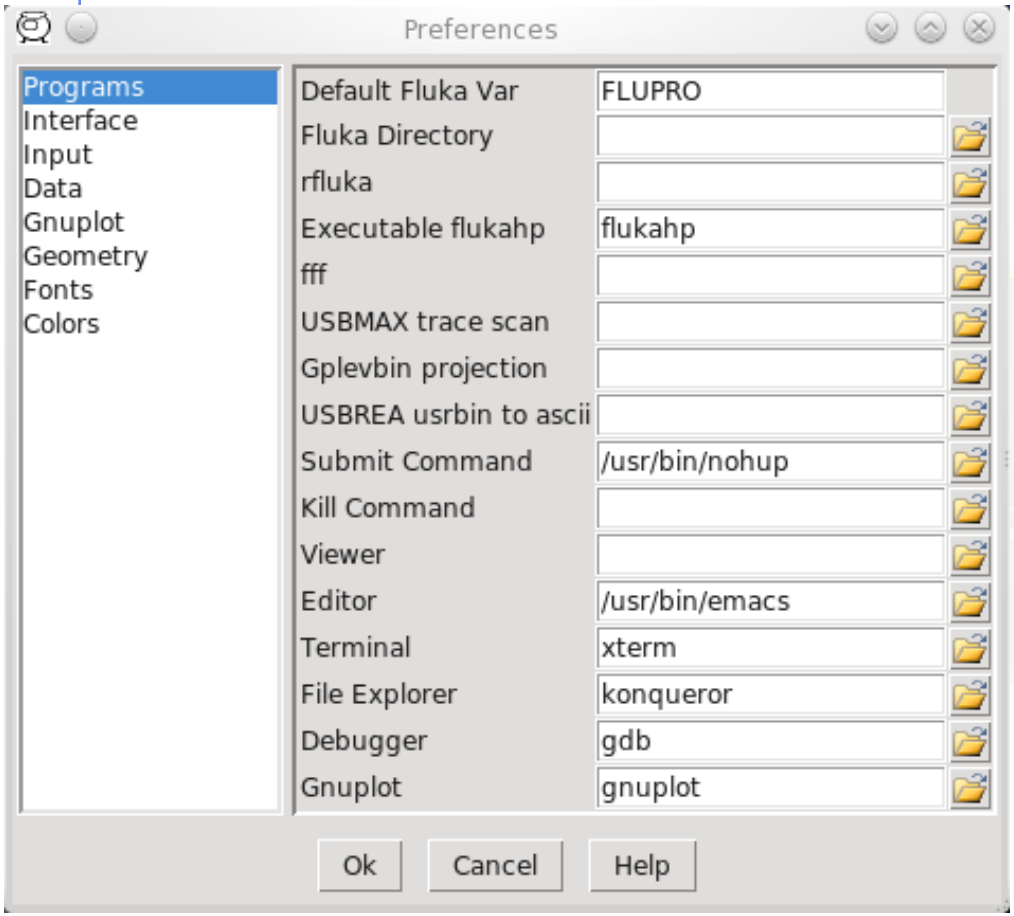
Geometry plot overlay (useful for LATTICE's):

-Auto- generates automatically from FLUKA a geometry at the middle position of the projection

otherwise you can use any existing geometry plot from the drop down list.

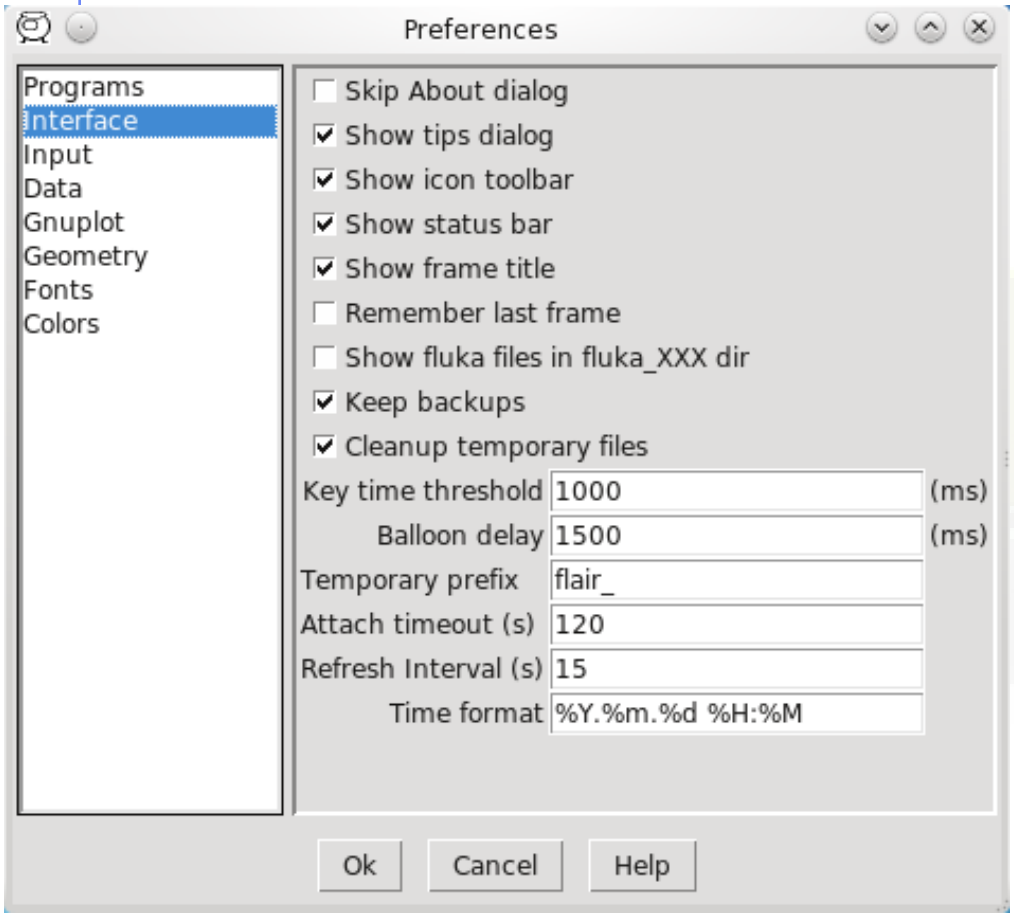
Be carefull to proerly match the axes that you are using

Configuration Dialog: Programs



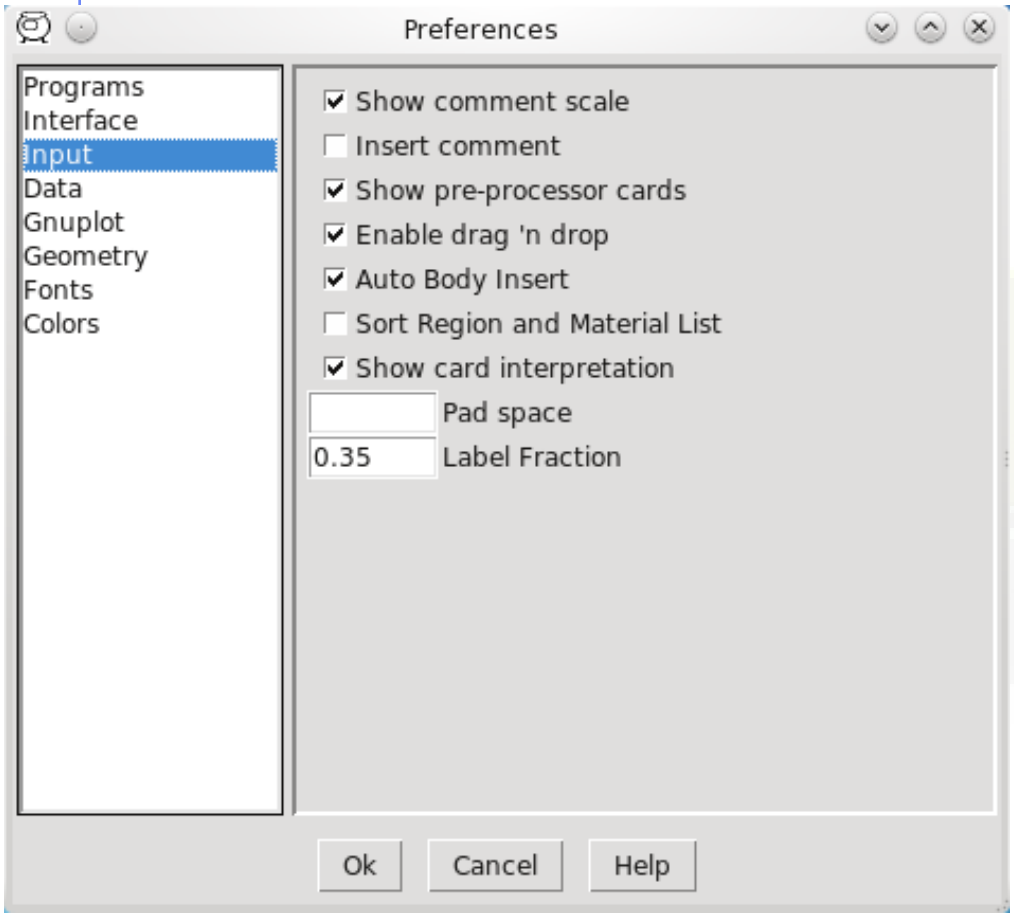
- Set FLUKA directory
- Override default programs to use
- Processing programs are in the “Data” section

Configuration Dialog: Interface



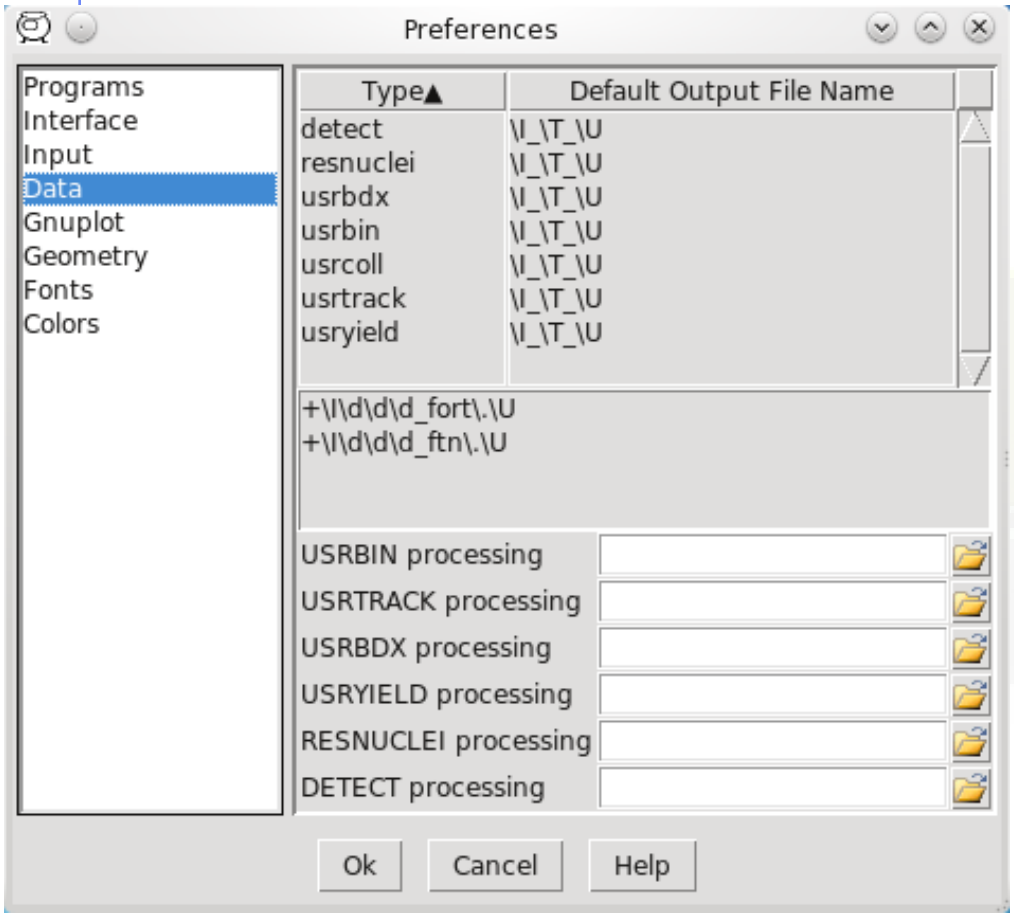
- General interface settings
- Keep backups when files are saved as (file~)
- Automatically Cleanup temporary files. Disable only if you want to inspect files after Debug or Plot when an error occurs
- Key time to reset the type-in search in listboxes
- Balloon delay time
- Time format for files (follows python&C syntax)
- Time out to attach to a running simulation
- Automatic refresh interval of information

Configuration Dialog: Input Editor



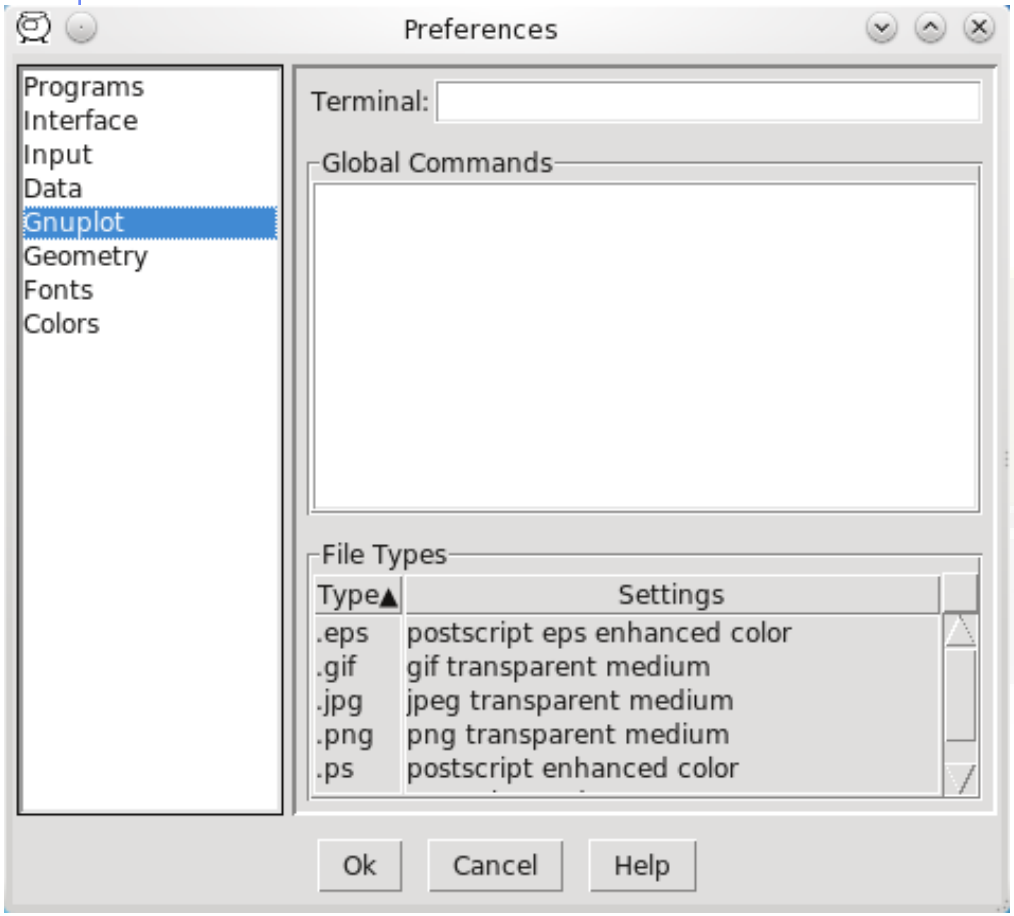
- Show alignment scale
- Automatically insert comment
- Always display preprocessor cards
- Enable drag'n'drop
- Automatic body insertion while editing the region expression
- Sort the region and material list
- Display card interpretation at the bottom of the screen

Configuration Dialog: Data



- Define how to generate the automatic filenames
  - \I will be replaced by input
  - \T by card name
  - \t by card character
    - usrbdx x
    - usrbin b
    - usrcoll c
    - usrtrack t
    - usryield y
    - resnuclei r
  - \U the abs(unit-number)

Configuration Dialog: Gnuplot



Terminal:

additional options to supply to default terminal

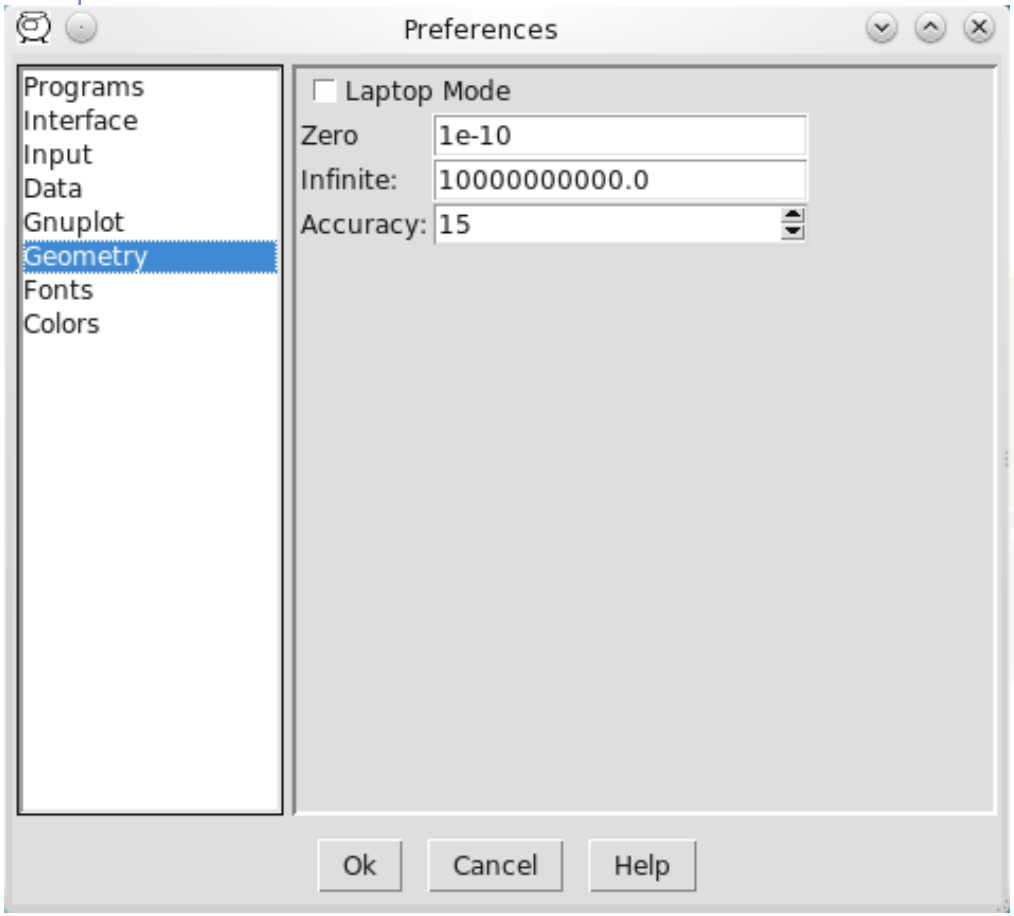
Global Commands:

gnuplot commands to be executed before any plot

File Types:

Right-click: to Add/Delete/Modify file types.

Configuration Dialog: Geometry



Laptop Mode:

check to swap middle with right mouse buttons. Middle button is used in GeometryEditor for panning, zooming, rotating etc…

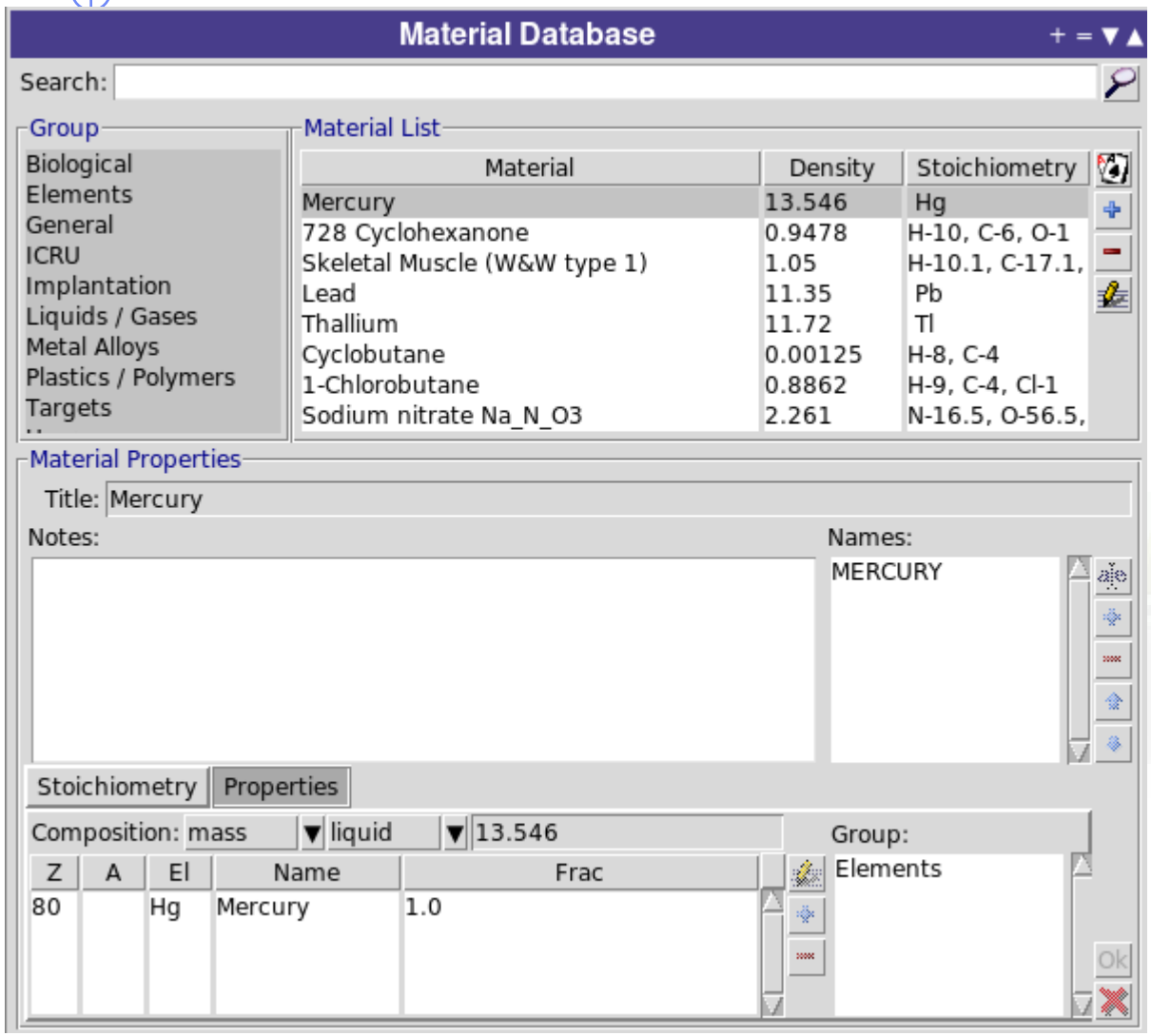
Zero:

Infinite:

Accuracy:

same as in the Bodies Transformation dialog

Materials Database



search database

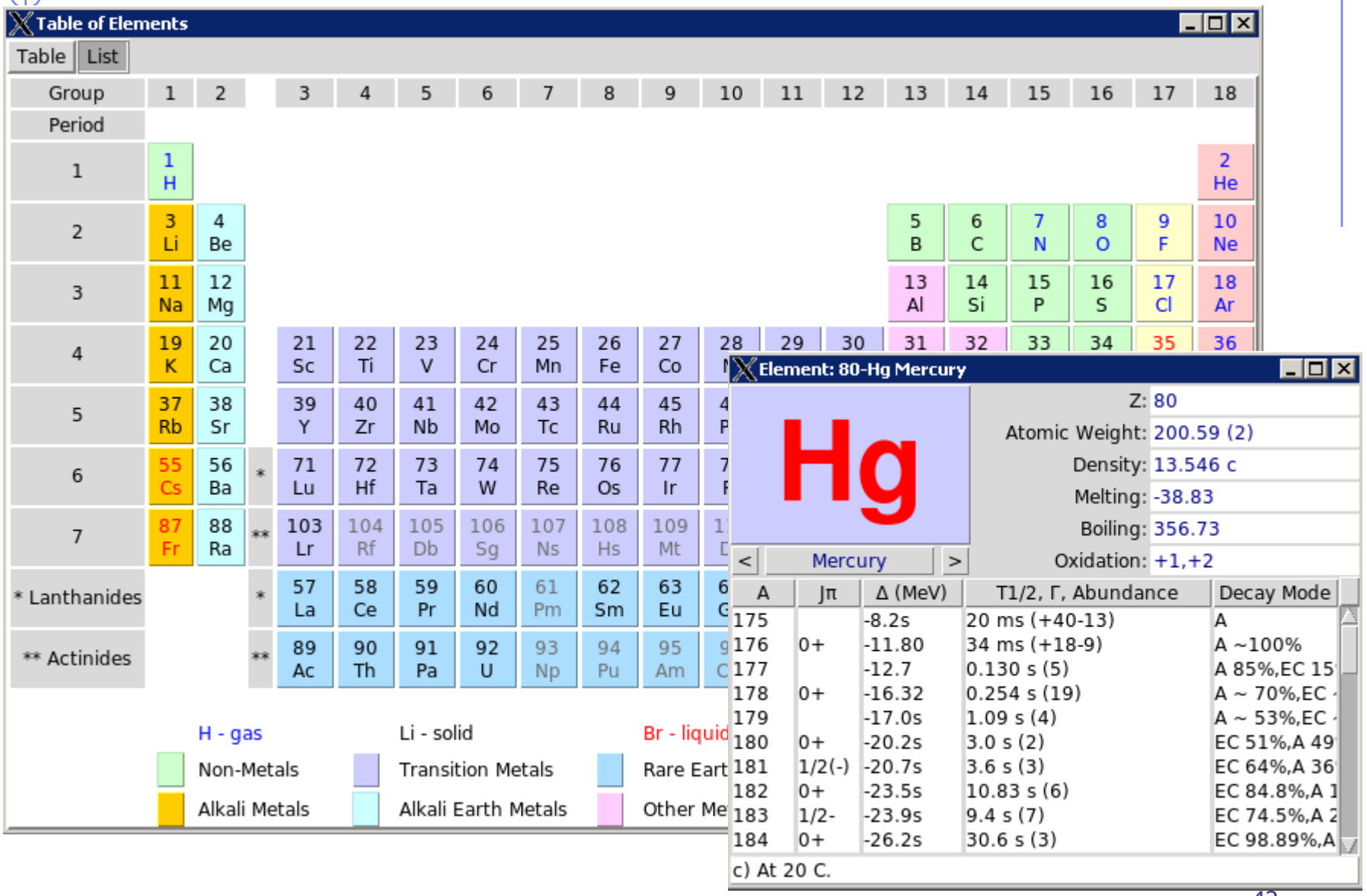
insert material to input
add/del material
edit material

add names to be used by FLUKA

Modify Stoichiometry and properties of material

WARNING: When modifying the database a local copy will be created in ~/.flair folder!!!

Periodic Table



Import / Export

Importing

- Input: merge parts or entire input file with the current
- Mcnp: import mcnp geometry into FLUKA. (experimental)

Exporting

- Gnuplot: save active plot to a gnuplot script
- Makefile: create a makefile for compiling the executable
- Mcnp: save input in MCNP format: Geometry, Materials, Importances
- Povray: save geometry into povray 3D format

Programming Interface: API

There is work presently going on to decouple the functionality from the interface, some of the basic classes can be used to input processing

file: Input.py - to manipulate input files

import Input

Input.init([database]) to initialize the database of cards

Most commonly used classes:

Card	containing the description of each card
Input	manipulating the FLUKA input file

file: Project.py - to manipulate project files

API: class Card

Constructor: Input.Card(tag, what [,comment [,extra]])

what is a list starting with what[0]=sdum

Important Methods:

setWhat(n, value)	set value to what#n
nwhats()	return number of whats
what(n)	return value of what#n
numWhat(n)	return numeric value of what#n
intWhat(n)	return integer value of what#n
clone()	return a copy of the card
setEnable(e)	enable/disable card

API: class Input

Constructor: Input.Input()

initialize the structure to hold an input file

Important Variables:

cardlist	a list with pointers to cards
cards	a dictionary with pointers to cards grouped per tag

Important Methods:

read(filename)	read input from file
write(filename)	write input to filename
addCard(card,pos)	add card to position pos (or end of file)
delCard(pos)	delete card from position pos
preprocess()	preprocess input to check for active cards
setEnable(e)	enable/disable card

API: class Project

Constructor: Project.Project()

initialize the structure to hold a project file

Important Methods:

clear()	to re-initialize project
load(filename)	load project from file filename
save([filename])	save project to filename
runCmd(run)	create run command

API: example

Read an input file and modify the random number seed

```
import Input
Input.init()
input = Input.Input()
input.read("test.inp")
try:
    rndcard = self.cards["RANDOMIZ"][0]
    rndcard.setWhat(2,5723)
except:
    print "No RANDOMIZe card found"
    sys.exit(0)
input.write("test2.inp")
```

API: .flair file structure

```
# comments
Variable: Value
Notes:
    multi-line values are terminated with
    Ctrl-L

Run: name
    ...                          Block of Run related information
    Data:
    ...                          ... Including Data processing information
    EndData
EndRun
Plot: name
...                              Plot related informations
EndPlot
```