

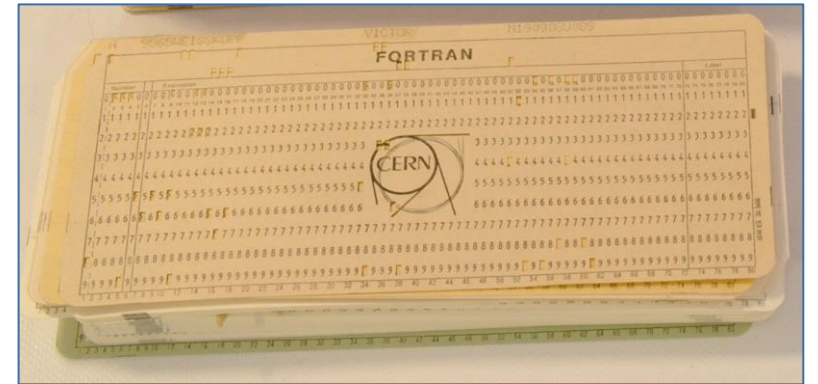


Introduction to Flair and basic input

A very basic introduction to perform your first simulation

A very short introduction

- FLUKA's story began a long time ago (1970s)...
...no graphical interfaces, input and output via text file
- Input file can be very long > 50k lines
- Input file based on “cards”: `.inp` file
- Each card has 1 name, 6 values (called WHATs), 1 string (called SDUM)

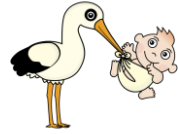


- Two examples of cards (the actual meaning is not relevant here):

BEAMPOS	4750.5	130.0	4866.5				NEGATIVE
BEAM	-0.4	0.2	5.0	1.E-4	1.E-4		ELECTRON
↑	↑	↑	↑	↑	↑	↑	↑
Card name	WHAT(1)	WHAT(2)	WHAT(3)	WHAT(4)	WHAT(5)	WHAT(6)	SDUM

A very short introduction

- In 2006, Flair was born!



FLUKA advanced interface

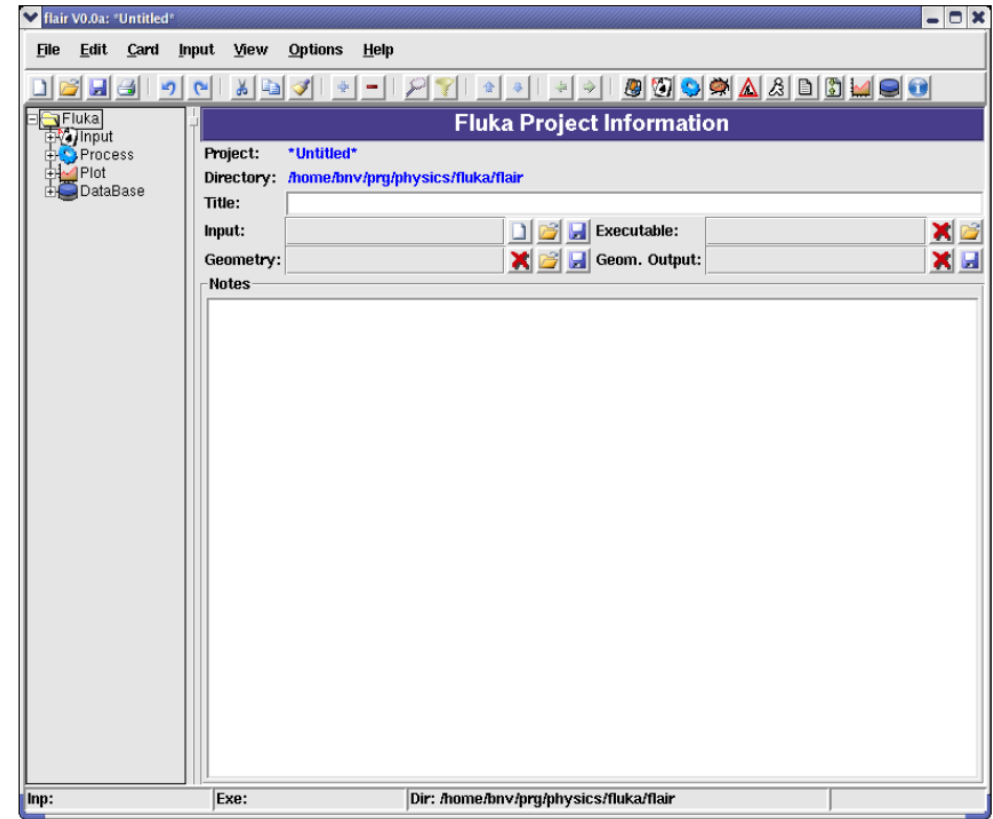
Input file creation

Geometry visualisation and construction

Simulation execution

Results visualisation

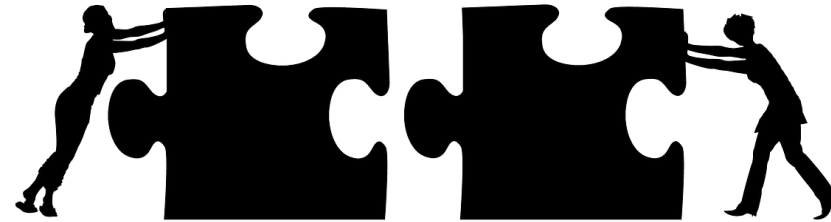
- Flair acts as an intermediate layer between the user and the input file
- It allows a user friendly editing of the FLUKA input
- Based on a `.flair` file and generates the `.inp` file that is run by FLUKA



Flair ≠ FLUKA

FLUKA & Flair

- Although strongly linked, they are two different things (`.inp` \neq `.flair`)
- FLUKA is a Monte Carlo transport code based on text files
- Flair is a graphical interface to FLUKA
- They work together but are different



- One can work with FLUKA only using text editors (for expert or long-time users)
- Flair has a lot of features very useful for expert users
- This entire course will be based on Flair

Starting Flair and basic nomenclature

- Flair can be started from the (Linux/WSL/macOS) command line, e.g.:

```
$ flair my_input.flair &
```

- Flair can also be used to open .inp files, e.g:

```
$ flair my_old_file.inp &
```

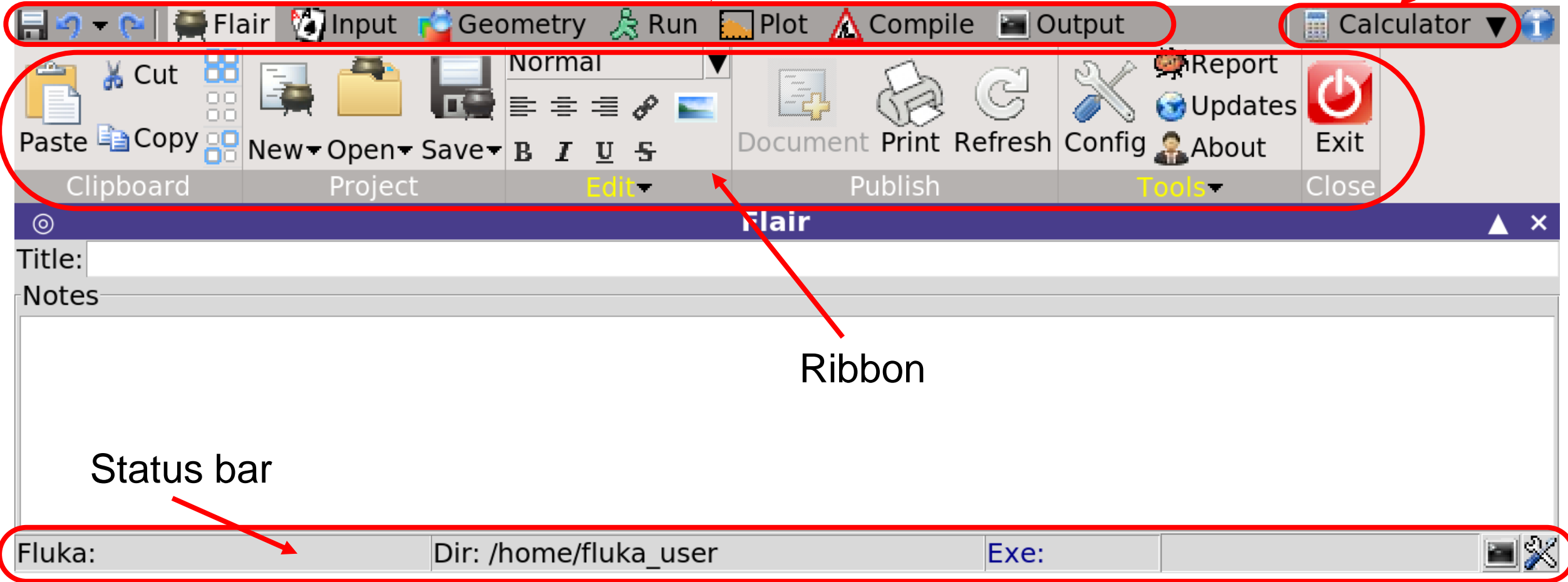
- Linux reminder about copying files:

```
$ cp exercise1.flair my_dir/.
```

Starting Flair and basic nomenclature

Configurable “Ribbon tab” or “Program tab”

Dynamic tab



What is each tab for?

The screenshot shows the Flair software interface with the following tabs circled in red: **Input**, **Geometry**, **Run**, **Plot**, **Compile**, and **Output**. Red arrows point from these tabs to the following descriptions:

- Input**: Build input and geometry
- Geometry**: Build geometry and plot results
- Run**: Run and merge results
- Plot**: Plot results
- Compile**: Compile own executable
- Output**: Visualise output files and messages

The interface also shows a menu bar with options like Cut, Copy, Paste, New, Open, Save, Document, Print, Refresh, Config, Report, Updates, About, and Exit. The status bar at the bottom displays 'Fluka: Dir: /home/fluka_user Exe:'.

The input as a text file

- Mentioned here just for completeness

.flair

```
TITLE
basic template
* Set the defaults for precision simulations
DEFAULTS PRECISIO
* Define the beam characteristics
BEAM
* Define the beam position
BEAMPOS
GEOBEGIN COMBNAME
0 0
* Black body
SPH blkbody 0.0 0.0 0.0 100000.0
* Void sphere
SPH void 0.0 0.0 0.0 10000.0
* 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
* Set the random number seed
RANDOMIZ 1.0
* Set the number of primary histories to be simulated in the run
START
STOP
-:--- basic.inp All (26,69) (Fluka)
```

.inp

.flair file includes
info & instructions
for the flair project

This course is based
on the use of Flair,
no further mention
of these text files

```
# flair project file
Version: 300
Mode: fluka
md5: c8e26fe184526e9282e8555b8fab2455
Input:
TITLE
fully-working template
#define pointless_define_1 10
#define pointless_define_2
*Set the defaults for precision simulations
DEFAULTS PRECISIO
*Define the beam characteristics
BEAM PROTON 0.8
*Define the beam position
BEAMPOS , 0. 0. -1.
GEOBEGIN COMBNAME
*Black body
SPH blkbody 0.0 0.0 0.0 100000.0
*Void sphere
SPH void 0.0 0.0 0.0 10000.0
*Cylindrical target
RCC target 0.0 0.0 0.0 0.0 0.0 10.0 5.0
END
*Black hole
REGION BLKBODY 5
+blkbody -void
*Void around
REGION VOID 5
+void -target
*Target
REGION TARGET 5
+target
END
GEOEND
*.....1.....2.....3.....4.....5.....6.....7..
ASSIGNMA , BLCKHOLE BLKBODY
ASSIGNMA , VACUUM VOID
ASSIGNMA , COPPER TARGET
USRBIN allpart 10 ALL-PART -21 6. 6. 11. -6. -6. -2. 120. 120. 130.
USRBIN edep 10 ENERGY -22 6. 6. 11. -6. -6. -2. 120. 120. 130.
*Set the random number seed
RANDOMIZ , 1.0
*Set the number of primary histories to be simulated in the run
START , 10000.
STOP
EndInput

Page: Plot

# Run information
Run: <default>
End
Run: test/test
Define: pointless_define_2=10
Start: 1000
StartRun: 1598620157
End
Run: small_prod/small
Define: pointless_define_2=10
Start: 1000
Last: 1
```

Input tab – 1: general info

- Standard looking “Windows” tab

Open and save input file

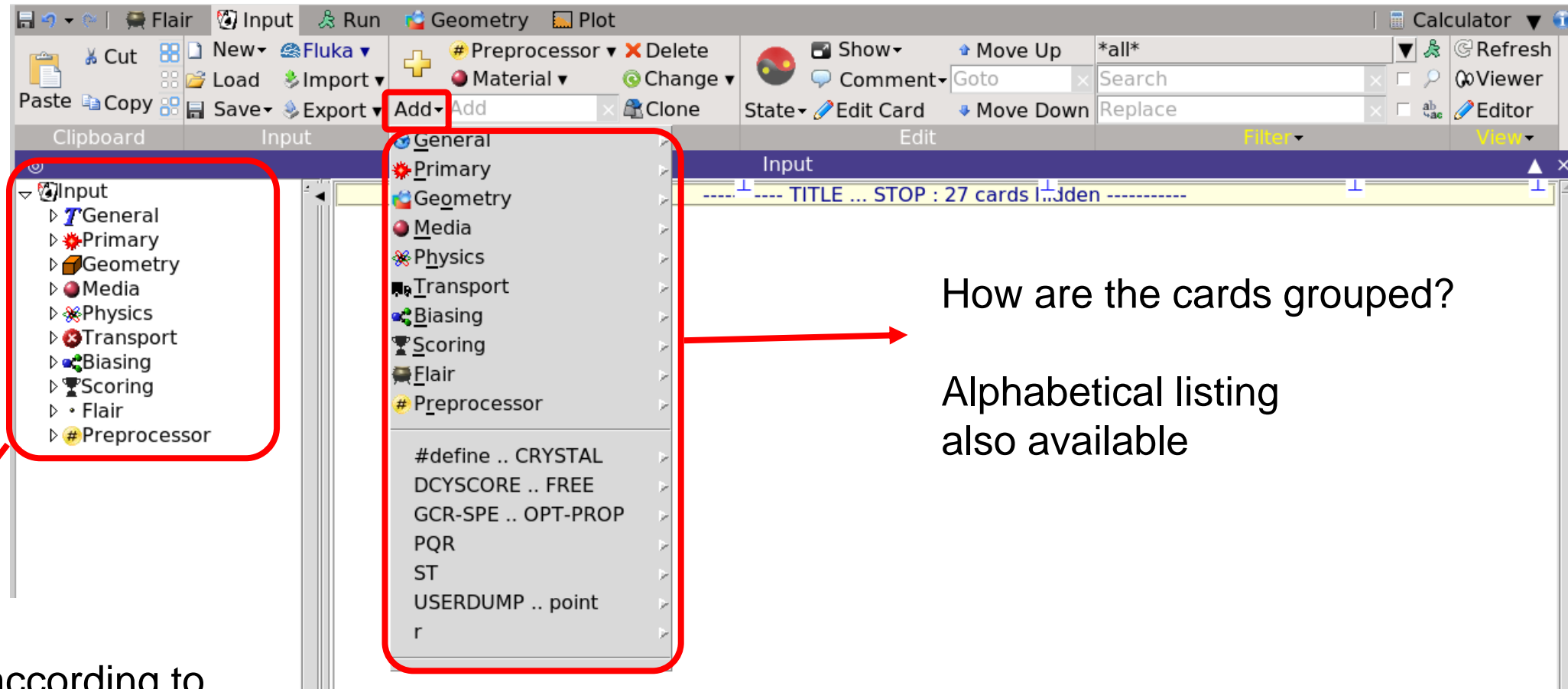
Add, clone, delete cards

Move cards up and down (with some limitations)
Recommended to use the suggested structure

Input file tree
Cards grouped according to their “field of action”

Input tab – 2: input file tree and card grouping

- Input file tree and card grouping





Input file tree
Cards grouped according to
their "field of action"

How are the cards grouped?

Alphabetical listing
also available

Input tab – 3: input file tree and card grouping

- General: defaults selection  this lecture
 - Primary: definition of the particle source
 - Geometry: definition of the geometry
 - Media: definition and assignment of “materials”
 - Physics: control specific physics processes
 - Transport: control specific transport details
 - Biasing: definition of biasing
 - Scoring: definition of estimators
 - Preprocessor: definition of preprocessor instructions
 - Flair: definition of flair add-ons for visualisation
- 
- dedicated lectures

Input tab – 4: General cards

TITLE

START

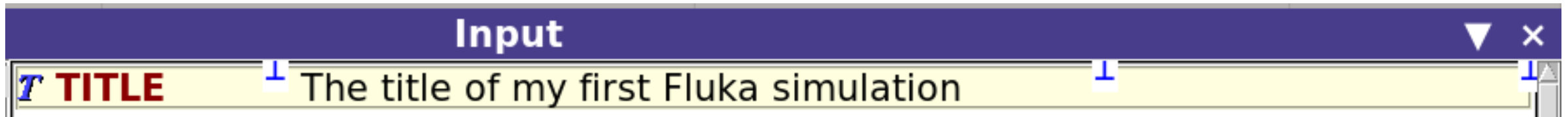
STOP

RANDOMIZe

DEFAULTS

TITLE

- Not a mandatory card
- Allows to assign a title to the simulation
- The title is printed in the output files



Input tab – 5: Mandatory cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

START

- Listed among the “Primary” cards
- It is a **mandatory** card (lack of it will result in an error)
- Allows to set the number of primary histories to be simulated
- Allows to set other parameters for advanced use

Set the number of primary histories to be simulated in the run

 **START**

No.: 10000.

Time:

Core: ▼

Report: default ▼

Input tab – 6: General cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

STOP

- Stop the execution of the program
- Not really mandatory (program stops at the end of the input)
- Can become handy for debugging purposes



STOP

Input tab – 7: General cards

TITLE

START

STOP

RANDOMIZ_e

DEFAULTS

RANDOMIZ

- Allows to initialise different random sequences
- For debugging purposes, the “random seed” must be the same
- Different “random seeds” are required in order to differentiate histories
- Flair takes care of the “random seeds” when spawning runs (see later)

Set the random number seed

 **RANDOMIZ**

Unit: 01 ▼

Seed: 123

Input tab – 8: General cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

DEFAULTS

- Allows to select the physics defaults (list of predefined defaults available)
- Physics defaults can be overridden with specific cards
- Can be preceded only by the **TITLE** and **GLOBAL** cards
- Given the progress over time in computer power, it is a reasonable approach to:
 - always select the most detailed physics defaults: **PRECISIO**
 - depending on the needs of the problem, override specific defaults

Set the defaults for precision simulations

 **DEFAULTS**

: PRECISIO ▼

Input tab – 9: Expressions

- It is possible to specify values using expressions
- Possible to parametrize input and launch runs with different values
- Fields starting with “=” will be evaluated by flair, e.g.:

```
BEAMPOS      x: =2+10*length
```

- Expressions are stored in the `.flair` file
- Expressions are also stored in the `.inp` file as comments, e.g.:

```
!@what.1=2+10*length
```

- The cards in the `.inp` file contain the evaluated values

Do not change by hand, they will be overwritten by Flair!!!

Input tab – 10: Expressions

- See manual for details (see later for the manual)
- Useful predefined quantities
 - Units, e.g.: *MeV, mm, ms...* (warning: only treated as conversion factors)
 - Constants: *fwhm, c, qe...*
 - Particle masses: *Mp, Me...*
- All common mathematical functions: *sin(x), cos(x), exp(x)...*
- Some physics functions
- Card reference functions
 - *what(n)*
 - *body(name,what)*
 - *card(tag,sdum/id, what)*

Input tab – 11: “Reg:”, “to Reg:”, “Step:”

- Recurring feature in FLUKA
- Not just regions:
 - Regions
 - Materials
 - Detectors
 - Lattices
 - Particles
 - ...

✂ EMFCUT

✂ EMFCUT

Fudgem:
 ASSIGNMAT

 BIASING

Opt: ▼
 AUXSCORE

Delta Ray: ▼

 LATTICE ▼

 LOW-PWXS
 db: ▼

 OPT-PROD

 PHOTONUC
 E>0.7GeV: off ▼



 EMF-BIAS
 Old brems.: off ▼
 Compton: off ▼

 LAM-BIAS
 Mat: ▼

Type: transport ▼			
e-e+ Threshold: Total ▼	e-e+ E:		γ:
Reg: ▼	to Reg: ▼		Step:
Type: PROD-CUT ▼			
e-e+ Threshold: Total ▼	e-e+ E:		γ:
Mat: ▼	to Mat: ▼		Step:
Mat: ▼	Reg: ▼	to Reg: ▼	Field: ▼
Mat(Decay): ▼	Step:		Imp:
Type: ▼	RR:		Step:
Reg: ▼	to Reg: ▼		Set: ▼
Type: ▼	Part: ▼		Isomer: 0
Z: 0	A: 0		
Det: ▼	to Det: ▼		Step:
Reg: ▼	to Reg: ▼		Step:
Lat: ▼	to Lat: ▼		Step:
Mat: ▼	to Mat: ▼		Step:
IAZ: ▼	S(α,β): ▼		T:
Type: ▼			
Mat: ▼	to Mat: ▼		Step:
Type: ▼			All E: off ▼
Δ resonance: off ▼	Quasi D: off ▼		Giant Dipole: off ▼
Mat: ▼	to Mat: ▼		Step:
Type: ▼	Ethr e-e+:		Ethr γ:
Bremsstrahlung: off ▼	Pair Prod.: off ▼		e+ ann @rest: off ▼
Bhabha&Moller: off ▼	Photo-electric: off ▼		e+ ann @flight: off ▼
Reg: ▼	to Reg: ▼		Step:
Type: ▼	x mean life:		x λ inelastic:
Part: ▼	to Part: ▼		Step:



Input tab – 12: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 1: “CARBON” is assigned to all regions from “region_1” to “region_4”

 ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step:	to Reg: region_4 ▼ Field: ▼
 ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step: 1	to Reg: region_4 ▼ Field: ▼

Input tab – 13: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 2: “CARBON” is assigned to “region_1” and “region_3”

 ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step: 2	to Reg: region_3 ▼ Field: ▼
 ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step: 2	to Reg: region_4 ▼ Field: ▼

Input tab – 14: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 3: activate “PHOTONUC” (exact meaning not relevant here) for “CARBON”, “OXYGEN”, “ALUMINUM”, “COPPER”, etc.

PHOTONUC	Type: ▼	Quasi D: off ▼	All E: off ▼
E>0.7GeV: off ▼	Δ resonance: off ▼	to Mat: ▼	Giant Dipole: off ▼
	Mat: CARBON ▼		Step: 2
REGION region_1		Neigh:	
expr: +reg1		CARBON	
REGION region_2		Neigh:	
expr: +reg2		NITROGEN	
REGION region_3		Neigh:	
expr: +reg3		OXYGEN	
REGION region_4		Neigh:	
expr: +reg4		MAGNESIU	
ASSIGNMAT	Mat: CARBON ▼	Neigh:	
	Mat(Decay): ▼	ALUMINUM	
	Mat: CARBON ▼	IRON	
		COPPER	
		SILVER	Reg: ▼
		SILICON	Field: ▼
		GOLD	Reg: ▼

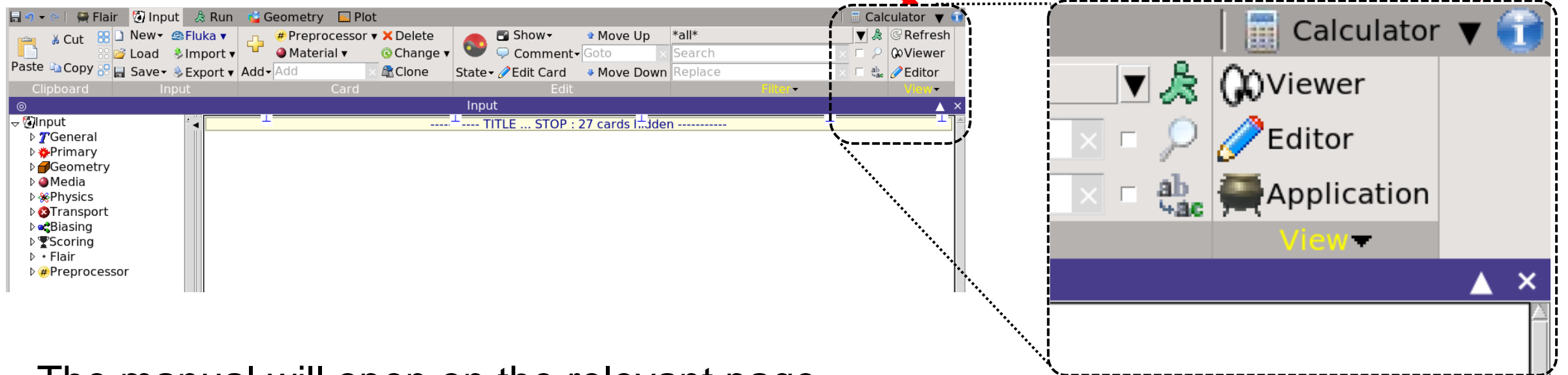
Input tab – 15: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- The same concept applies to all other cases:
materials, particles, lattices, etc.
- Special variables:
 - @LASTREG i.e. the last defined region
 - @LASTMAT i.e. the last defined material
 - @LASTPART i.e. the last pre-defined particle

as of today: AOMEGAC0 ($\overline{\Omega_c^0}$)

The FLUKA manual

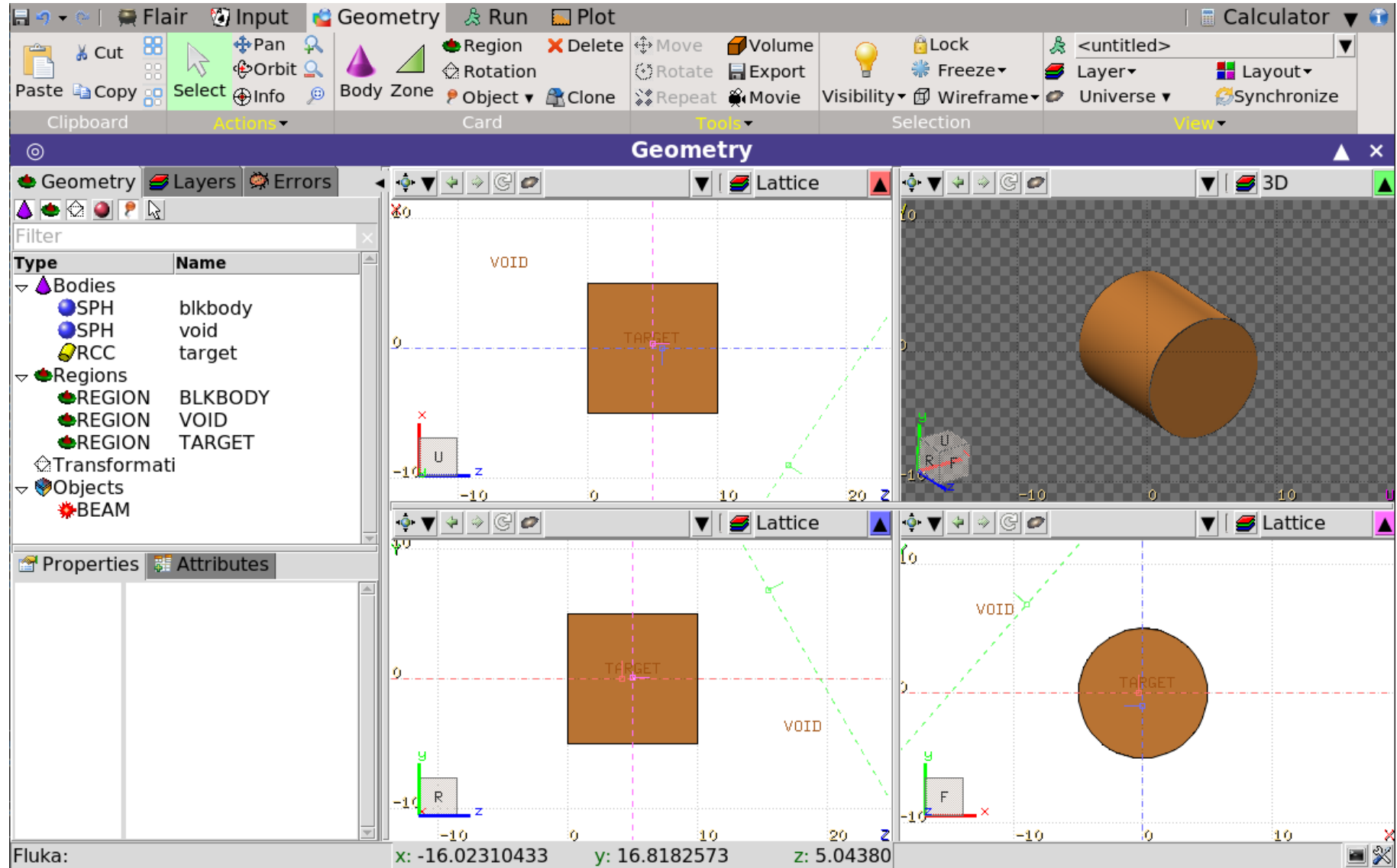
- Can be accessed using the F1 key
- Can be accessed clicking on the “info” button



- The manual will open on the relevant page
- The manual is also available on the FLUKA web page fluka.cern

Geometry tab – 1

- Visualise and edit geometry
- Plot results
- Dedicated lectures



Geometry tab – 2

- Viewports automatically refreshed when input is changed

Layout management

The screenshot shows the FLUKA Geometry tab interface. The top menu bar includes options like Flair, Input, Geometry, Run, and Plot. Below it is a toolbar with various tools such as Cut, Paste, Copy, Select, Pan, Orbit, Info, Body Zone, Region, Rotation, Object, Clone, Repeat, Move, Rotate, Repeat, Volume, Export, Movie, Visibility, Wireframe, Lock, Freeze, Layer, Universe, Synchronize, and Transform. A red box highlights the 'Layout' dropdown menu in the top right corner, with an arrow pointing to the text 'Layout management'.

On the left side, three red arrows point to specific UI elements:

- Filters:** Points to the 'Geometry', 'Layers', and 'Errors' buttons.
- Objects Listbox:** Points to a list of objects with columns for 'Type' and 'Name'. The list includes:

Type	Name
SPH	blkbody
SPH	void
RCC	target
Regions	
REGION	BLKBODY
REGION	VOID
REGION	TARGET
Transformati	
Objects	
BEAM	
- Properties & Attributes Listbox:** Points to a table of properties for the selected 'target' object:

Properties	Attributes
name	target
comment	Cylindrical target
type	RCC
x	0.0
y	0.0
z	0.0
Hx	0.0
Hy	0.0
Hz	10.0
R	5.0
@xmid	0.0

The main area contains three viewports: a top-left 2D view, a top-right 3D view, and a bottom-left 2D view. The 3D view shows a brown cylindrical object with a green circle and a blue circle overlaid. The bottom-left view shows the same object from a different perspective. The status bar at the bottom displays coordinates: x: 4.831159629, y: 0, z: 18.22692971.

Geometry tab – 3

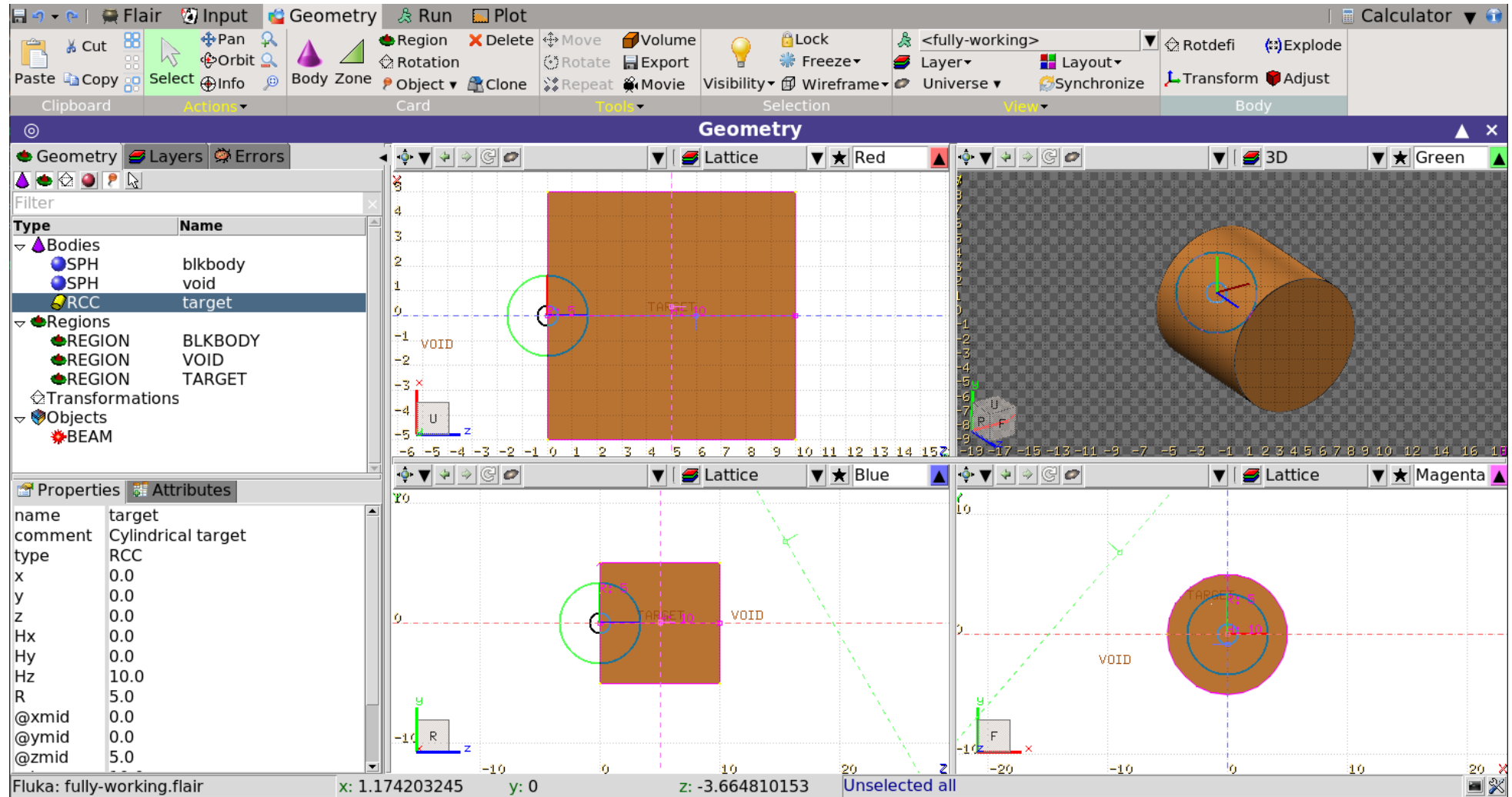
The screenshot displays the FLUKA Geometry tab interface. The top toolbar contains various tools like Cut, Paste, Copy, Select, Pan, Orbit, Info, Body Zone, Region, Delete, Move, Volume, Rotate, Repeat, Lock, Freeze, Wireframe, Visibility, Universe, Layer, Layout, Synchronize, Rotdefi, Explode, Transform, and Adjust. The main workspace is divided into four viewports: Red, Green, Blue, and Magenta, each showing a different perspective of a cylindrical target. The Red viewport shows a side view of the cylinder, the Green viewport shows a perspective view, the Blue viewport shows a top-down view, and the Magenta viewport shows a bottom-up view. The left-hand panel contains a tree view with categories like Bodies, Regions, Transformations, and Objects, and a Properties panel for the selected 'target' object. The status bar at the bottom shows the current coordinates and selection status.

Type	Name
SPH	blkbody
SPH	void
RCC	target
REGION	BLKBODY
REGION	VOID
REGION	TARGET

Property	Value
name	target
comment	Cylindrical target
type	RCC
x	0.0
y	0.0
z	0.0
Hx	0.0
Hy	0.0
Hz	10.0
R	5.0
@xmid	0.0
@ymid	0.0
@zmid	5.0

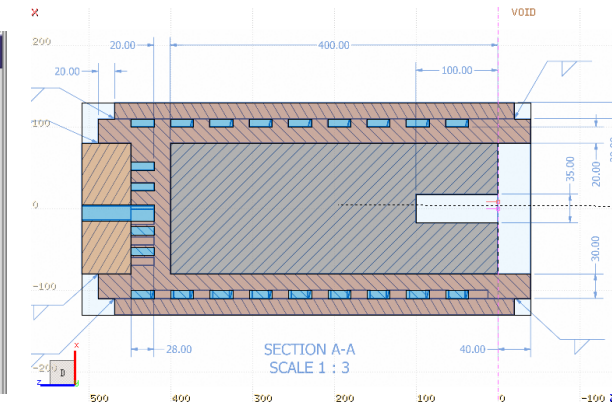
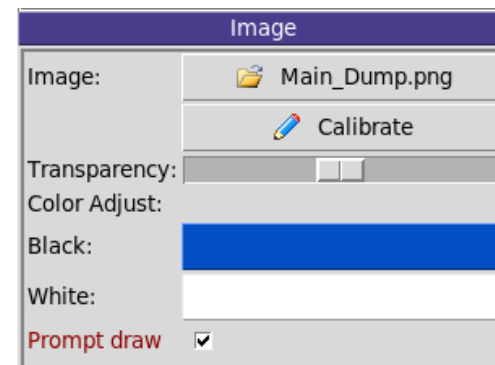
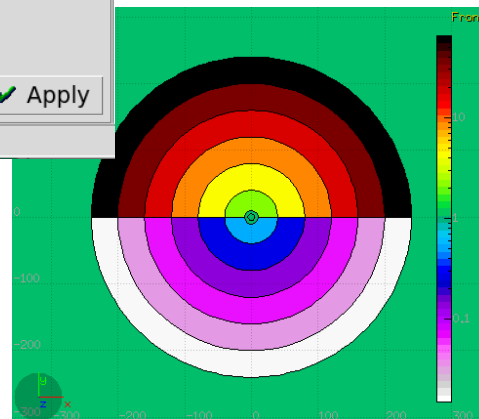
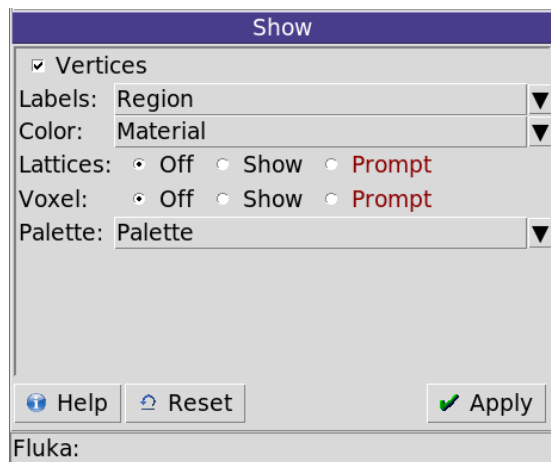
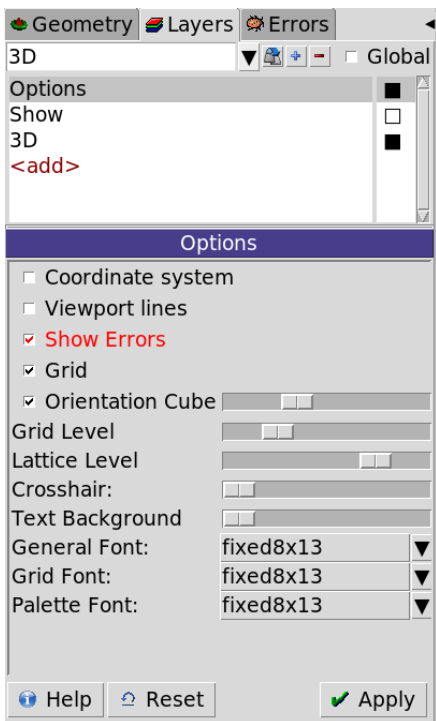
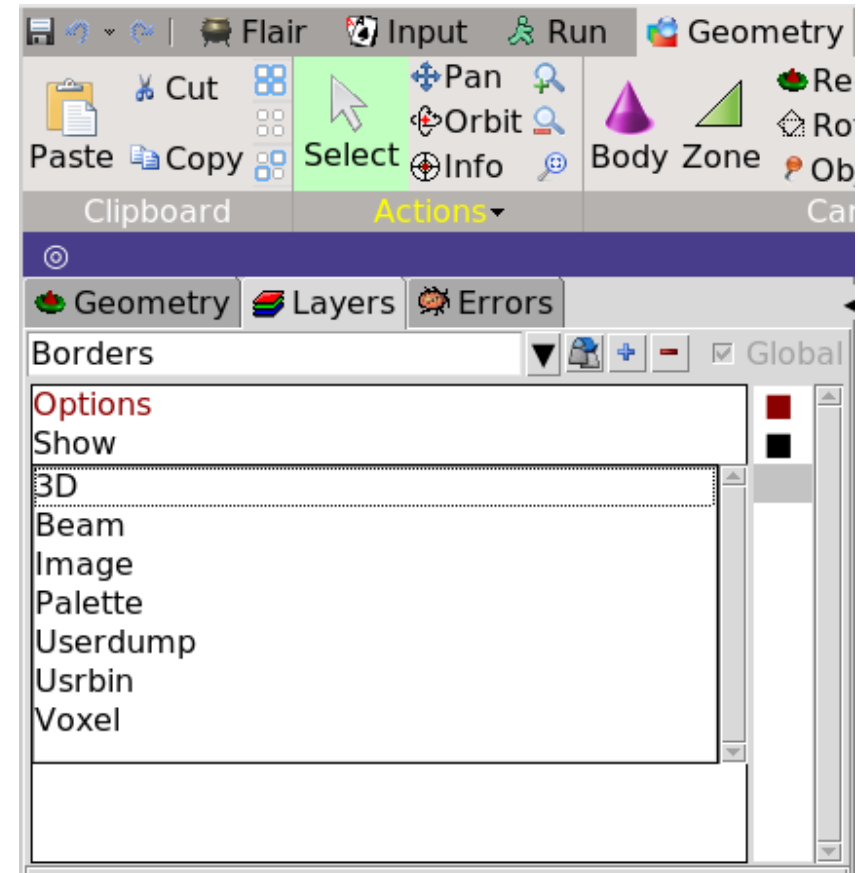
Geometry tab – 4

- Possible to navigate with mouse and keyboard (see dedicated lecture)



Geometry tab – 5

- Possible to add layers for better visualisation:
 - Appearance (fonts, etc.)
 - Scoring (see Scoring I lecture)
 - Special quantities (e.g. region importance)
 - Background images (to help building geometry)



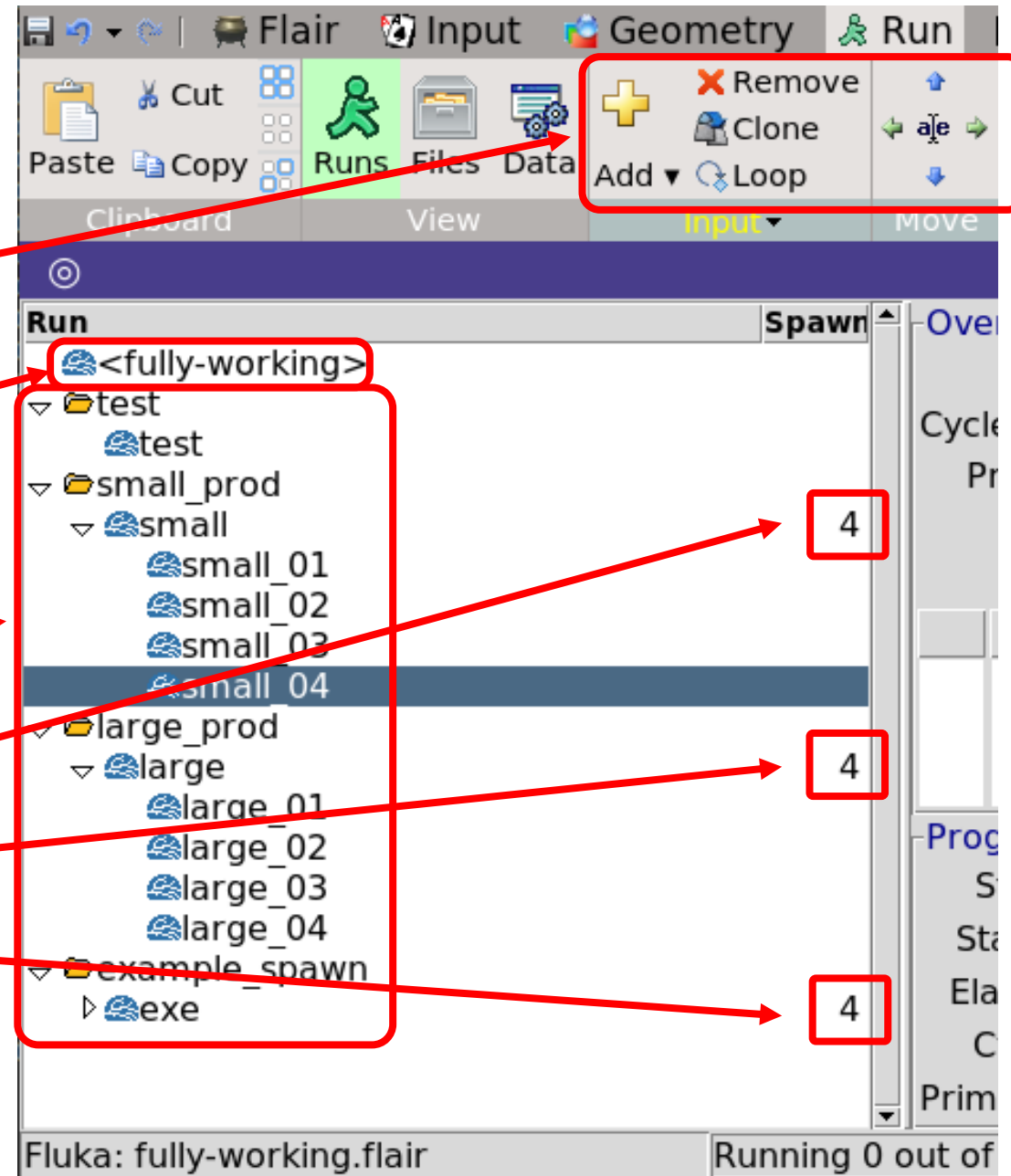
Run tab

- 3 views: “Runs”, “Files”, and “Data”

• Shared listbox:
List of simulations associated with the project

Run tab – Runs view – 1

- Management of the various simulations
- Basic input file of the Flair project
- Different simulations associated with the Flair project
- Number of spawns



Run tab – Runs view – 2

Cycle control: how many cycles to run, starting from which cycle

Executable

Number of primaries

Override

Title:

Cycles From: 0 No: 5 To: 5

Primaries: 1000.0 Time: 0.0 Rnd: 0

Mode: Exe: my.x

Defines: Default Defines

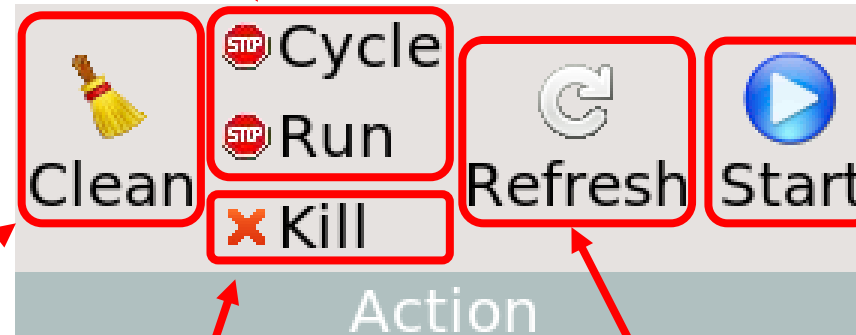
	Name	Value
<input type="checkbox"/>	pointless_define_1	
<input checked="" type="checkbox"/>	pointless_define_2	10.

#defines

Dedicated lecture

Run tab – Runs view – 3

Cleanly stop the cycles/runs currently running



Start a simulation

Remove files from a previous simulation

Refresh the progress field

Kill the current simulations

Run tab – Runs view – 4

The screenshot shows the 'Run' window in FLUKA. On the left is a tree view with folders like 'test', 'small_prod', 'large_prod', and 'example-spawn'. The 'small' folder under 'small_prod' is expanded, showing 'small_01' through 'small_04'. 'small_01' is selected. Below the tree is a 'Progress' section for the selected run. It displays: Status: Running; Started: 2023.10.25 16:00:22; Elapsed: 15.8 s; Input: small_prod/small_01; ETA: 2023.10.25 16:13:37; Cycle: 3m 1s; Dir: fluka_31164; Time/prim: 197 ms; Run: 12m 56s. Below this is a 'Cycles' progress bar (green and yellow) and 'Primaries' progress bar (green). At the bottom, it says 'Running 4 out of 14'.

Input file actually running

Temporary directory

Status

Estimated end of the simulation

Time per primary

Overall job progress bar

Cycle progress bar

Time since the start of the cycle

Time until the end of the cycle

Time until the end of the simulation

Run tab – Runs view – 4

The screenshot shows the 'Run' tab interface. On the left is a tree view of the job structure:

- <fully-working>
- test
- small_prod (4)
- small (4)
- small_01 (selected)
- small_02
- small_03
- small_04
- large_prod (4)
- large (4)
- large_01
- large_02
- large_03

The main area displays the progress for the selected run 'small_01':

Status: Running	Input: small_prod/small_01	Dir: fluka_20680
Started: 2025.05.07 09:52:08	ETA: 2025.05.07 09:57:25	Time/prim: 78.1 ms
Elapsed: 9.38 s [1m 48s]	Cycle: 1m 8s	Run: 5m 5s
Cycles:	Current: 2 [5] Completed: 20%	
Primaries:	Current: 121 [1000] Completed: 12%	

At the bottom, it shows 'Running 4 out of 14'.

Input file actually running

Temporary directory

Status

Estimated end of the simulation

Time per primary

Overall job progress bar

Cycle progress bar

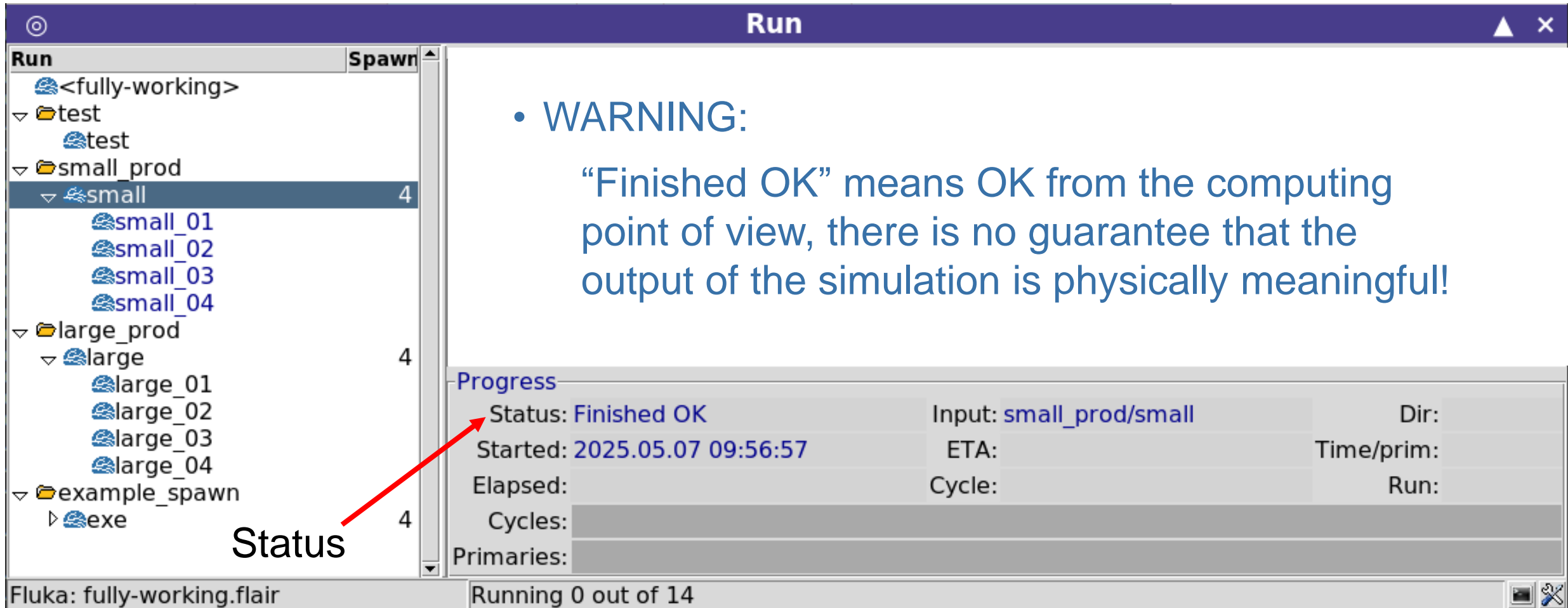
Time since the start of the cycle/run

Time until the end of the cycle

Time until the end of the simulation

Run tab – Runs view – 5

- At the end of the simulations...



The screenshot shows a software interface with a 'Run' window. On the left, a tree view displays the hierarchy of simulation runs: <fully-working>, test, small_prod, small, large_prod, large, and example_spawn. The 'small' folder under 'small_prod' is selected, showing sub-items small_01 through small_04. A red arrow points from the word 'Status' in the bottom left to the 'Status: Finished OK' entry in the 'Progress' window on the right. The 'Progress' window displays the following information:

Progress			
Status:	Finished OK	Input:	small_prod/small
Started:	2025.05.07 09:56:57	ETA:	
Elapsed:		Cycle:	
Cycles:		Run:	
Primaries:			

At the bottom of the interface, the status bar shows 'Fluka: fully-working.flair' and 'Running 0 out of 14'.

- WARNING:

“Finished OK” means OK from the computing point of view, there is no guarantee that the output of the simulation is physically meaningful!

After running – 1

- Content of the working directory

```
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ ls  
fully-working.flair  fully-working.inp  my.exe  small_prod  tutorial.flair  
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ _
```

- Content of the working sub-directory

```
fluka_user:/home/fluka_user$ cd small_prod/  
fluka_user:/home/fluka_user/small_prod$ ls  
ransmall_01001  ransmall_04005      small_01004_fort.21  small_02003_fort.21  small_03002_fort.21  small_04001_fort.21  
ransmall_01002  ransmall_04006      small_01004_fort.22  small_02003_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01003  small_01.inp        small_01005.err      small_02004.err      small_03003.err      small_04002.err  
ransmall_01004  small_01.out        small_01005.log      small_02004.log      small_03003.log      small_04002.log  
ransmall_01005  small_01001.err     small_01005.out      small_02004.out      small_03003.out      small_04002.out  
ransmall_01006  small_01001.log     small_01005_fort.21  small_02004_fort.21  small_03003_fort.21  small_04002_fort.21  
ransmall_02001  small_01001.out     small_01005_fort.22  small_02004_fort.22  small_03003_fort.22  small_04002_fort.22  
ransmall_02002  small_01001_fort.21  small_02.inp        small_02005.err      small_03004.err      small_04003.err  
ransmall_02003  small_01001_fort.22  small_02.out        small_02005.log      small_03004.log      small_04003.log  
ransmall_02004  small_01002.err     small_02001.err      small_02005.out      small_03004.out      small_04003.out  
ransmall_02005  small_01002.log     small_02001.log      small_02005_fort.21  small_03004_fort.21  small_04003_fort.21  
ransmall_02006  small_01002.out     small_02001.out      small_02005_fort.22  small_03004_fort.22  small_04003_fort.22  
ransmall_03001  small_01002_fort.21  small_02001_fort.21  small_03.inp        small_03005.err      small_04004.err  
ransmall_03002  small_01002_fort.22  small_02001_fort.22  small_03.out        small_03005.log      small_04004.log  
ransmall_03003  small_01003.err     small_02002.err      small_03001.err      small_03005.out      small_04004.out  
ransmall_03004  small_01003.log     small_02002.log      small_03001.log      small_03005_fort.21  small_04004_fort.21  
ransmall_03005  small_01003.out     small_02002.out      small_03001.out      small_03005_fort.22  small_04004_fort.22  
ransmall_03006  small_01003_fort.21  small_02002_fort.21  small_03001_fort.21  small_04.inp        small_04005.err  
ransmall_04001  small_01003_fort.22  small_02002_fort.22  small_03001_fort.22  small_04.out        small_04005.log  
ransmall_04002  small_01004.err     small_02003.err      small_03002.err      small_04001.err      small_04005.out  
ransmall_04003  small_01004.log     small_02003.log      small_03002.log      small_04001.log      small_04005_fort.21  
ransmall_04004  small_01004.out     small_02003.out      small_03002.out      small_04001.out      small_04005_fort.22  
fluka_user:/home/fluka_user/small_prod$ _
```

After running – 2

- Content of the working directory

```
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ ls  
fully-working.flair  fully-working.inp  my.exe  small_prod  tutorial.flair  
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ .
```

- Content of the working sub-directory

```
fluka_user:/home/fluka_user$ cd small_prod/  
fluka_user:/home/fluka_user/small_prod$ ls  
ransmall_01001  ransmall_04005  small_01004_fort.21  small_02003_fort.21  small_03002_fort.21  small_04001_fort.21  
ransmall_01002  ransmall_04006  small_01004_fort.22  small_02003_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01003  small_01.inp  small_01005.err  small_02004  err  
ransmall_01004  small_01.out  small_01005.log  small_02004  log  
ransmall_01005  small_01001.err  small_01005.out  small_02004  out  
ransmall_01006  small_01001.log  small_01005_fort.21  small_02004  fort.21  
ransmall_02001  small_01001.out  small_01005_fort.22  small_02004  fort.22  
ransmall_02002  small_01001_fort.21  small_02005.err  small_03004.err  small_04003.err  
ransmall_02003  small_01001_fort.22  small_02005.log  small_03004.log  small_04003.log  
ransmall_02004  small_01002.err  small_02005.out  small_03004.out  small_04003.out  
ransmall_02005  small_01002.log  small_02001.log  small_02005_fort.21  small_03004_fort.21  small_04003_fort.21  
ransmall_02006  small_01002.out  small_02001.out  small_02005_fort.22  small_03004_fort.22  small_04003_fort.22  
ransmall_03001  small_01002_fort.21  small_02001_fort.21  small_03005.err  small_04004.err  
ransmall_03002  small_01002_fort.22  small_02001_fort.22  small_03005.log  small_04004.log  
ransmall_03003  small_01003.err  small_02002.err  small_03005.out  small_04004.out  
ransmall_03004  small_01003.log  small_02002.log  small_03005_fort.21  small_04004_fort.21  
ransmall_03005  small_01003.out  small_02002.out  small_03005_fort.22  small_04004_fort.22  
ransmall_03006  small_01003_fort.21  small_02002_fort.21  small_03001_fort.21  small_04005.err  
ransmall_04001  small_01003_fort.22  small_02002_fort.22  small_03001_fort.22  small_04005.log  
ransmall_04002  small_01004.err  small_02003.err  small_03002.err  small_04005.out  
ransmall_04003  small_01004.log  small_02003.log  small_03002.log  small_04005_fort.21  
ransmall_04004  small_01004.out  small_02003.out  small_03002.out  small_04005_fort.22  
fluka_user:/home/fluka_user/small_prod$ .
```

.inp and **.out** files
specific to each spawn

small_01.inp
small_01.out

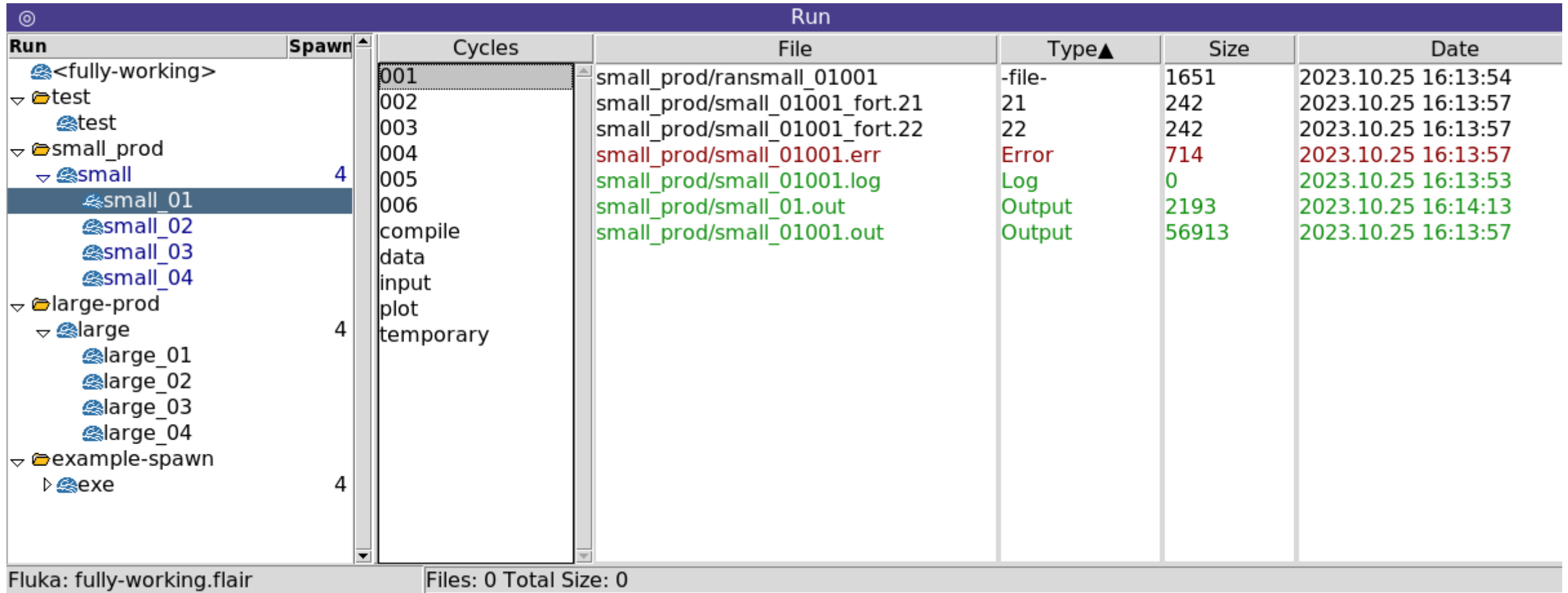
small_02.inp
small_02.out

small_03.inp
small_03.out

small_04.inp
small_04.out

Run tab – Files view – 1

- Generated files accessible via the Files view



Run	Spawr	Cycles	File	Type▲	Size	Date
<fully-working>		001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
test		002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
test		003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
small_prod		004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
small	4	005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
small_01		006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
small_02		compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
small_03		data				
small_04		input				
large_prod		plot				
large	4	temporary				
large_01						
large_02						
large_03						
large_04						
example-spawn						
exe	4					

Fluka: fully-working.flair Files: 0 Total Size: 0

Run tab – Files view – 2

- File per each cycle:
 - one (1) fluka .out file & one (1) flair .out file
 - one (1) .log file
 - one (1) .err file
 - one (1) random seed file
 - one (1) scoring file per each logical unit scoring used

Cycles	File	Type▲	Size	Date
001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
data				
input				
plot				
temporary				

Run tab – Files view – 3

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`

Cycles	File	Type▲	Size	Date
001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
data				
input				
plot				
temporary				

Run tab – Files view – 4

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`
- In this example 7 files were generated:

`small_01001.err`

`small_01001.log`

`small_01001.out`

`ransmall_01001`

`small_01001_fort.21`

`small_01001_fort.22`

`small_01.out`

Run tab – Files view – 4

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`
- In this example 7 files were generated:

`small_01001.err`

`small_01001.log`

`small_01001.out`

`ransmall_01001`

`small_01001_fort.21`

`small_01001_fort.22`

`small_01.out`

← renaming is planned

Run tab – Files view – 5

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 1 Cycle 5

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01005	-file-	1651	2023.1
002	small_prod/small_01005_fort.21	21	242	2023.1
003	small_prod/small_01005_fort.22	22	242	2023.1
004	small_prod/small_01005.err	Error	714	2023.1
005	small_prod/small_01005.log	Log	0	2023.1
006	small_prod/small_01005.out	Output	81778	2023.1
compile	small_prod/small_01.out	Output	2193	2023.1
data				
input				
plot				
temporary				

Run tab – Files view – 6

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 2 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_02001	-file-	1651	2
002	small_prod/small_02001_fort.21	21	242	2
003	small_prod/small_02001_fort.22	22	242	2
004	small_prod/small_02001.err	Error	714	2
005	small_prod/small_02001.log	Log	0	2
006	small_prod/small_02001.out	Output	81901	2
compile	small_prod/small_02.out	Output	2193	2
data				
input				
plot				
temporary				

Run tab – Files view – 7

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 1 Cycle 6

Cycles	File	Type▲	Size	
001	small_prod/ransmall_02006	-file-	1651	2
002	small_prod/small_02.out	Output	2193	2
003				
004				
005				
006				
compile				
data				
input				
plot				
temporary				

Random sequence file for the next cycle is generated

Run tab – Data view – 1

- All the generated files need to be merged to be analysed

```
small_01001_fort.21  
small_01002_fort.21  
small_01003_fort.21  
small_01004_fort.21`  
small_01005_fort.21
```

```
small_02001_fort.21  
small_02002_fort.21  
small_02003_fort.21  
small_02004_fort.21  
small_02005_fort.21
```

```
small_03001_fort.21  
small_03002_fort.21  
small_03003_fort.21  
small_03004_fort.21  
small_03005_fort.21
```

```
small_04001_fort.21  
small_04002_fort.21  
small_04003_fort.21  
small_04004_fort.21  
small_04005_fort.21
```

Run tab – Data view – 2

- Flair automatically identifies the logical units used from the input file

The screenshot displays the Flair software interface. The top menu bar includes 'Flair', 'Input', 'Run', 'Geometry', and 'Plot'. The toolbar contains various icons for file operations and actions. The left sidebar shows a tree view of the project structure:

- Run <fully-working>
 - test
 - test
 - small_prod
 - small (4)
 - small_01
 - small_02
 - small_03
 - small_04
 - large-prod (4)
 - large (4)
 - large_01
 - large_02
 - large_03
 - large_04
 - example-spawn (4)
 - exe

The main window shows a table titled 'Detectors' with the following data:

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

Below the table, there are tabs for 'Files' and 'Parameters'. The 'Files' tab is active, showing a table with columns: File, Type, Size, and Date.

At the bottom of the window, the status bar shows 'Fluka: fully-working.flair' and 'Files: 40'.

Run tab – Data view – 3

- Flair finds all the corresponding file (per spawn and per cycle)

The screenshot shows the Flair software interface. The top menu bar includes options like Flair, Input, Run, Geometry, and Plot. Below the menu bar is a toolbar with various icons for file management and actions. The main workspace is divided into two panes. The left pane shows a tree view of the project structure, with the 'Run' tab selected. The right pane shows a table of data for the selected 'Run'.

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

File	Type	Size	Date
small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
small_prod/small_01002_fort.21	21	242	2023.10.25 16:14:01
small_prod/small_01002_fort.22	22	242	2023.10.25 16:14:01
small_prod/small_01003_fort.21	21	242	2023.10.25 16:14:05

Run tab – Data view – 4

- Process can be forced by hand:

- 1 - Select the run
- 2 - Refresh
- 3 - Scan
- 4 - Process (merge)

- Processed binary results files are generated (specific extensions: .bnn, .bnx, .rnc, etc., more in other lectures)

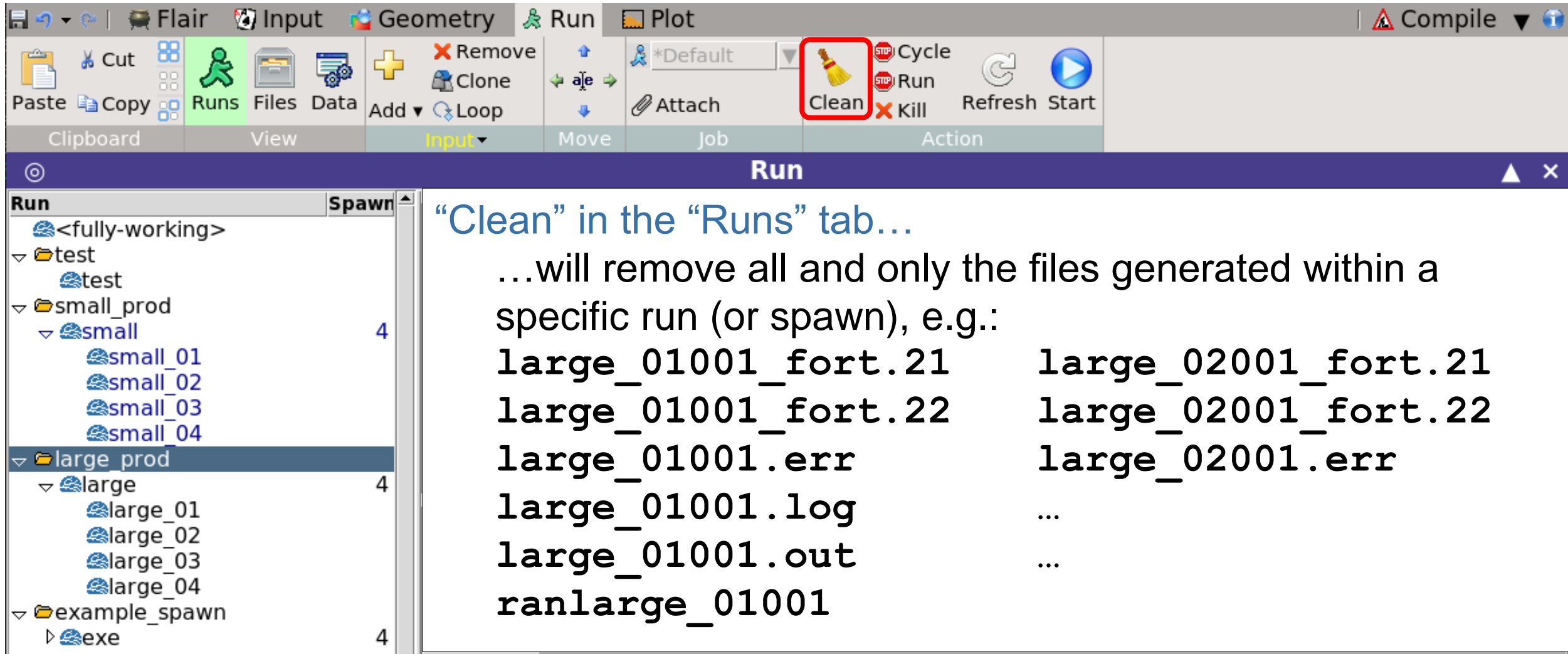
The screenshot shows the Flair software interface. The menu bar includes Flair, Input, Run, Geometry, and Plot. The toolbar contains icons for Scan (3), Refresh (2), and Process (4). The left pane shows a tree view of runs, with the 'small' run selected (1). The right pane shows a table of detectors and files.

Run	Type	Output
small_prod/small	usrbin	small_prod/small_21.bnn
small_prod/small	usrbin	small_prod/small_22.bnn

File	Type	
small_prod/small_01001_fort.21	21	242
small_prod/small_01001_fort.22	22	242
small_prod/small_01002_fort.21	21	242
small_prod/small_01002_fort.22	22	242
small_prod/small_01003_fort.21	21	242

Run tab – Cleaning – 1

- Removing files generated for the cycles and merge files are different actions!



The screenshot shows the FLUKA software interface. The top menu bar includes 'Flair', 'Input', 'Geometry', 'Run', and 'Plot'. The 'Run' tab is active, and the 'Clean' button (represented by a broom icon) is highlighted with a red box. Below the menu bar, there are several toolbars for 'Clipboard', 'View', 'Input', 'Move', 'Job', and 'Action'. The 'Run' window is open, showing a tree view of the project structure on the left and a list of files to be cleaned on the right.

Run

Run Spawn

- <fully-working>
- test
 - test
- small_prod
 - small 4
 - small_01
 - small_02
 - small_03
 - small_04
- large_prod
 - large 4
 - large_01
 - large_02
 - large_03
 - large_04
- example_spawn
 - exe 4

“Clean” in the “Runs” tab...
...will remove all and only the files generated within a specific run (or spawn), e.g.:

large_01001_fort.21	large_02001_fort.21
large_01001_fort.22	large_02001_fort.22
large_01001.err	large_02001.err
large_01001.log	...
large_01001.out	...
ranlarge_01001	

Run tab – Cleaning – 2

- Removing files generated for the cycles and merge files are different actions!

The screenshot shows the Flair software interface. The 'Data' tab is active, and the 'Clean' button (represented by a broom icon) is highlighted with a red box. The 'Run' panel on the left shows a tree view of runs, with 'small' selected under 'small_prod'. The 'Detectors' table on the right shows the following data:

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

“Clean” in the “Data” tab...
...will remove only the merged results, e.g.:
small_21.bnn
small_22.bnn

Compile tab

- Only very basic information is given here

Routine(s) to be compiled

Add routines

Select linker

Build executable

Executable: `my_exe.exe` Options:

File	Type	Size	Date
source.f	Fortran	9203	2020.08.31 15:16:21
mgdraw.f	Fortran	14783	2020.08.31 15:16:23
own_routine.cc	C++	24064	2020.08.31 15:17:07

- User routines are discussed in a dedicated lecture...

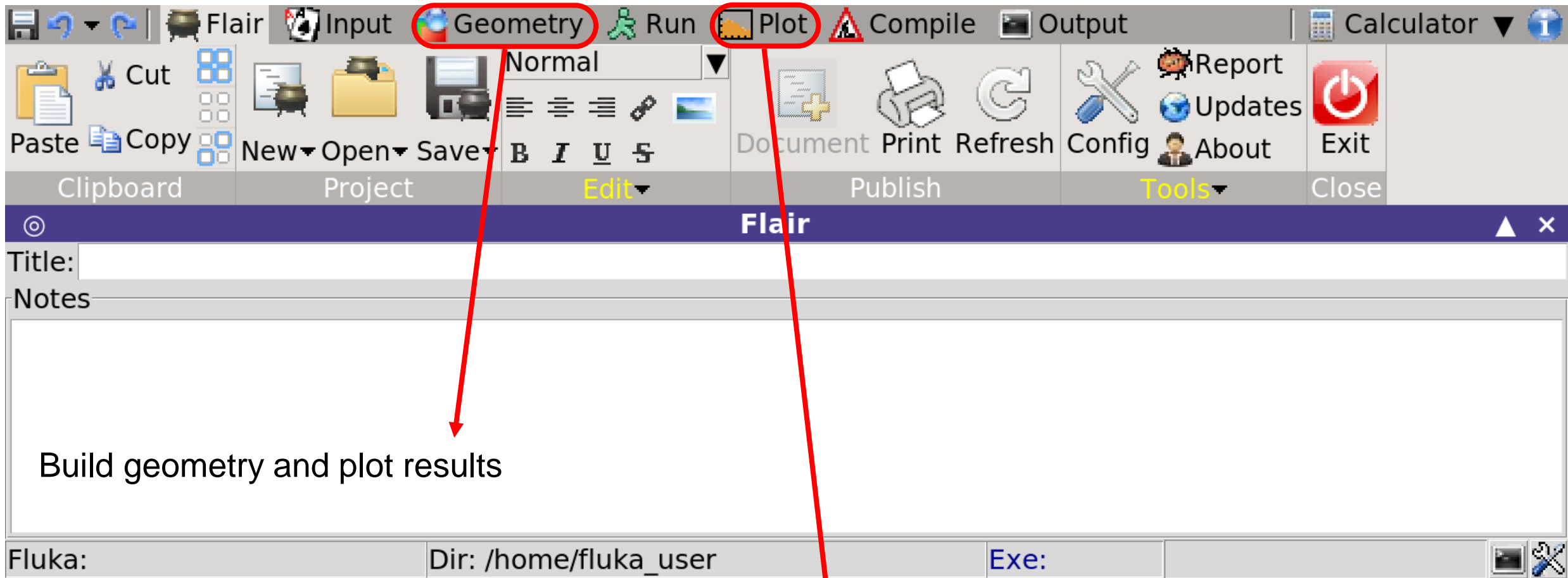
Do you remember slide 6?

The screenshot shows the Flair software interface. The top toolbar contains several buttons: 'Input', 'Geometry', 'Run', 'Plot', 'Compile', and 'Output', each circled in red. Below the toolbar is a menu bar with options like 'Cut', 'Copy', 'Paste', 'New', 'Open', 'Save', 'Document', 'Print', 'Refresh', 'Config', 'Report', 'Updates', 'About', and 'Exit'. The main window has a title bar 'Flair' and a 'Notes' section. The 'Notes' section contains the following text:

- Build input and geometry (linked from 'Input' and 'Geometry')
- Build geometry and plot results (linked from 'Geometry' and 'Plot')
- Run and merge results (linked from 'Run')
- Plot results (linked from 'Plot')
- Compile own executable (linked from 'Compile')
- Visualise output files and messages (linked from 'Output')

The status bar at the bottom shows 'Fluka: Dir: /home/fluka_user Exe:'.

Do you remember slide 6?

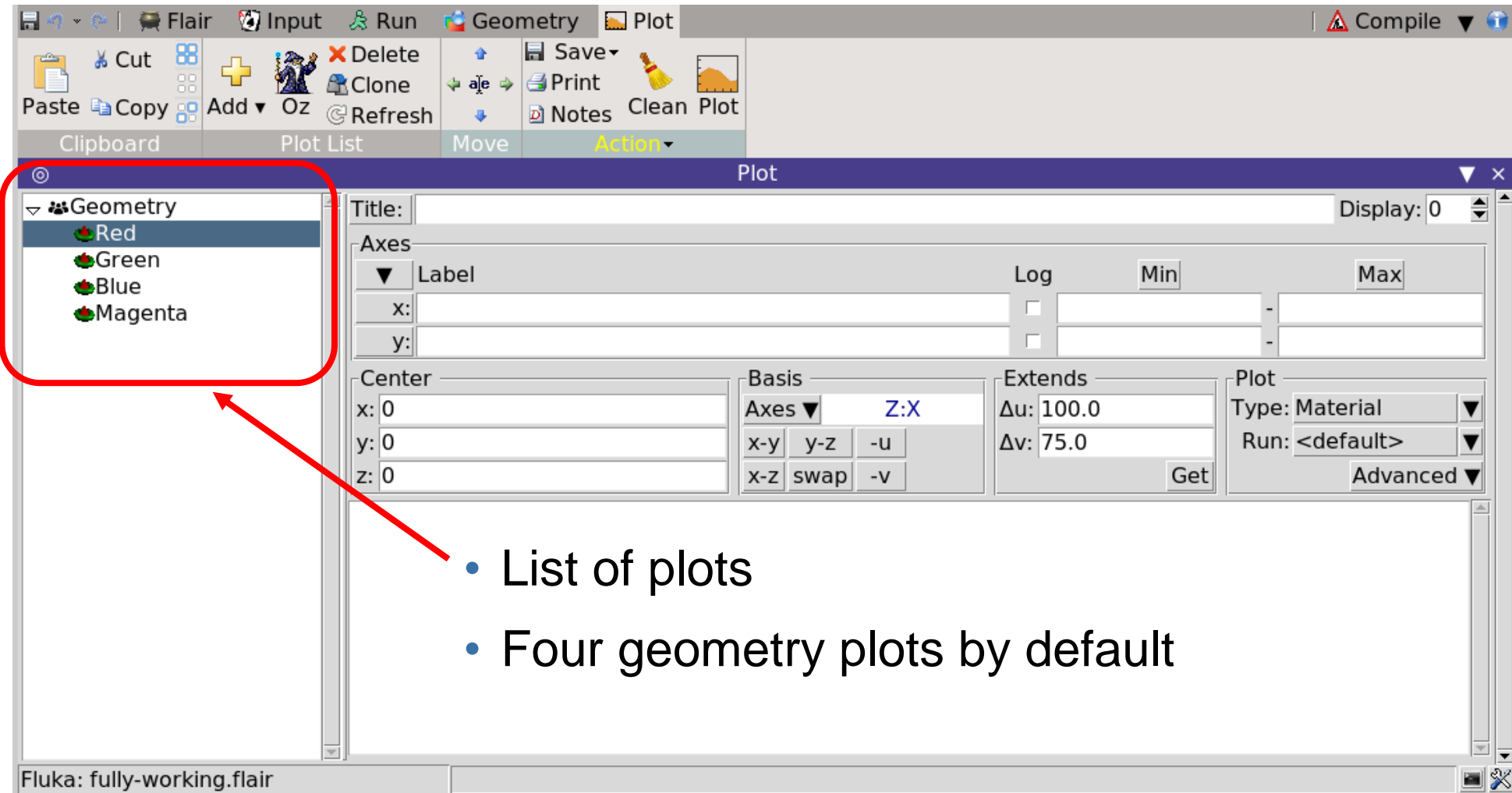


Build geometry and plot results

Plot results

Plotting results in the Plot tab – 1

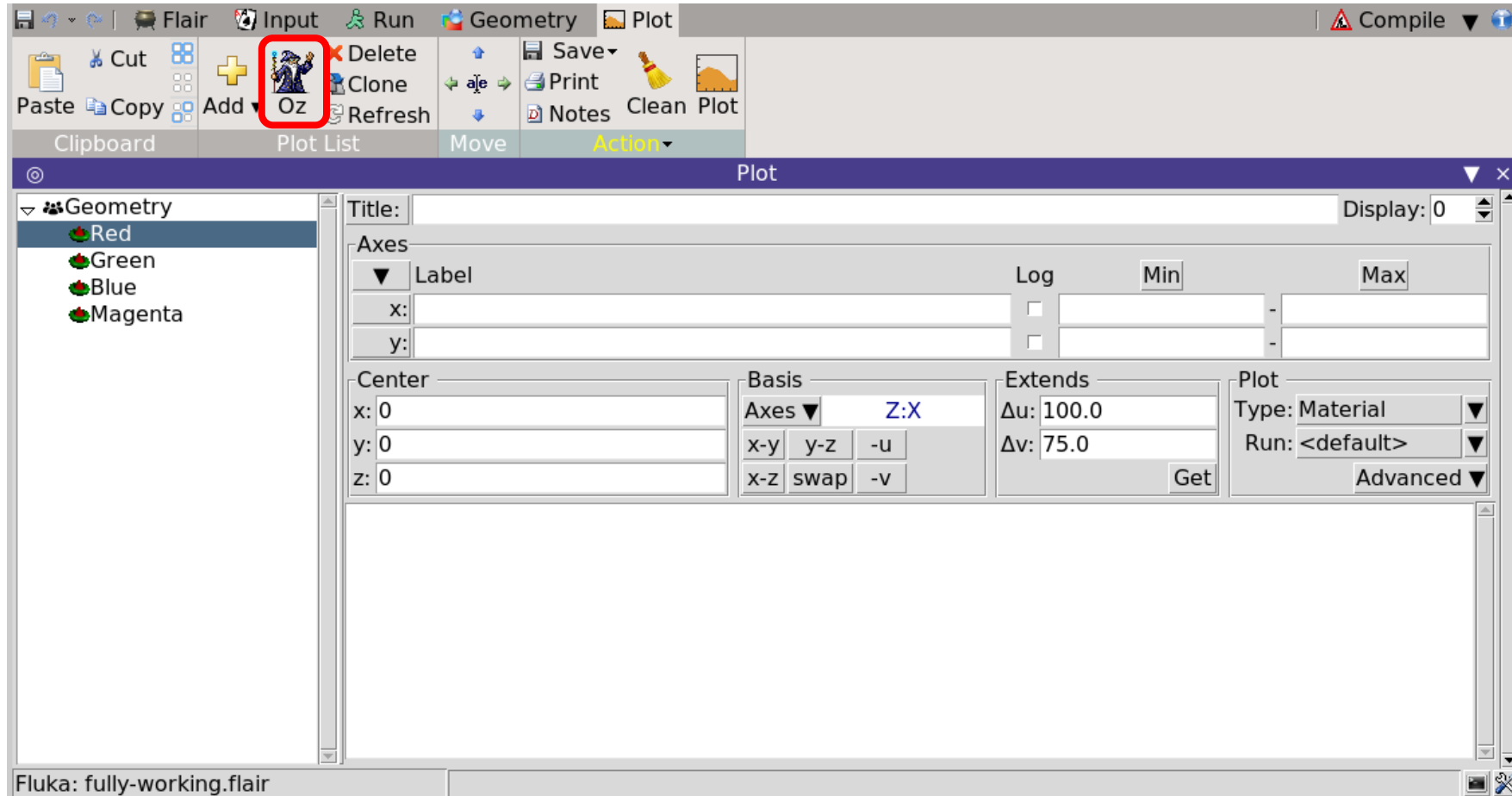
- Possible to plot geometry and all built-in scoring results



- List of plots
- Four geometry plots by default

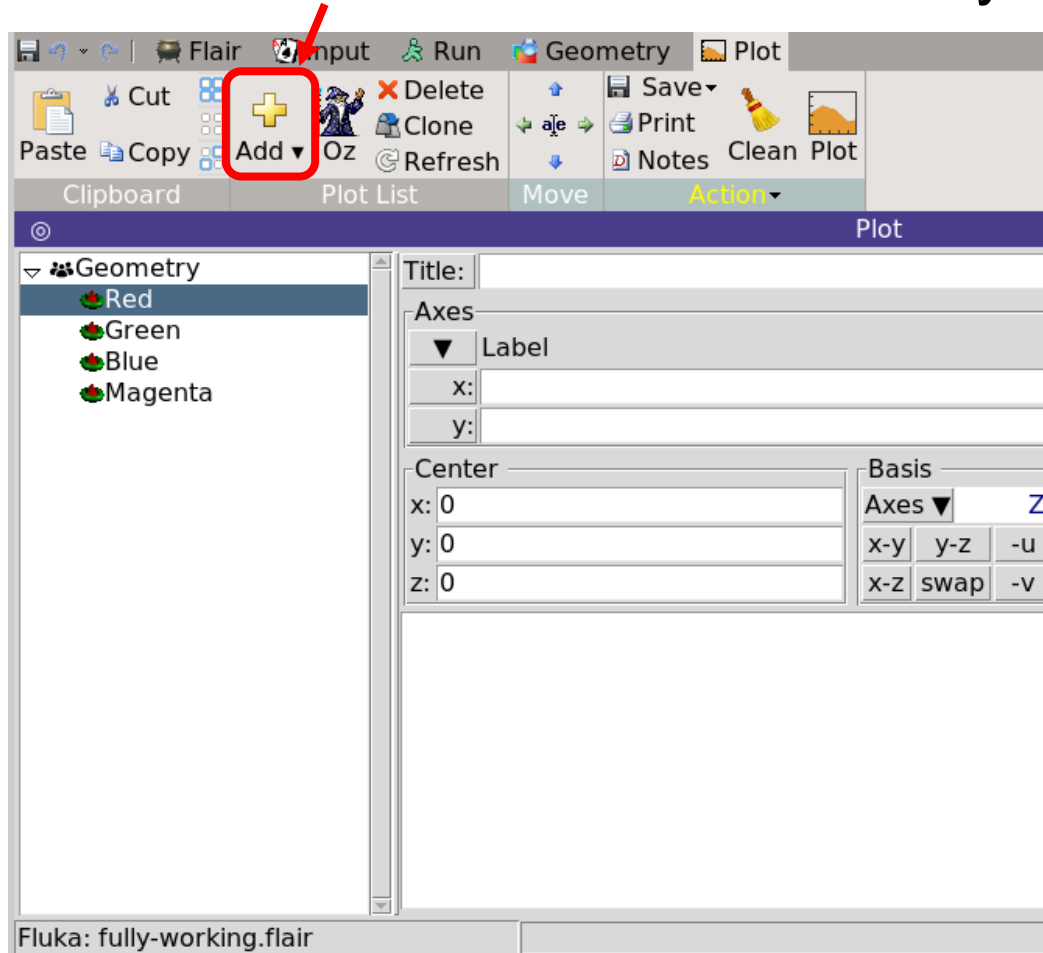
Plotting results in the Plot tab – 2

- It is possible to automatically generate the plots for all scorings in the input
- The program scans the input when “Oz” is invoked

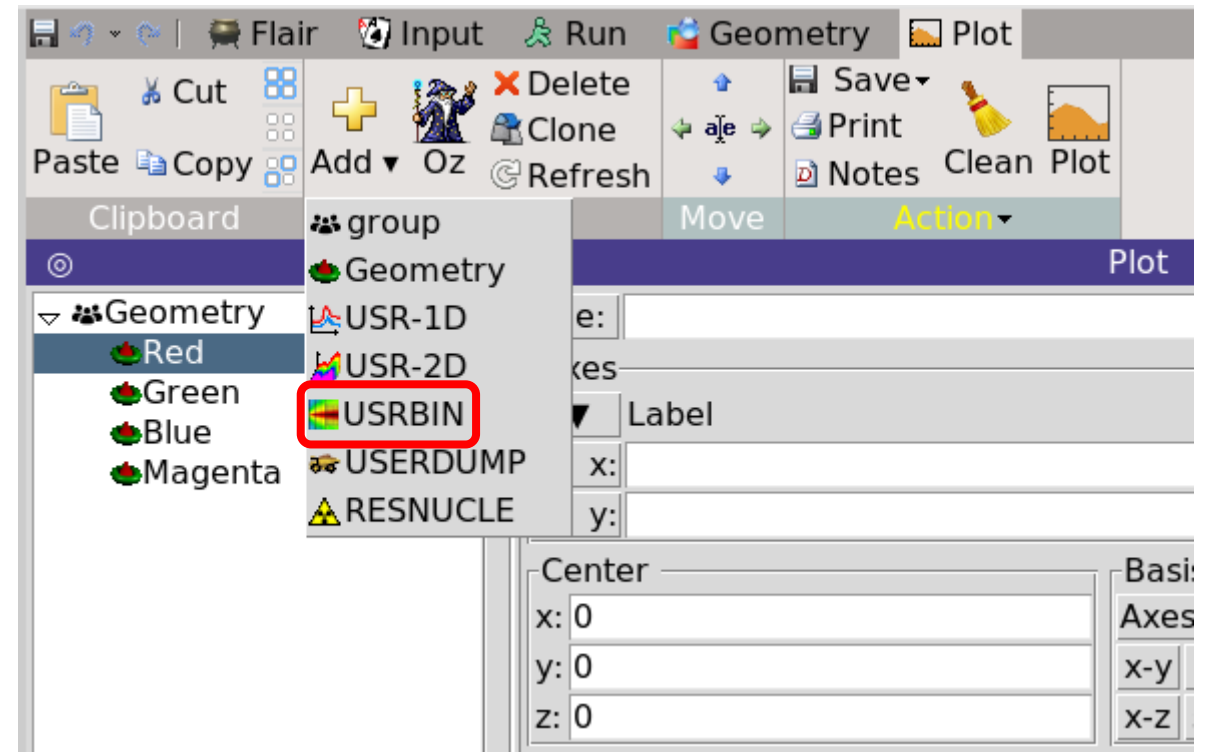


Plotting results in the Plot tab – 3

- It is possible to add plots by hand, one by one
- Click on “Add” and select the one you like from the pull down menu



(here, we'll see only USRBIN)



Plotting results in the Plot tab – 4

Name of the file that will be saved

Title of the plot

Display ID

The screenshot shows the Flair software interface with the Plot tab active. The main configuration panel is titled "Plot" and contains the following sections:

- Title:** USBIN small prod/small 21
- Display:** 0
- Axes:** A table with columns for Label, Log, Min, and Max.

Label	Log	Min	Max
x:	<input type="checkbox"/>	-	-
y:	<input type="checkbox"/>	-	-
cb:	<input checked="" type="checkbox"/>	-	-
- Binning Detector:** File: small_prod/small_21.bnn, Title: (empty), Cycles: 20, Primaries: 20000, Weight: 20000.0, Time: *****, Sum file *****
- Binning Info:** Det: 1 edep, Type: 10: X-Y-Z, Score: ENERGY, X: [-7.0 .. 7.0] x 140 (0.1), Y: [-7.0 .. 7.0] x 140 (0.1), Z: [-2.0 .. 12.0] x 140 (0.1)
- Projection & Limits:** X: 1, Y: 1, Z: 1, Norm: (empty), UMesh: (empty), Type: 2D Projection, Geometry Use: -Auto-, Pos: (empty), Axes: Auto

Red arrows in the image point to the following elements:

- The file name "small_prod/small_21_plot" in the tree view on the left.
- The "Title" field in the Plot configuration panel.
- The "Display" field in the Plot configuration panel.

Plotting results in the Plot tab – 5

Plenty of options for plot customisation

Title: Title of my plot Display: 0

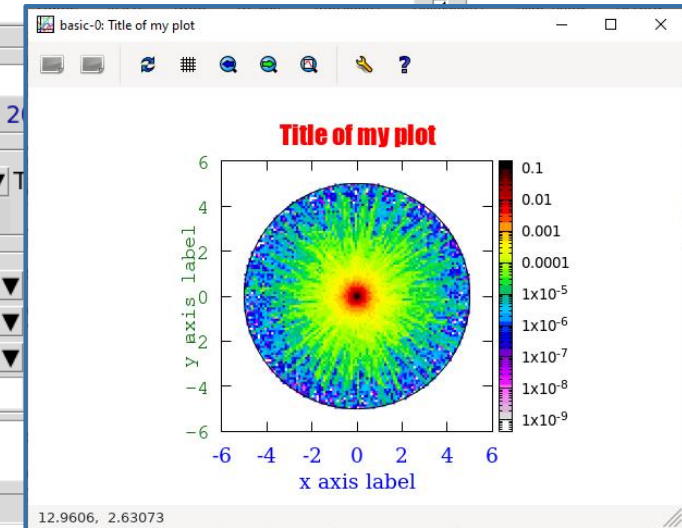
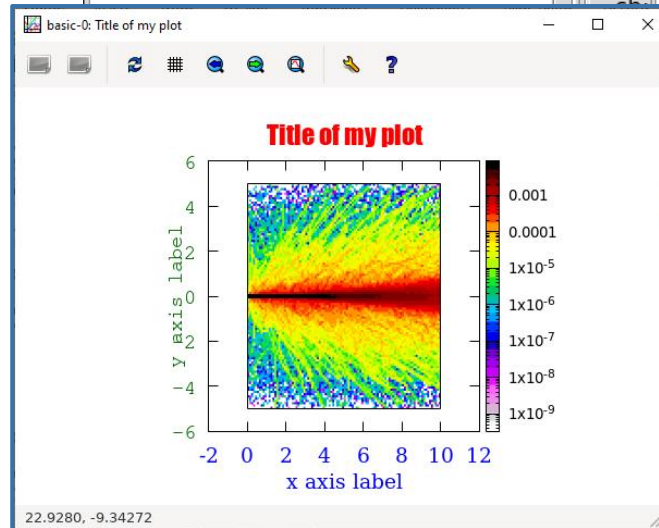
Options

font: fantasy color: options:

grid aspect: Auto lines: solid

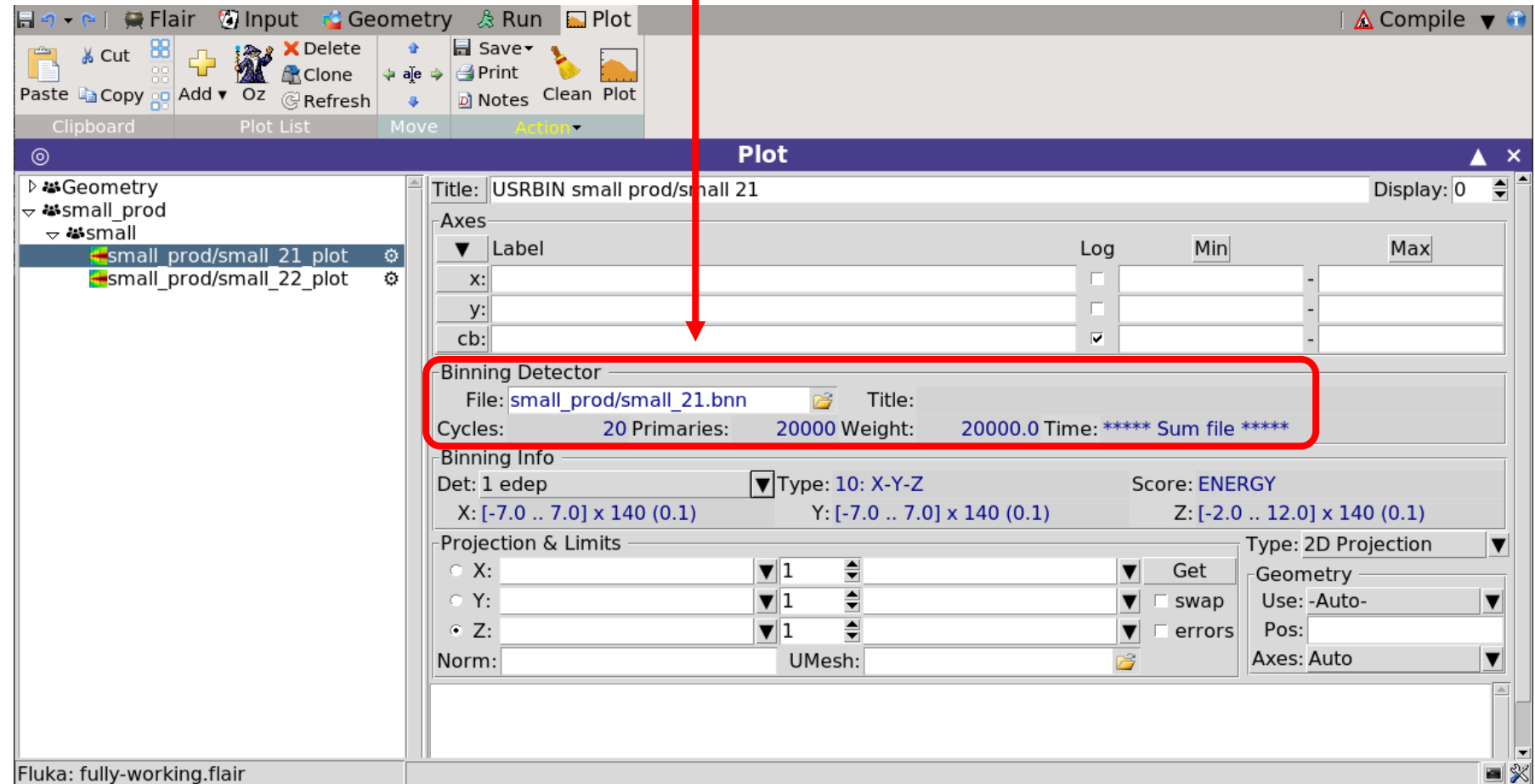
Axes

Label	Log	Min	Max
x: x axis label			
label font: serif 14 color: options:			
tics font: serif 14 color: options:			
y: y axis label			
label font: Courier 14 color: options:			
tics font: Courier 14 color: options:			



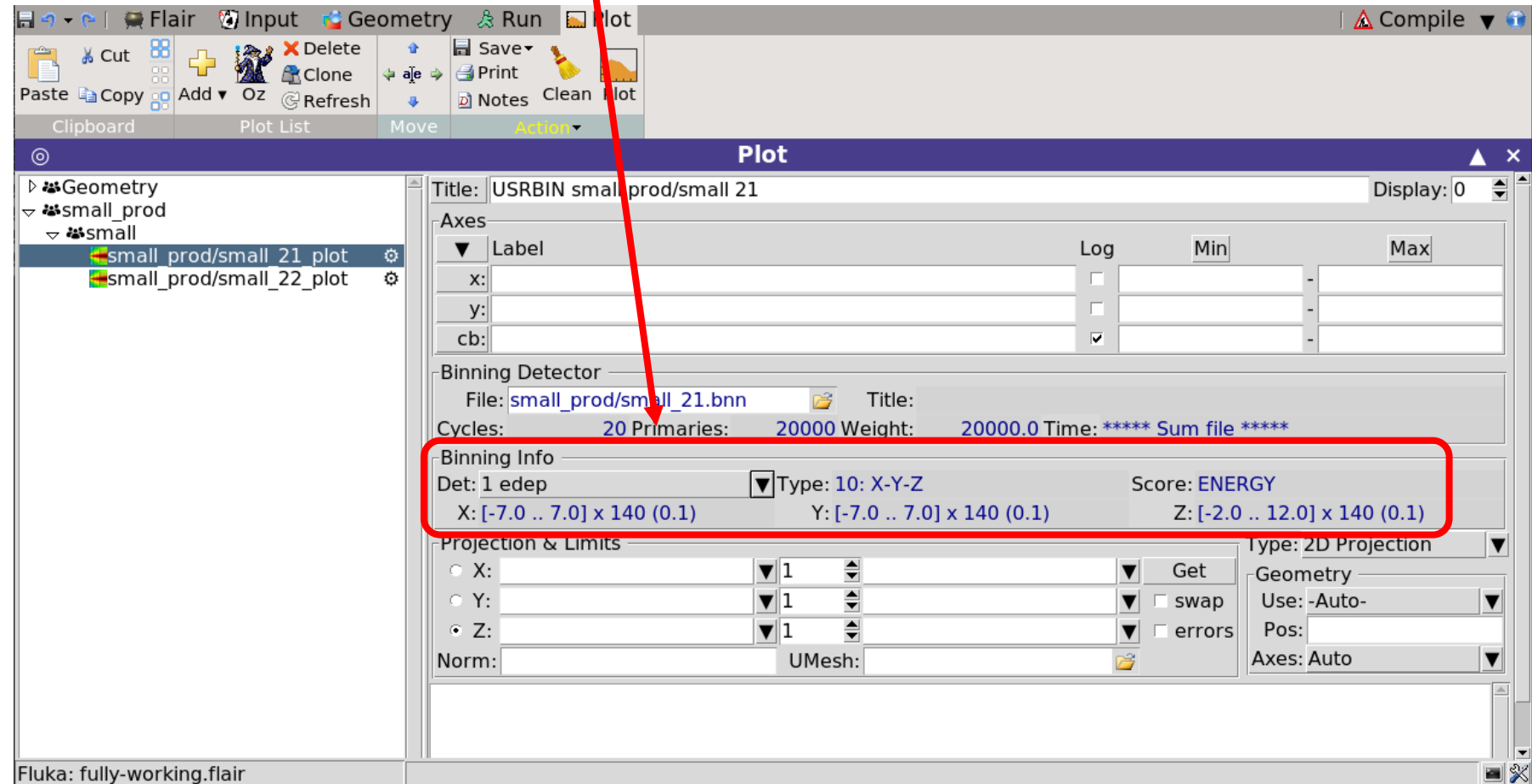
Plotting results in the Plot tab – 6

- Selection of the file containing the results of the simulations
- Opens standard pop-up for file selection
- Extra info available
 - #primaries
 - #cycles



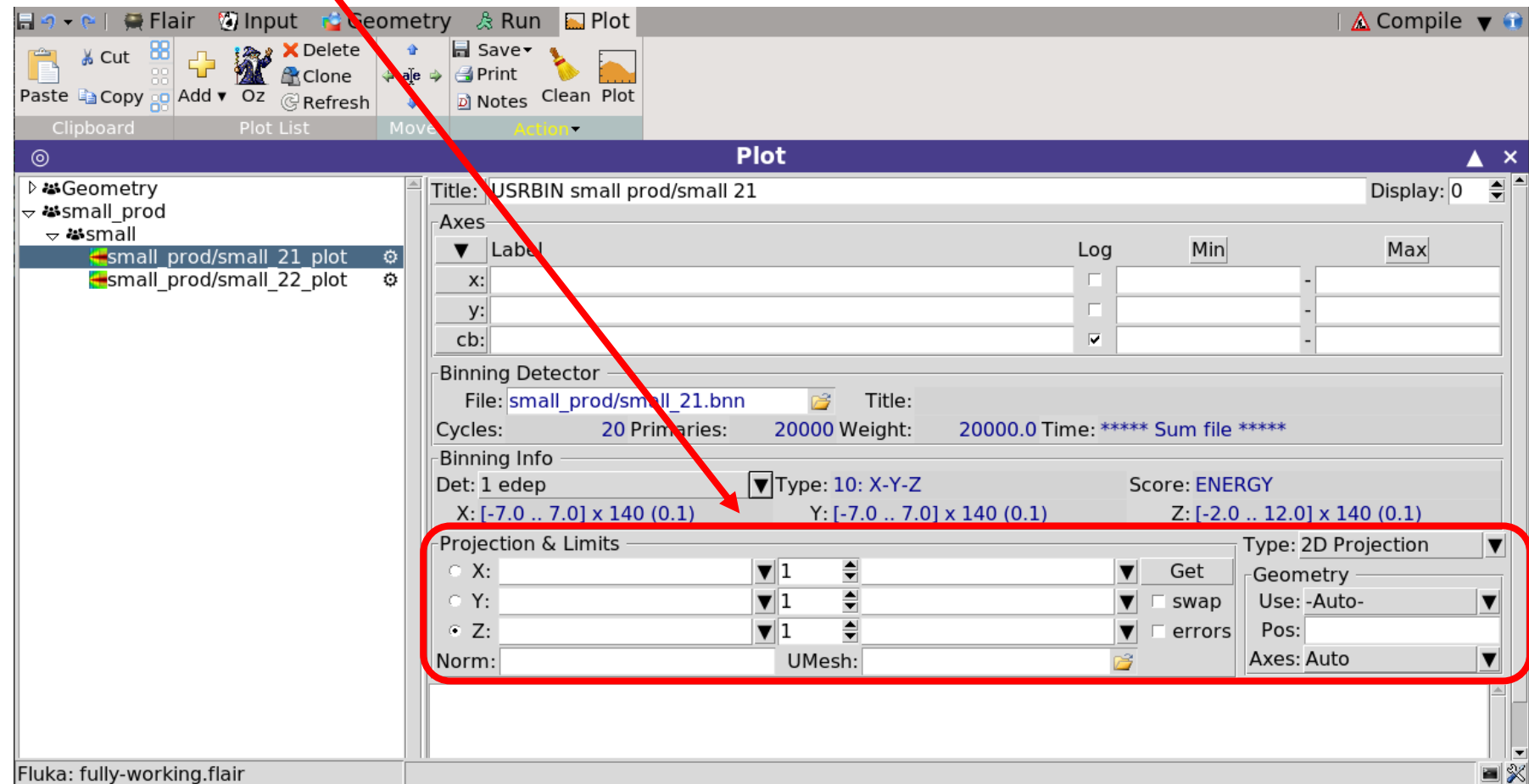
Plotting results in the Plot tab – 7

- Selection of the scoring within the chosen file (see Scoring I lecture)
- Standard pull-down menu
- Extra info available
 - Quantity scored
 - Type of mesh
 - Mesh details
 - Min & max values



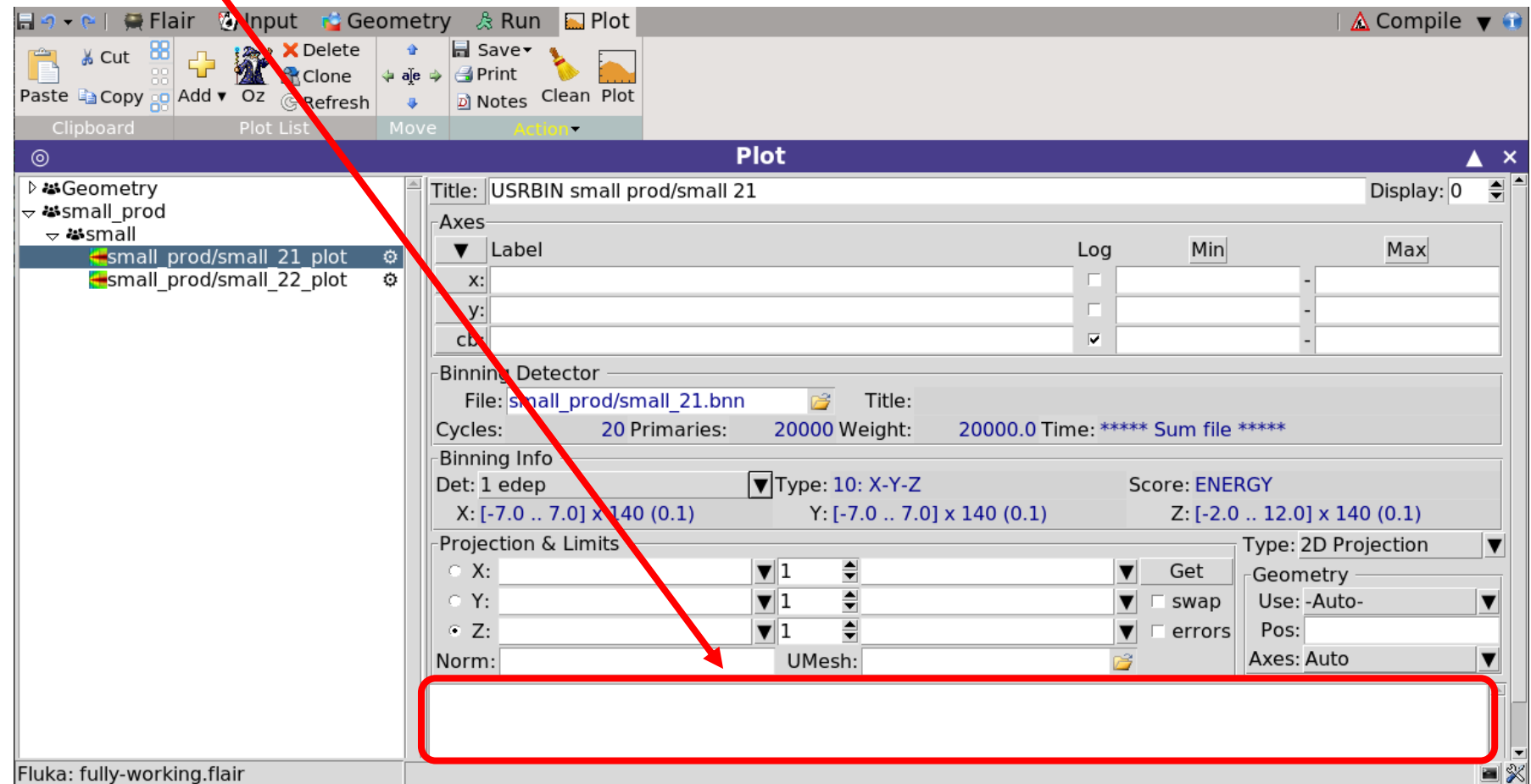
Plotting results in the Plot tab – 8

- Selection of plot type and options
 - 2D vs 1D projections
 - Plot extension
 - Uncertainty
 - Graphical options
 - Normalisation



Plotting results in the Plot tab – 9

- Additional plot customisation
 - Gnuplot commands
 - Plot extents
 - Axis location
 - Label offsets
 - Label format
 - And much more...



Plotting results in the Plot tab – 10

Projection & Limits

X: 1 Get

Y: 1 swap

Z: 1 errors

Norm:

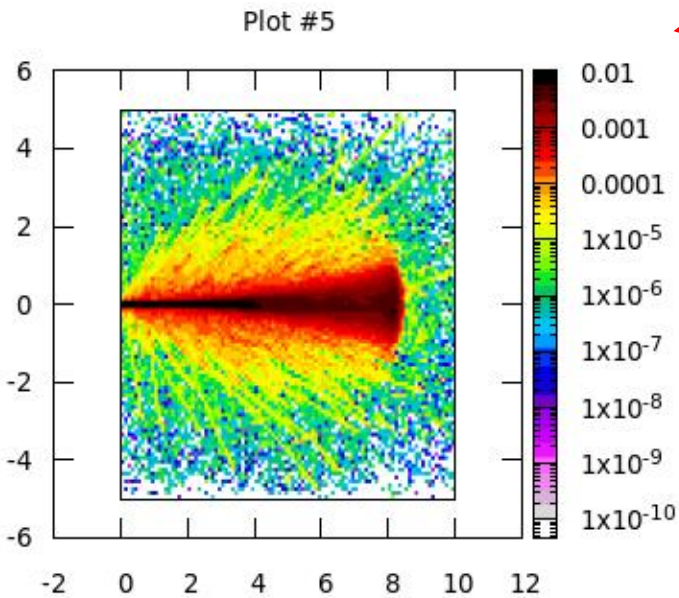
Type: 2D Projection

Geometry

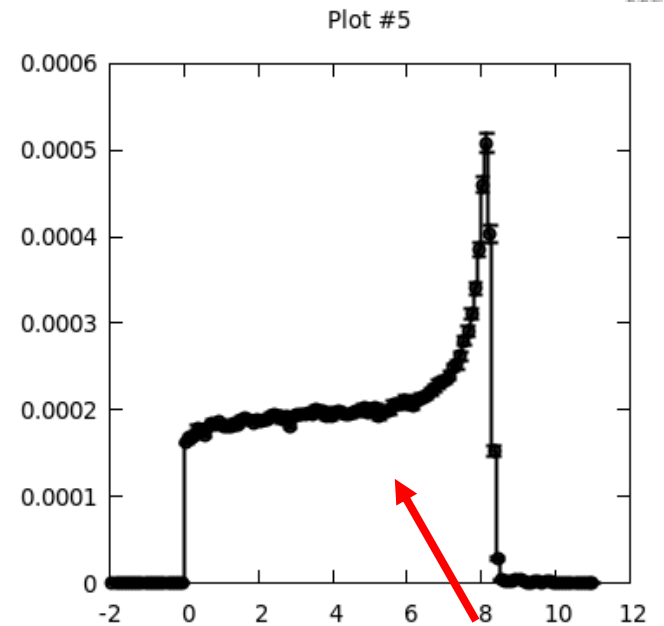
Use: -Auto-

Pos:

Axes: Auto



2D vs 1D projections



Projection & Limits

X: 1 Get

Y: 1 swap

Z: 1 errors

Norm:

Type: 1D Projection

Options

Type: histerror

Color: black Line width: 2

Point type: circle Point size: 1

Plotting results in the Plot tab – 11

Projection & Limits

X: ▼ 1 ▼ Get

Y: ▼ 1 ▼ swap

Z: ▼ 1 ▼ errors

Norm:

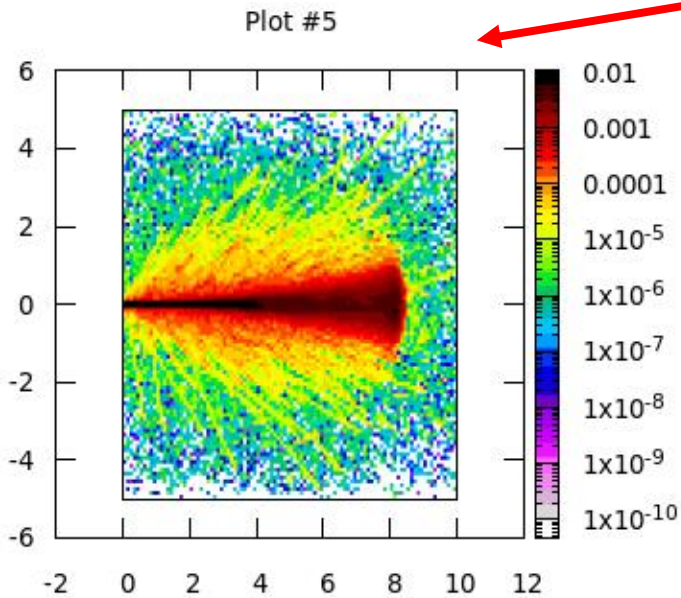
Type: 2D Projection ▼

Geometry

Use: -Auto- ▼

Pos:

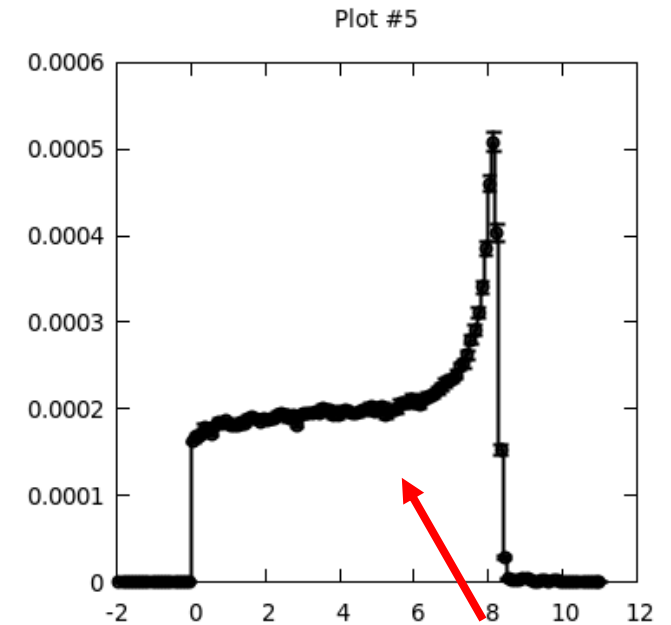
Axes: Auto ▼



- 2D: the result is averaged over the selected coordinate i.e. a X-Z plot is averaged over the Y coordinate

Plotting results in the Plot tab – 12

- 1D: the result is projected along the selected coordinate and averaged over the non-selected coordinates
i.e. a projection along the Z axis



Projection & Limits

<input type="radio"/> X:	▼ 1 ▲	▼	Get
<input type="radio"/> Y:	▼ 1 ▲	▼	<input type="checkbox"/> swap
<input checked="" type="radio"/> Z:	▼ 1 ▲	▼	<input type="checkbox"/> errors

Norm:

Type: 1D Projection ▼

Options

Type: histerror ▼	Line width: 2 ▲
Color: black ▼	Point size: 1 ▲
Point type: circle ▼	

Plotting results in the Plot tab – 13

Projection & Limits

X: 1

Y: -0.2

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

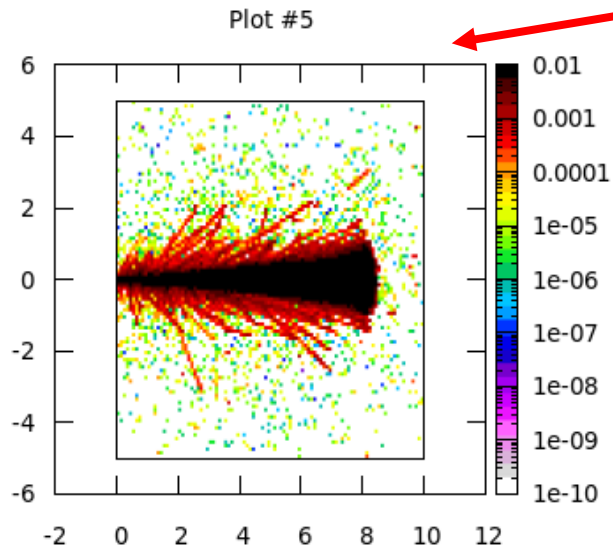
Pos:

Axes: Auto

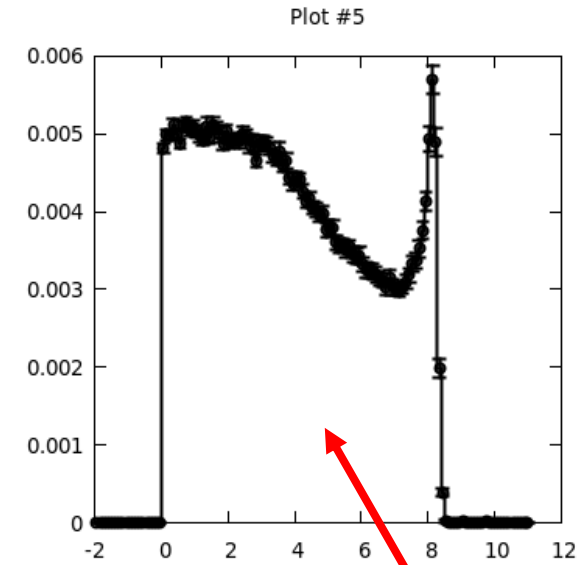
Get

swap

errors



- Plot extents
- The result is averaged only within the specified limits



Projection & Limits

X: 1

Y: -0.2

Z: 1

Norm:

Type: 1D Projection

Options

Type: histerror

Color: black

Point type: circle

Line width: 2

Point size: 1

Get

swap

errors

Plotting results in the Plot tab – 14

Projection & Limits

X: 1

Y: 1

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

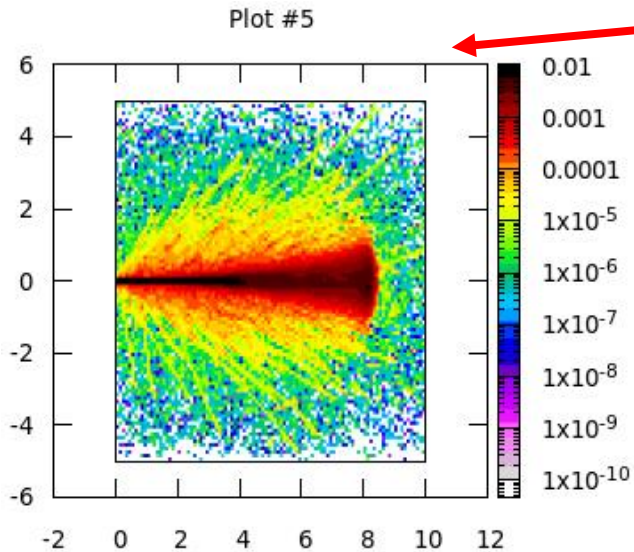
Pos:

Axes: Auto

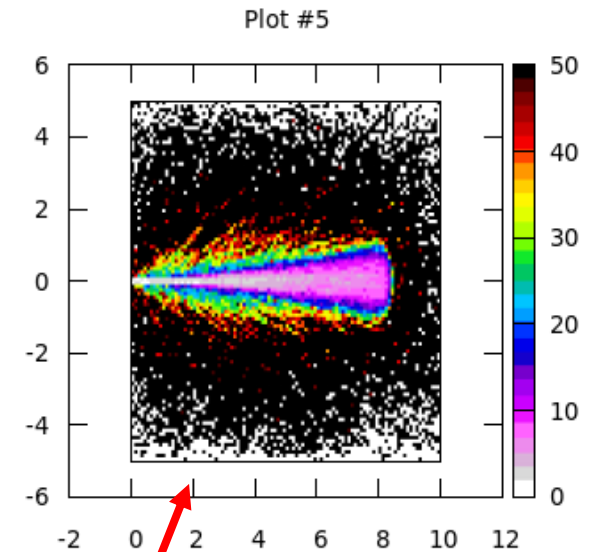
Get

swap

errors



- Uncertainty
- By ticking the “errors” box, it is possible to plot the statistical uncertainty (%)



Projection & Limits

X: 1

Y: 1

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

Pos:

Axes: Auto

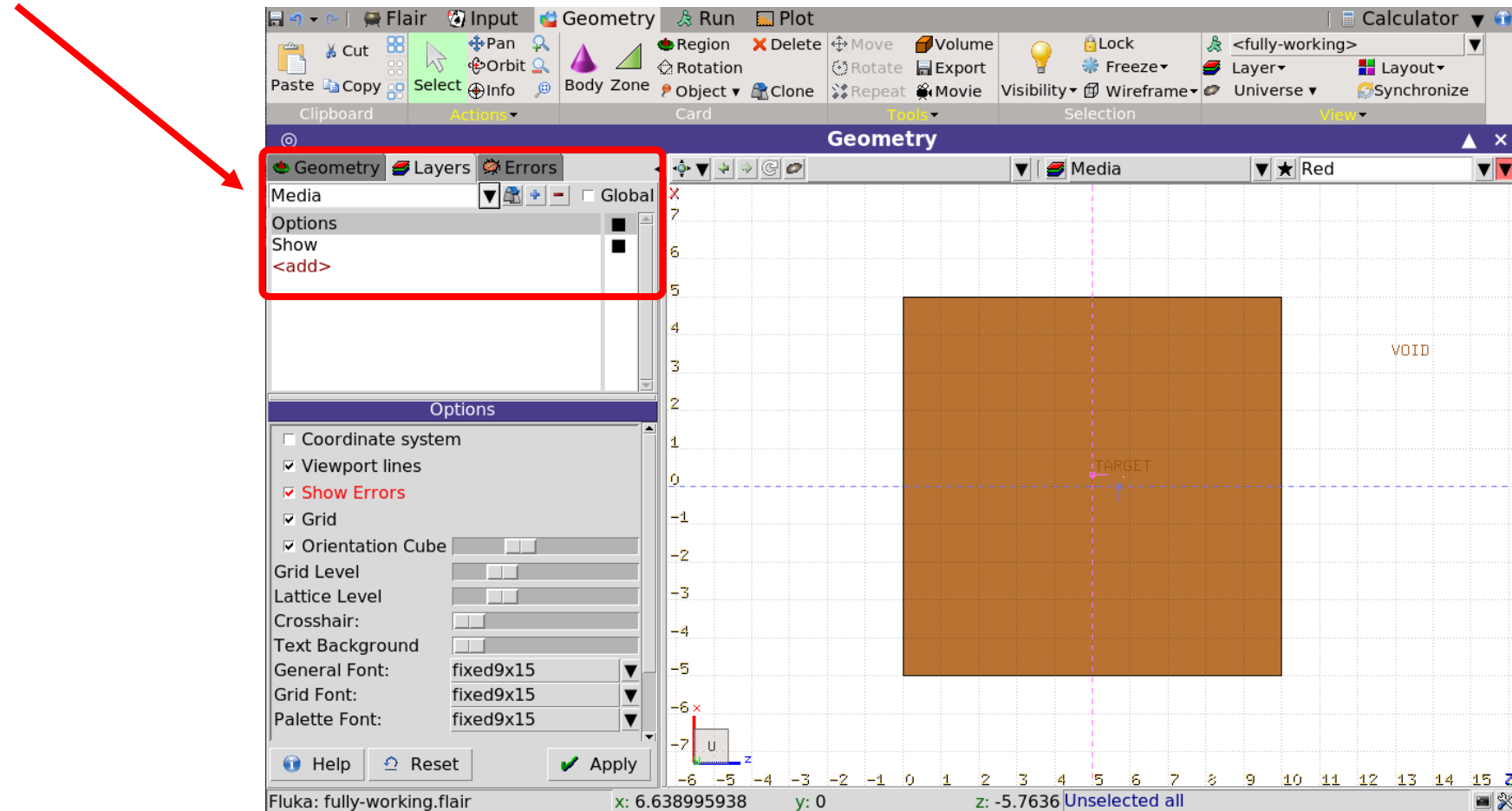
Get

swap

errors

Plotting results in the Geometry tab – 1

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one

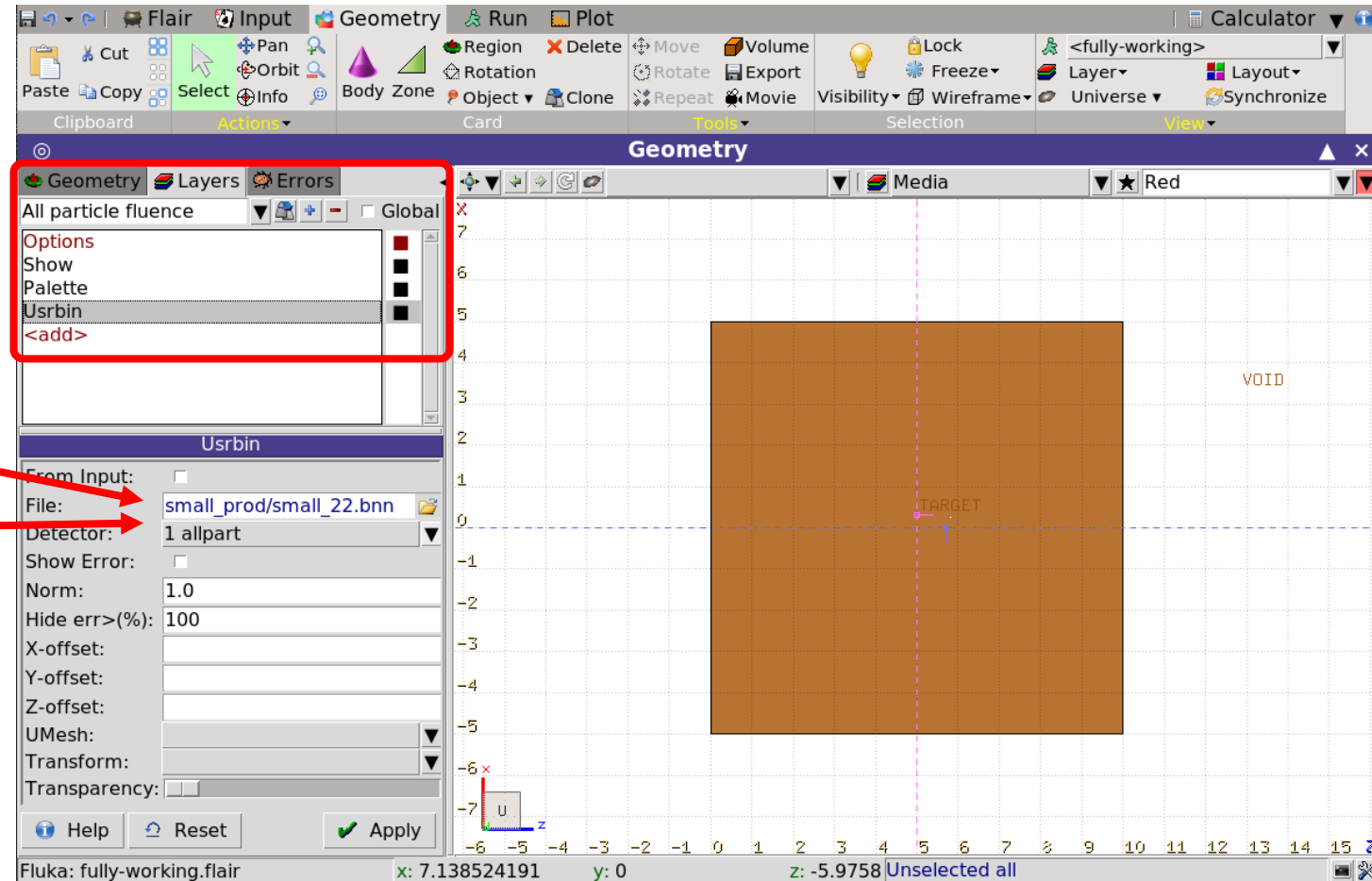


Plotting results in the Geometry tab – 2

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one

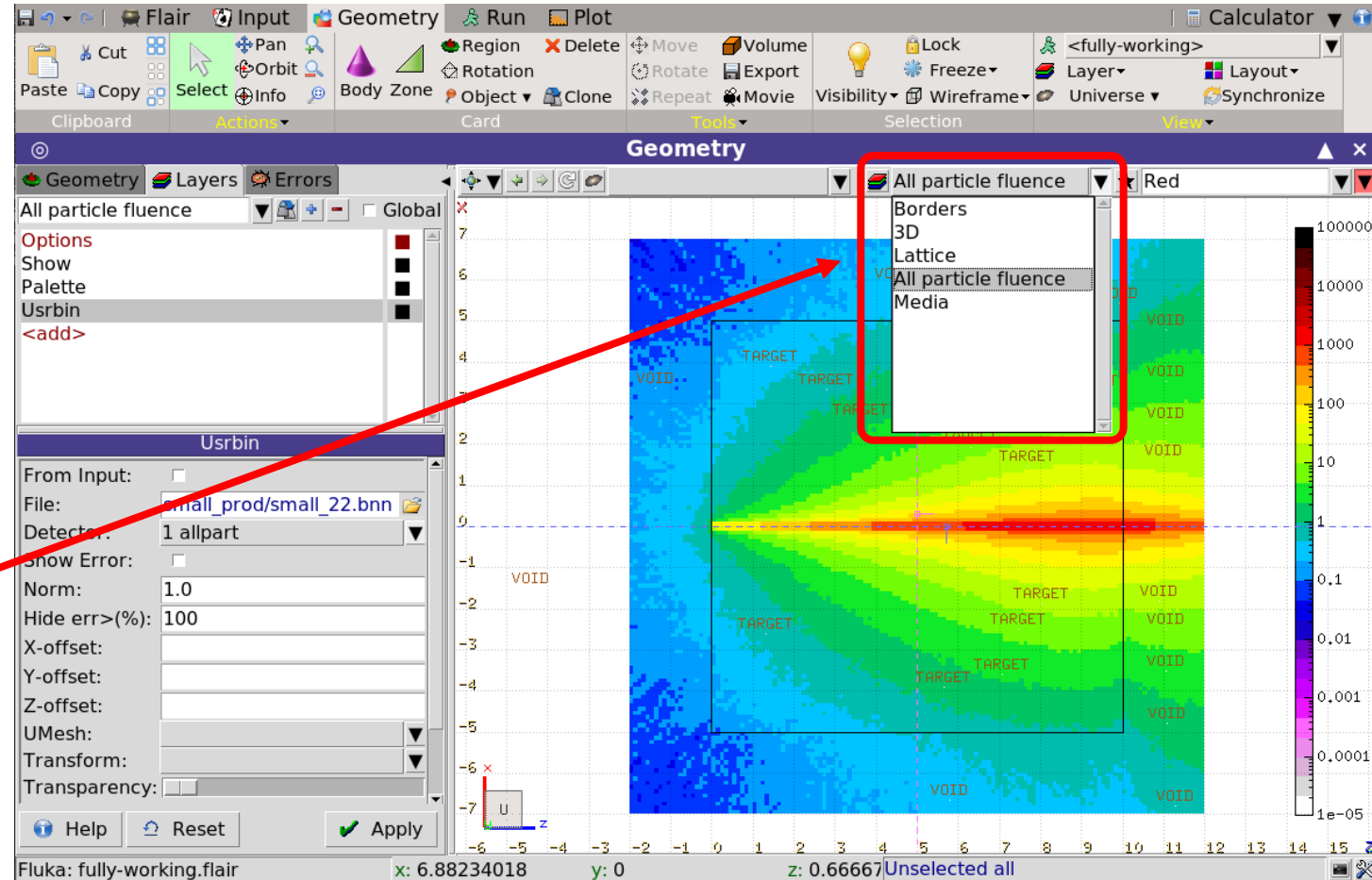
- <add> “Uusrbin”
(possible to add more than one)

- Select the file with the results
- Select the detector
- Play with normalisation, palette and other options



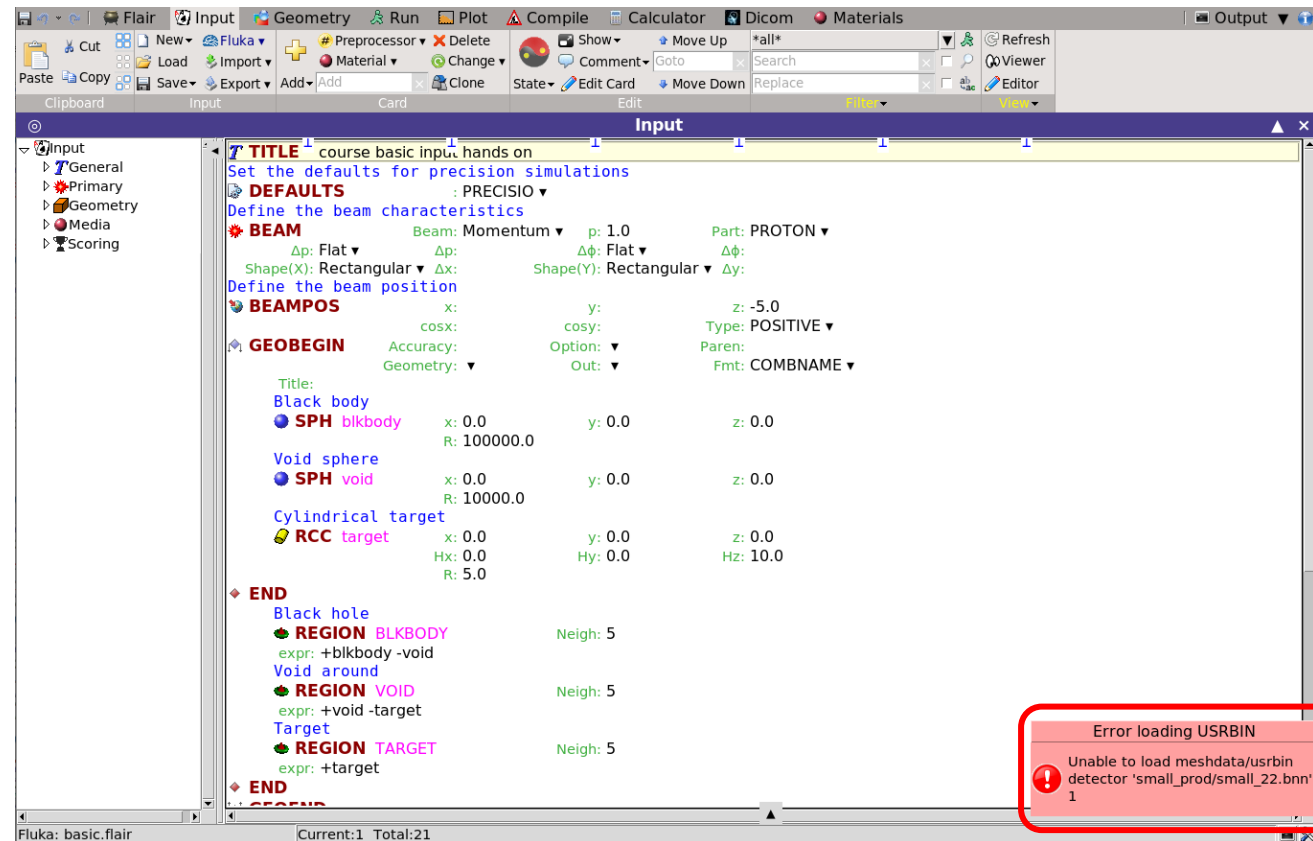
Plotting results in the Geometry tab – 3

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one
- <add> “Usrbin”
- Select the file with the results
- Select the detector
- Play with normalisation, palette and other options
- Select the layer to visualise



Plotting result in the Geometry tab – 4

- WARNING: if the USRBIN used in a layer is missing, an error message is issued
- Not necessarily something to be worried about
- This will happen in the hands-on that follows this lecture! Don't worry!



The screenshot shows the Flair software interface with the 'Input' tab selected. The input file content is as follows:

```
TITLE course basic input hands on
Set the defaults for precision simulations
DEFAULTS : PRECISIO
Define the beam characteristics
*BEAM      Beam: Momentum      p: 1.0      Part: PROTON
  Δp: Flat      Δφ: Flat
Shape(X): Rectangular Δx:      Shape(Y): Rectangular Δy:
Define the beam position
*BEAMPOS   x:      y:      z: -5.0
           cosx:   cosy:   Type: POSITIVE
*GEOBEGIN  Accuracy:  Option:  Paren:
           Geometry:  Out:    Fmt: COMBNAME
Title:
Black body
*SPH blkbody x: 0.0 y: 0.0 z: 0.0
           R: 100000.0
Void sphere
*SPH void x: 0.0 y: 0.0 z: 0.0
           R: 10000.0
Cylindrical target
*RCC target x: 0.0 y: 0.0 z: 0.0
           Hx: 0.0 Hy: 0.0 Hz: 10.0
           R: 5.0
END
Black hole
*REGION BLKBODY Neigh: 5
  expr: +blkbody -void
Void around
*REGION VOID Neigh: 5
  expr: +void -target
Target
*REGION TARGET Neigh: 5
  expr: +target
END
```

An error message box is visible in the bottom right corner, stating: "Error loading USRBIN. Unable to load meshdata/usrbin detector 'small_prod/small_22.bnn' 1".

Summary of the workflow

- Create your **input** in the Input tab and Geometry tab
- Verify your geometry in the Geometry tab
- **Run** the simulations and **merge** the output files in the Run tab
- **Plot** your results in the Plot tab and Geometry tab

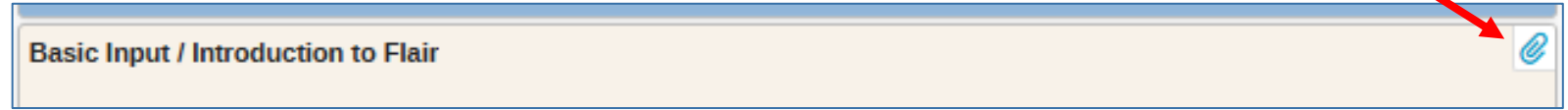
Time to practice!

- Let's start from the example file
and run a simulation step by step



Basic input and Flair introduction hands-on

- Download the input provided in the [indico timetable](#)
- Save it in a local directory
 - e.g. create a FLUKA_course directory in your Linux/WSL/macOS home directory
- Open it with Flair: `flair HO_basic_input.flair &`



Objective: become familiar with the most fundamental actions

- Start from the provided input
- Have a look at the general cards
- Go through the various Flair tabs
- Run a simple simulation and process the data
- Plot some of the results



