# Introduction to Flair and basic input

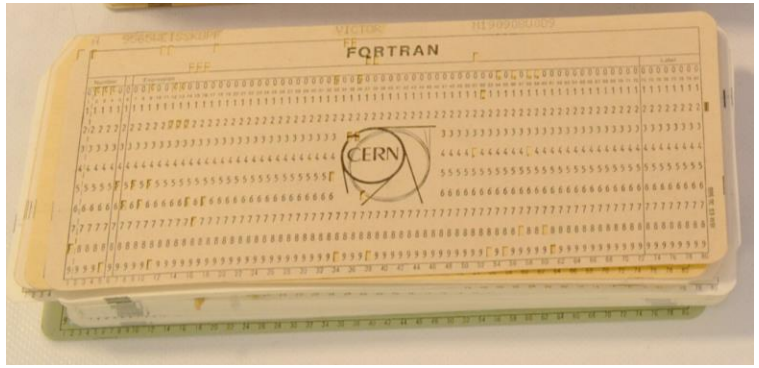
A very basic introduction to perform your first simulation

# A very short introduction

- Fluka's story began a long time ago (1970s)…

  …no graphical interfaces, input and output via text file

- Inputfile can be very long > 50k lines
- Inputfile based on "cards": `.inp` file
- Each card has 1 name, 6 values (called WHATs), 1 string (called SDUM)

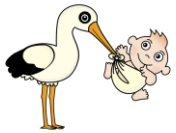



- Two examples of cards (the actual meaning is not relevant here):

```
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! 

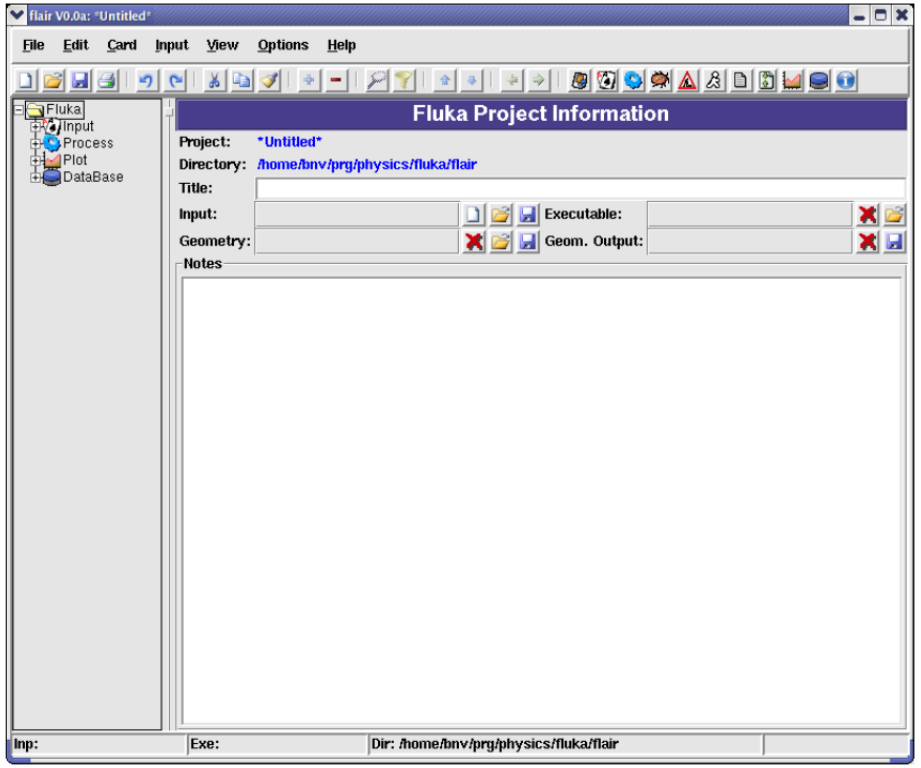
  Fluka advanced interface

  Input file creation

  Geometry visualisation and construction

  Simulation execution

  Results visualisation

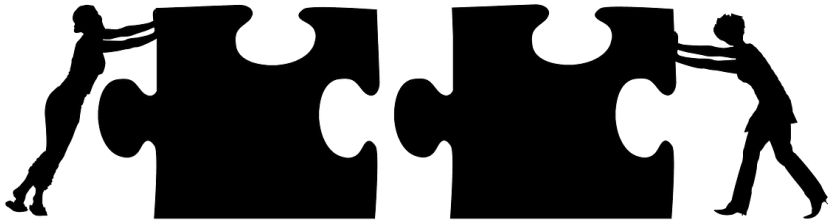



- Flair acts as an intermediate layer between the user and the inputfile
- It allows a user friendly editing of the Fluka input
- Based on a `.flair` file and generates the `.inp` file that is run by Fluka

**Flair ≠ Fluka**

# Fluka & Flair

- Although strongly linked, they are two different things (`.inp` ≠ `.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



- It is possible to work with Fluka only using text editors (for expert or old users)
- Flair is not just a graphical interface for text editing
- 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 command line, e.g.:

```
$ flair my_input.flair
```

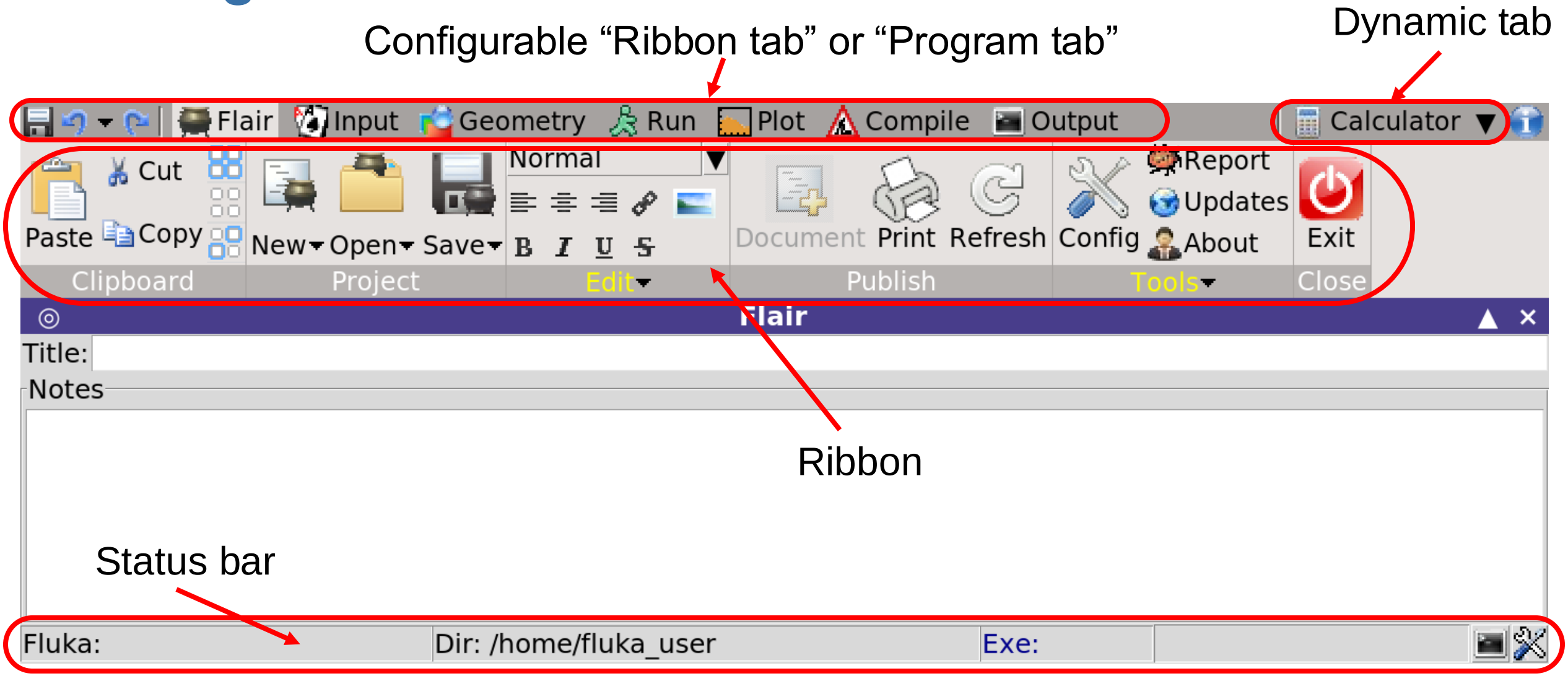
- Flair can also be used to open .inp files, e.g:

```
$ flair my_old_file.inp
```

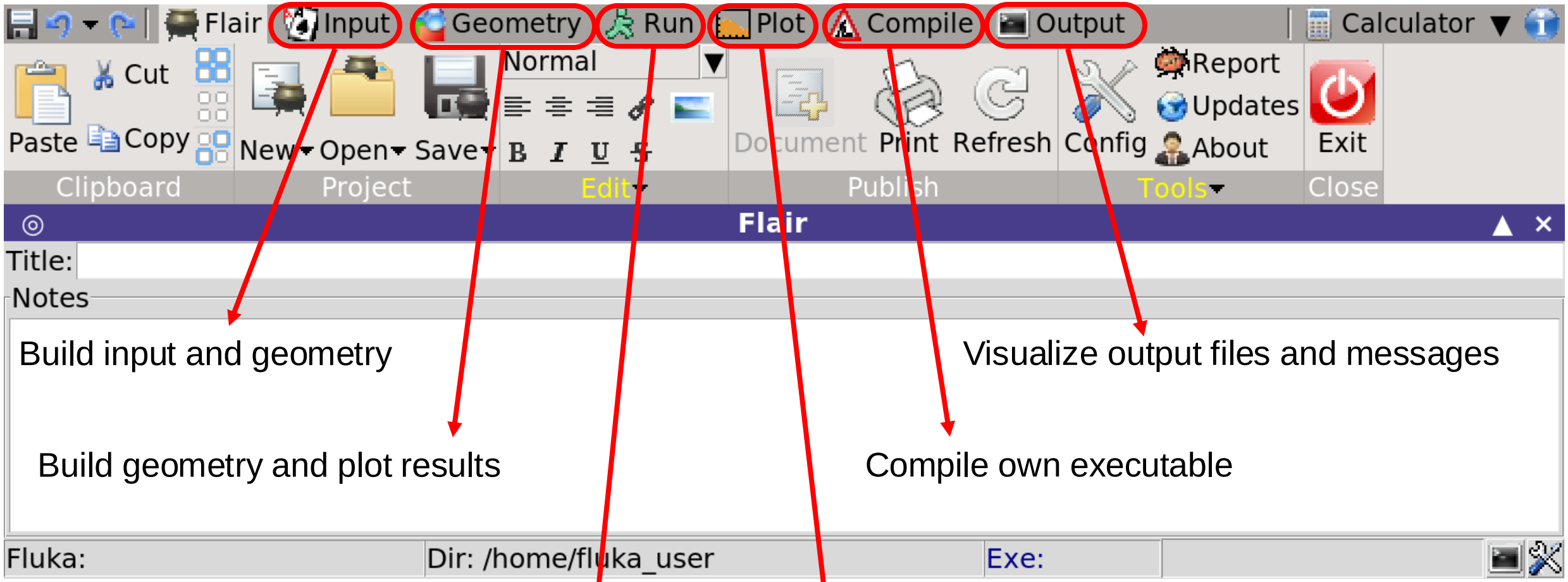
- Linux reminder about copying files:

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

# Starting Flair and basic nomenclature



# What's each tab for?




Build input and geometry

Visualize output files and messages

Build geometry and plot results

Compile own executable

Run and merge results

Plot results

# The input as a text file

- Mentioned here just for completeness

**.inp**

```
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)
```

**.flair** file includes info & instructions for the flair project

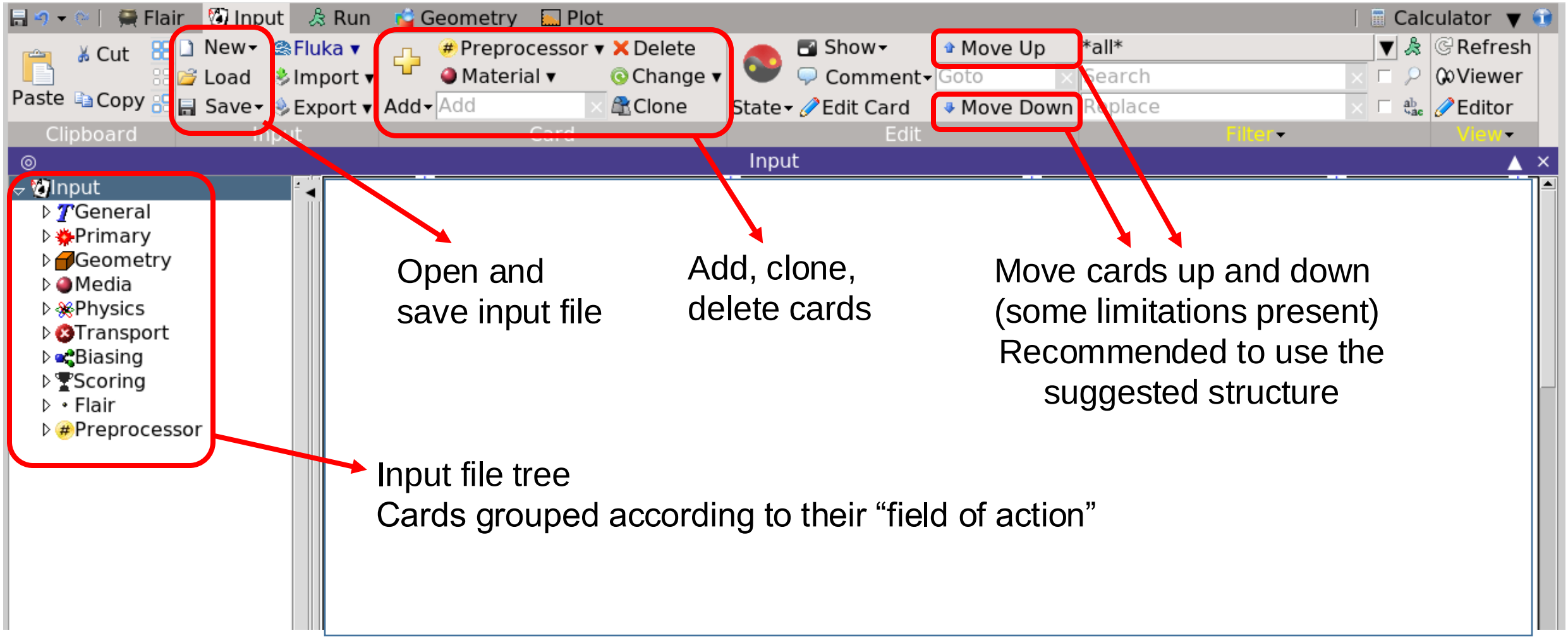
This course is based on the use of flair, no further mention of these text files

**.flair**

```
# 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
```

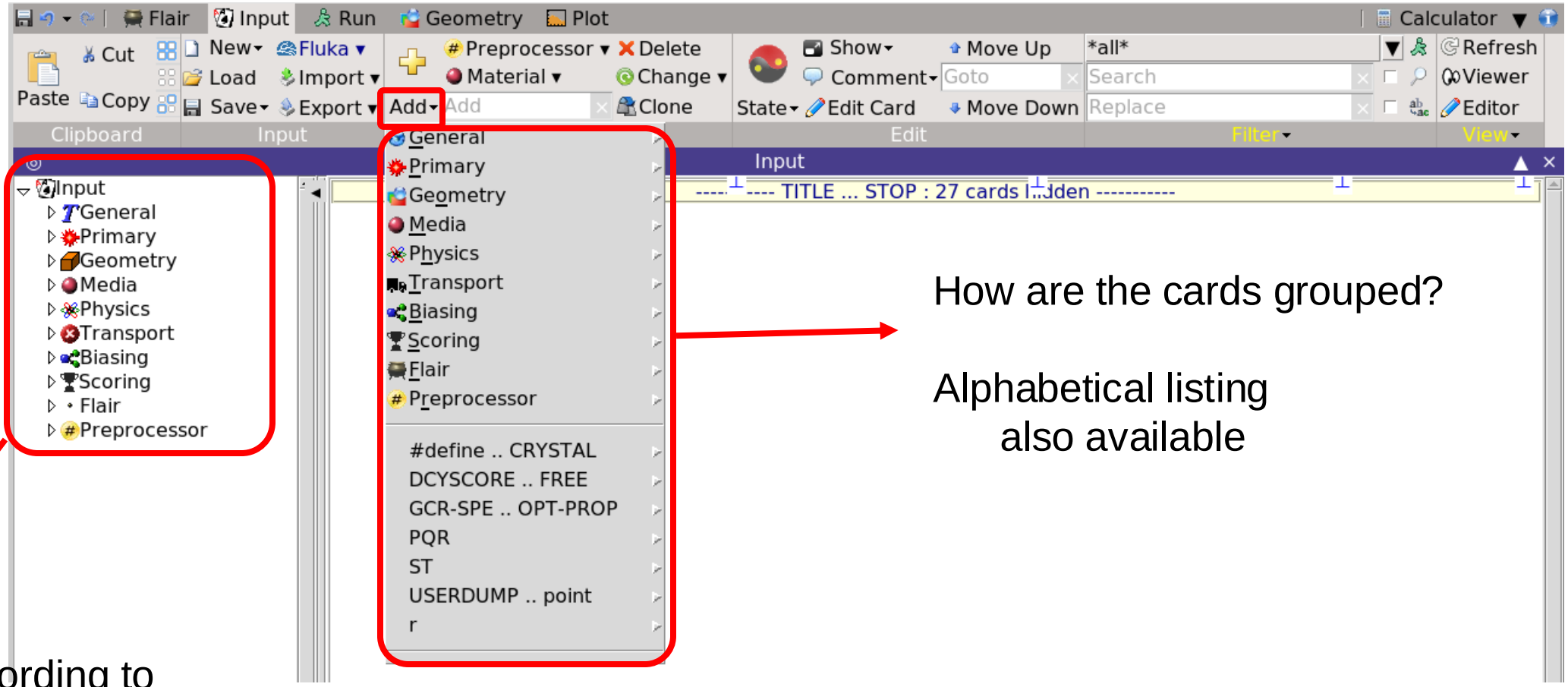
# Input tab – 1: general info

- Standard looking “Windows” tab



# Input tab – 2: input file tree and card grouping

- Input file tree and card grouping




How are the cards grouped?

Alphabetical listing
also available

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

# 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 visualization

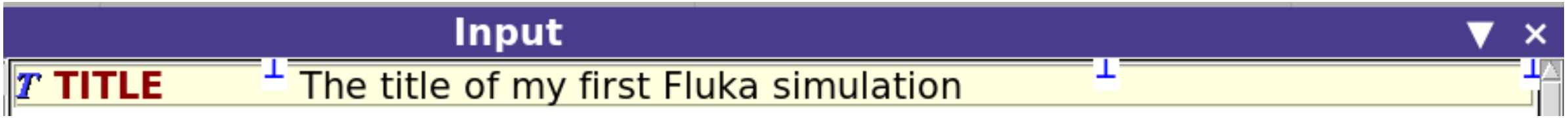
} dedicated lectures

# Input tab – 4: General cards

**TITLE** **START** **STOP** **RANDOMIZe** **DEFAULTS**

## TITLE

- Not a mandatory card
- Allows to assign a title to the simulations
- 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 particles 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.      Core:
                                        Time:            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

# Input tab – 7: General cards

| TITLE | START | STOP | RANDOMIZe | DEFAULTS |
|---|---|---|---|---|

## RANDOMIZ

- Allows to initialize 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 make parametric runs
- 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

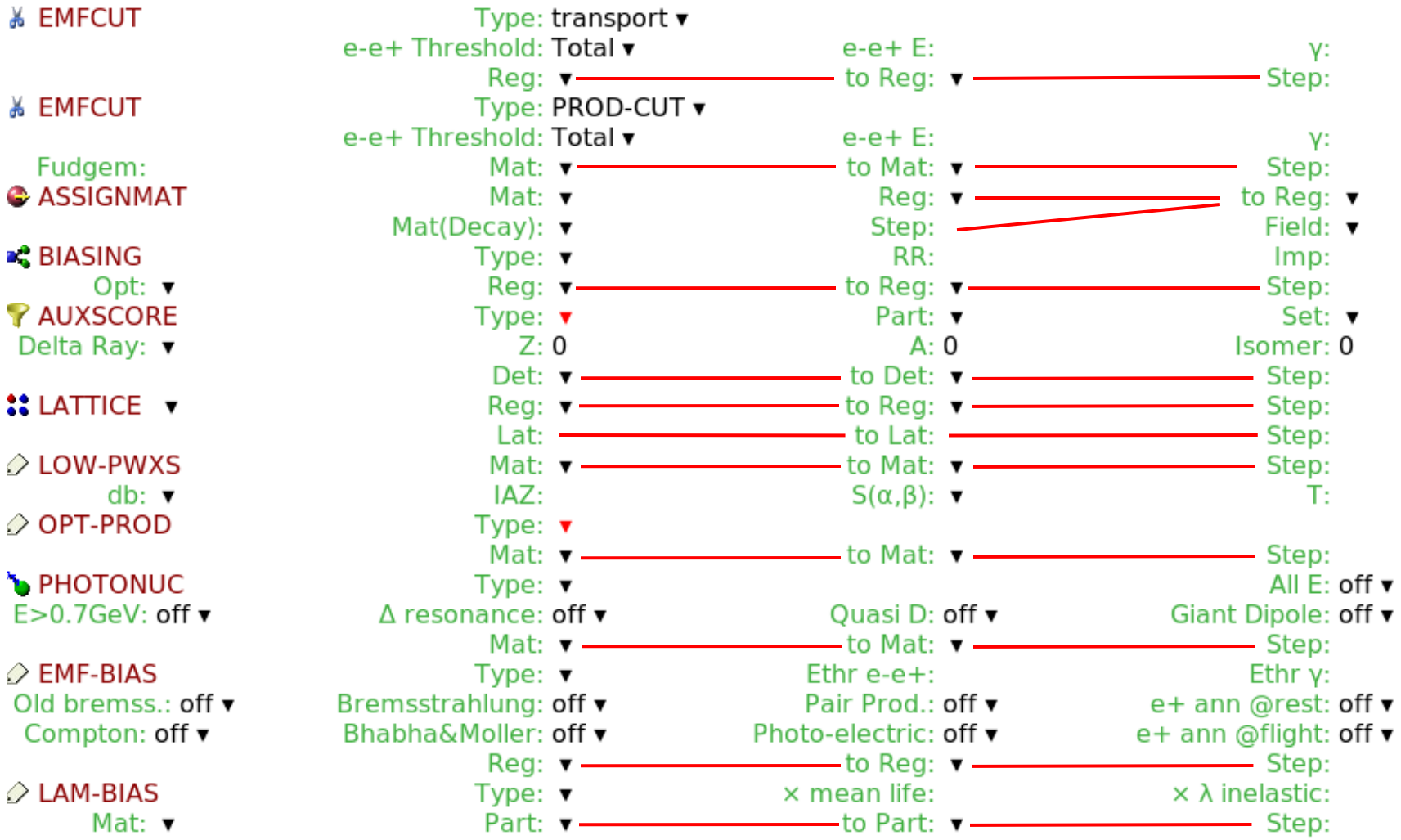
**Do not change by hand, they will be overwritten by flair!!!**

# Input tab – 10: Expressions

- See manual for details (see next slide 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
  - …



# 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”

```
REGION  region_1                                   Neigh:
      expr: +reg1
REGION  region_2                                   Neigh:
      expr: +reg2
REGION  region_3                                   Neigh:
      expr: +reg3
REGION  region_4                                   Neigh:
      expr: +reg4
ASSIGNMAT                  Mat: CARBON ▾     Reg: region_1 ▾     to Reg: region_4 ▾
                     Mat(Decay):  ▾          Step:                 Field:  ▾
```

```
REGION  region_1                                   Neigh:
      expr: +reg1
REGION  region_2                                   Neigh:
      expr: +reg2
REGION  region_3                                   Neigh:
      expr: +reg3
REGION  region_4                                   Neigh:
      expr: +reg4
ASSIGNMAT                  Mat: CARBON ▾     Reg: region_1 ▾     to Reg: region_4 ▾
                     Mat(Decay):  ▾          Step: 1               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”

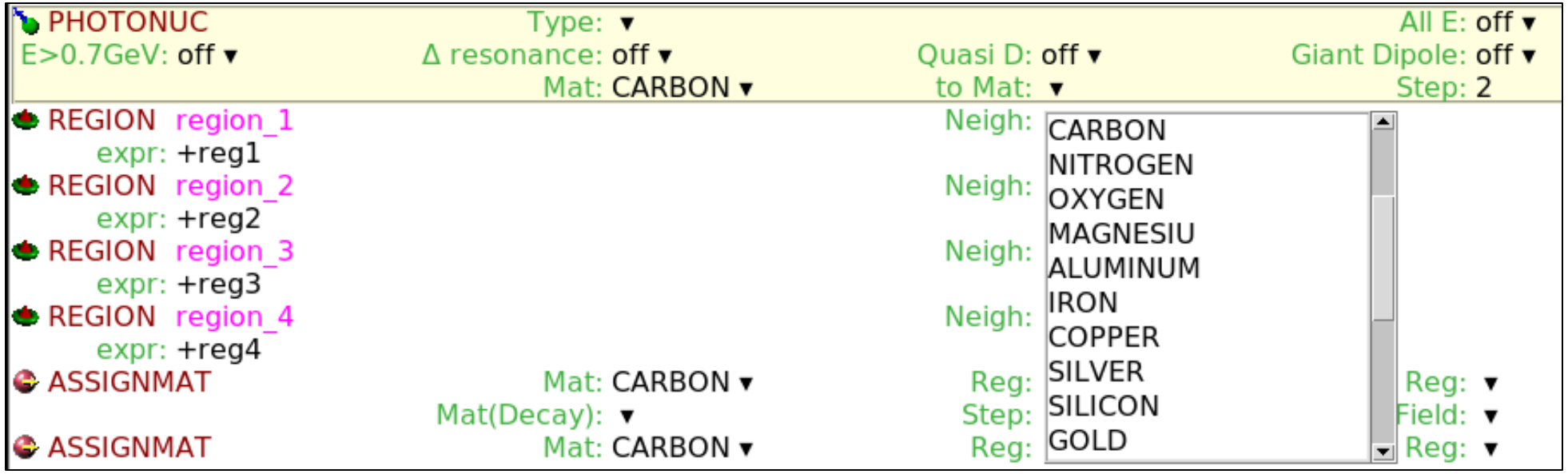
```
REGION  region_1                                   Neigh:
      expr: +reg1
REGION  region_2                                   Neigh:
      expr: +reg2
REGION  region_3                                   Neigh:
      expr: +reg3
REGION  region_4                                   Neigh:
      expr: +reg4
ASSIGNMAT                  Mat: CARBON ▾     Reg: region_1 ▾     to Reg: region_3 ▾
                     Mat(Decay):  ▾          Step: 2              Field:  ▾
```

```
REGION  region_1                                   Neigh:
      expr: +reg1
REGION  region_2                                   Neigh:
      expr: +reg2
REGION  region_3                                   Neigh:
      expr: +reg3
REGION  region_4                                   Neigh:
      expr: +reg4
ASSIGNMAT                  Mat: CARBON ▾     Reg: region_1 ▾     to Reg: region_4 ▾
                     Mat(Decay):  ▾          Step: 2              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.



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

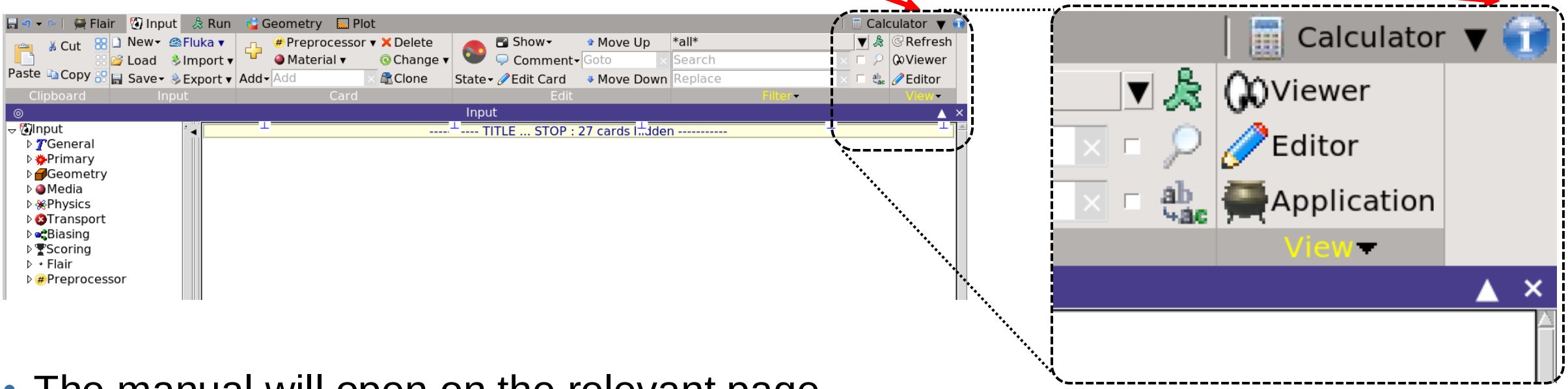
- Allows to assign a property to multiple “regions” (or whatever) in one single card
- The same concept applies to all other cases:

  materials, particles, lattices, etc.
- Special variables:
  - @LASTEREG 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 manual

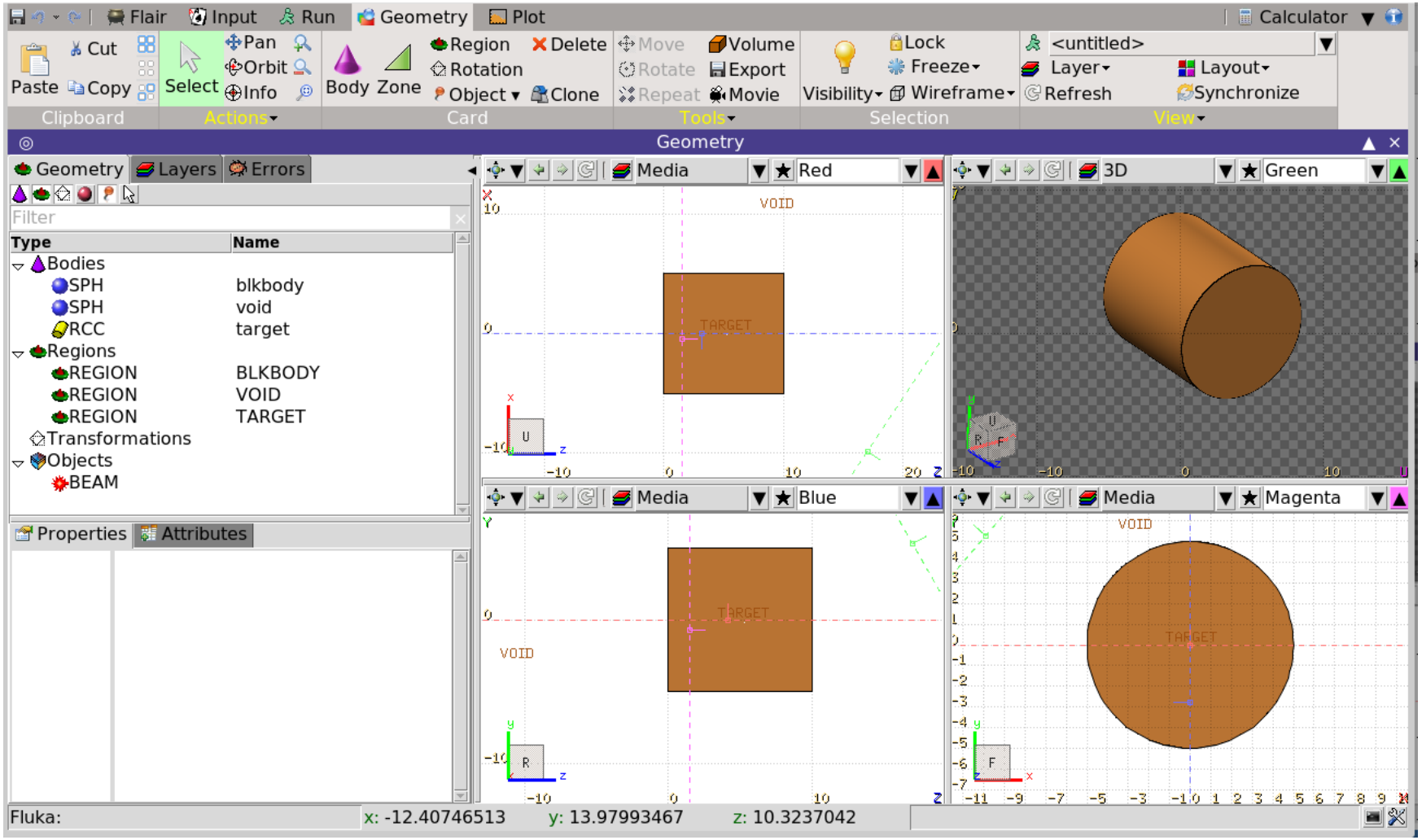
- Can be accessed using F1 button
- Can be accessed clicking on the “info” button




- The manual will open on the relevant page
- The manual is also available on the Fluka web page **`www.fluka.cern`**

# Geometry tab – 1

- Visualize and edit geometry
- Plot results
- Dedicated lectures



# Geometry tab – 2

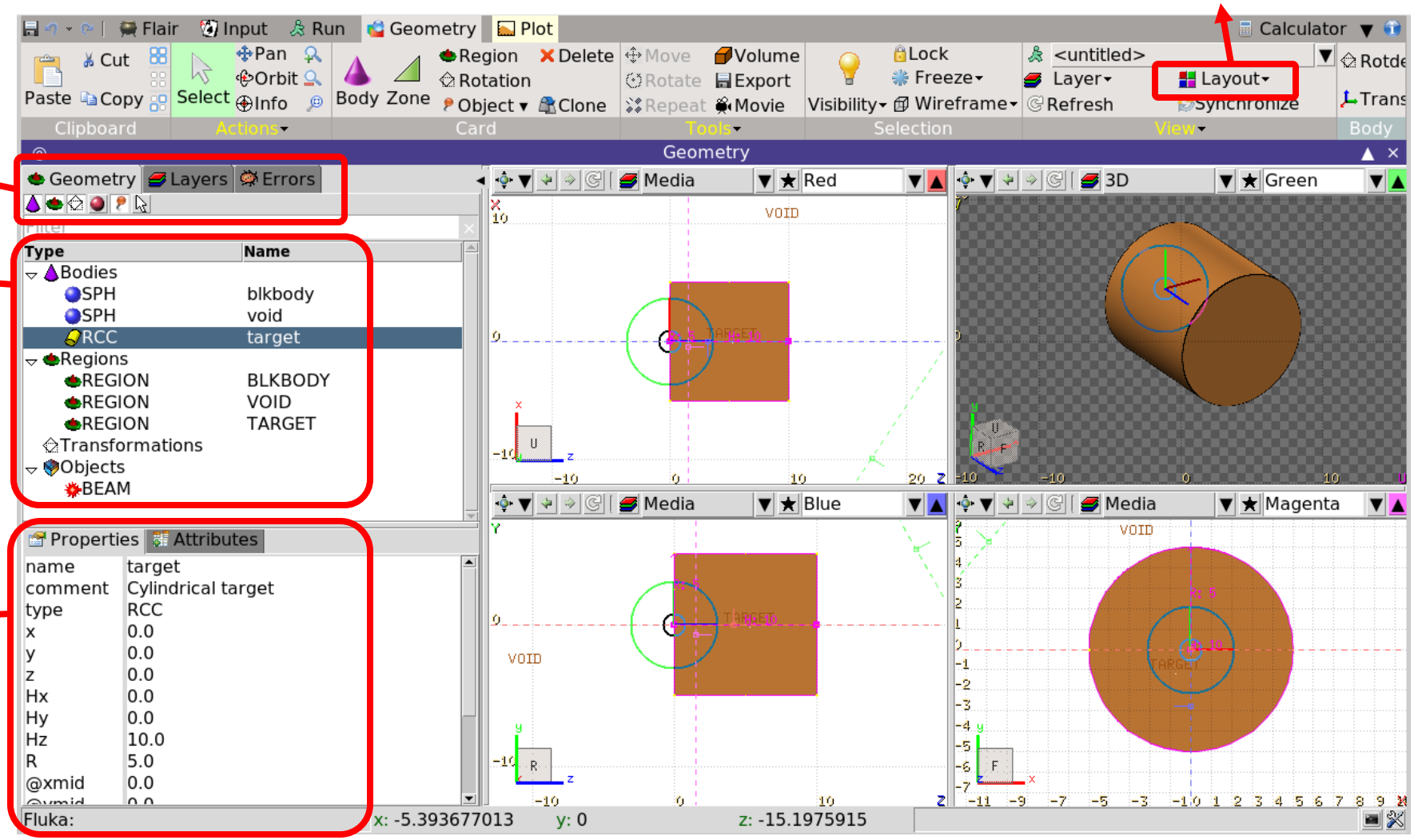
- Viewports automatically refreshed when input is changed

Layout management

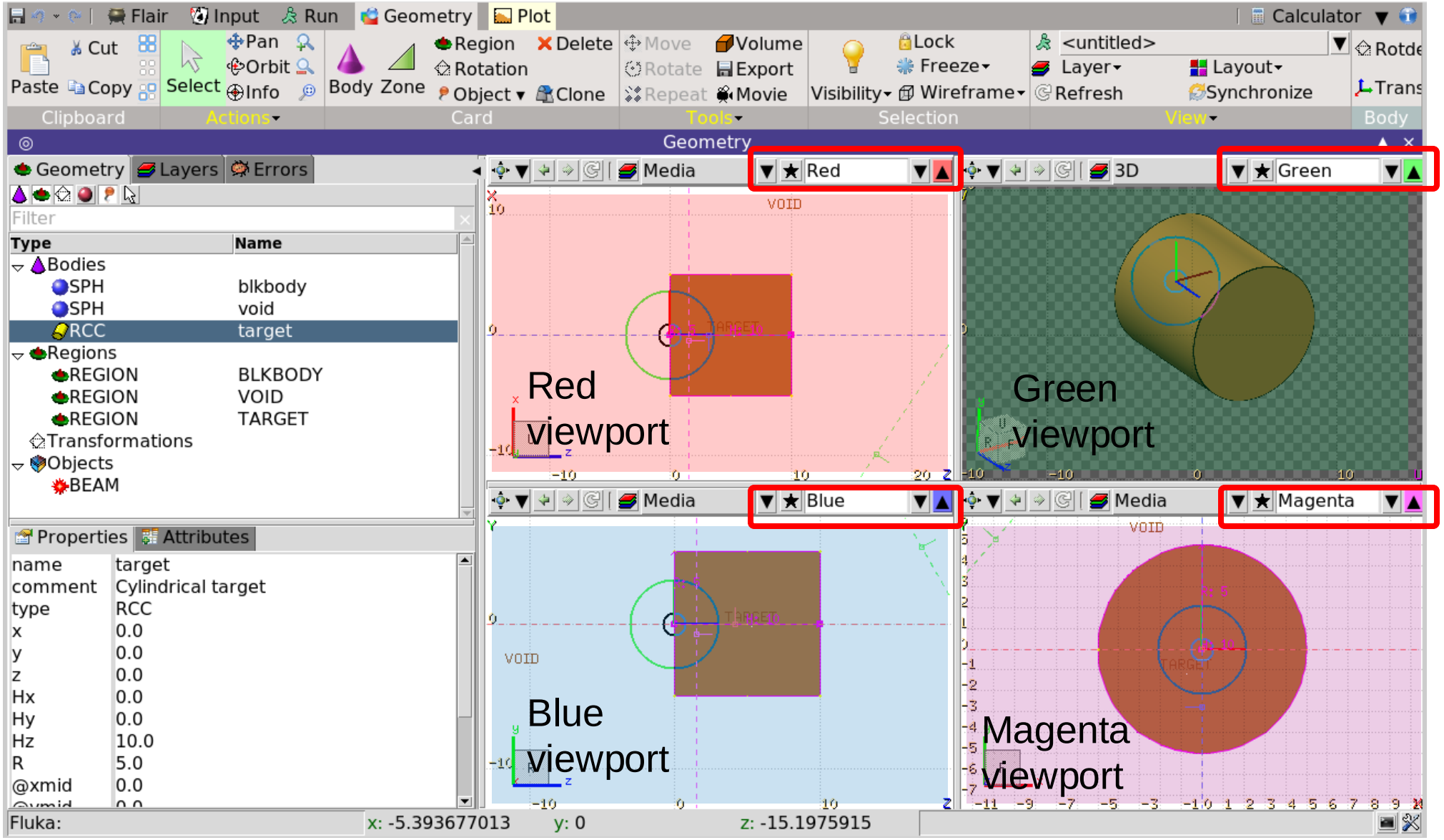
Filters

Objects Listbox

Properties & Attributes Listbox

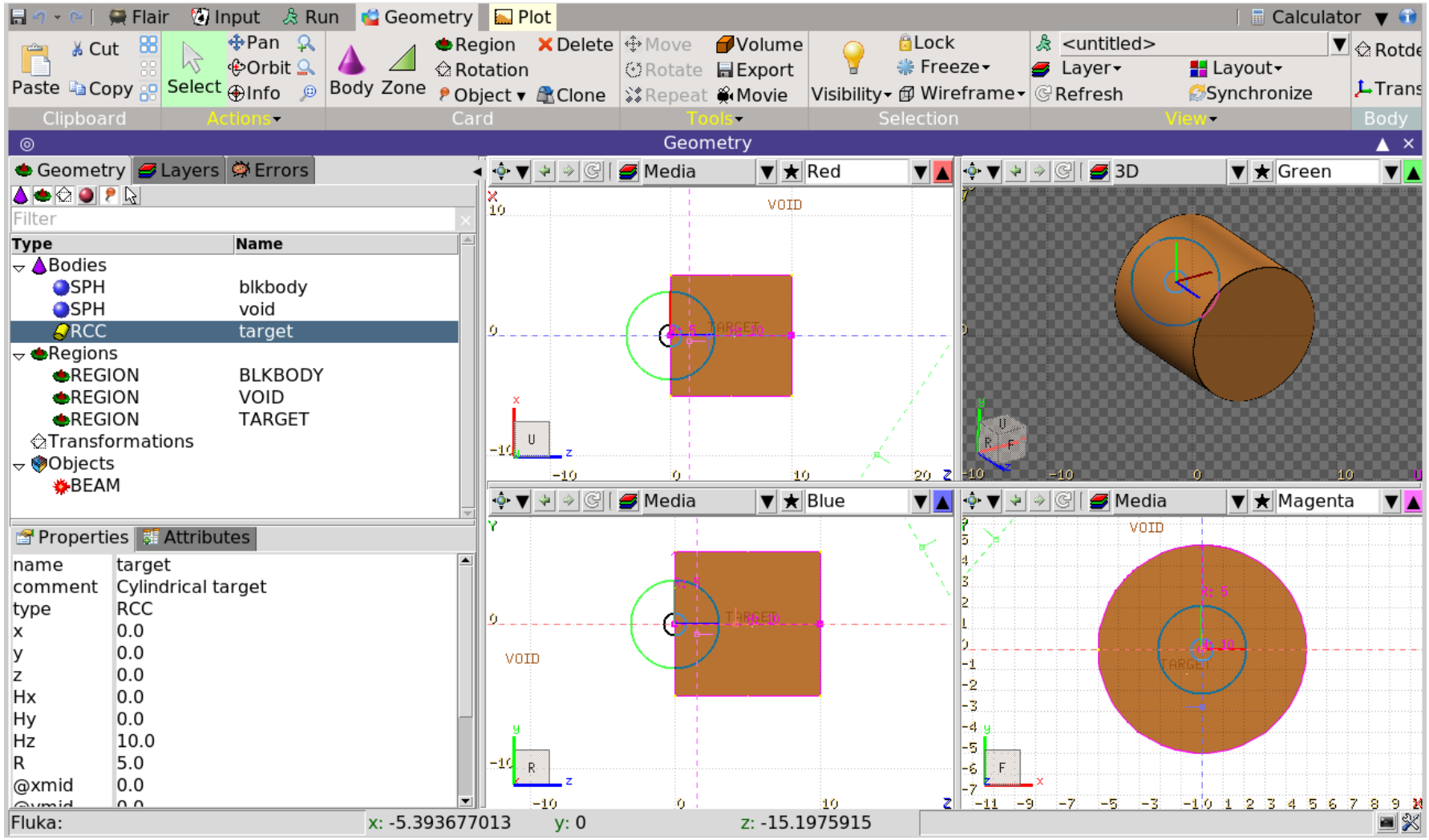


# Geometry tab – 3



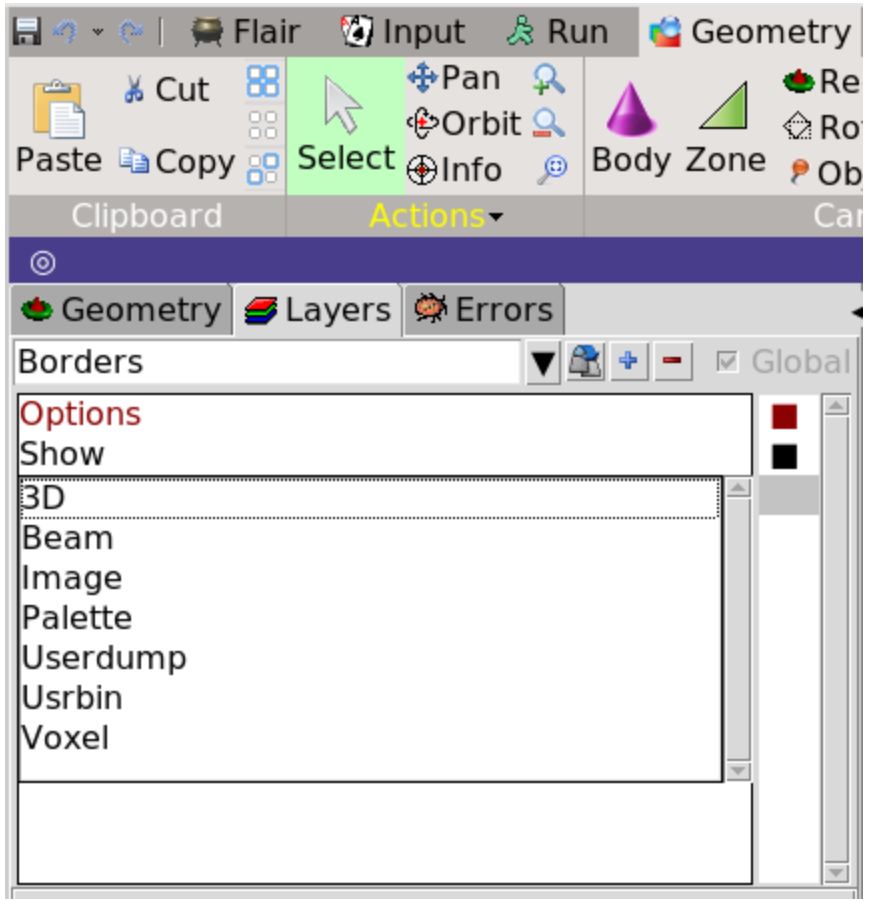
# Geometry tab – 4

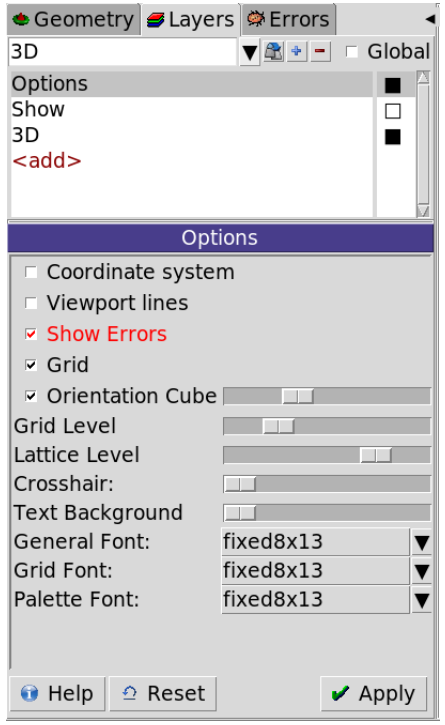
- Possible to navigate with mouse and keyboard (see dedicated lecture)

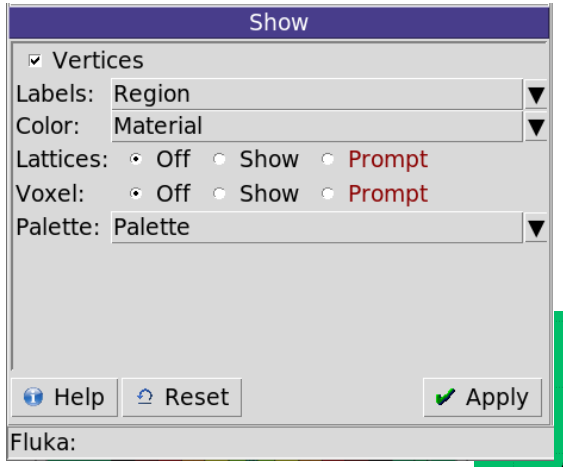


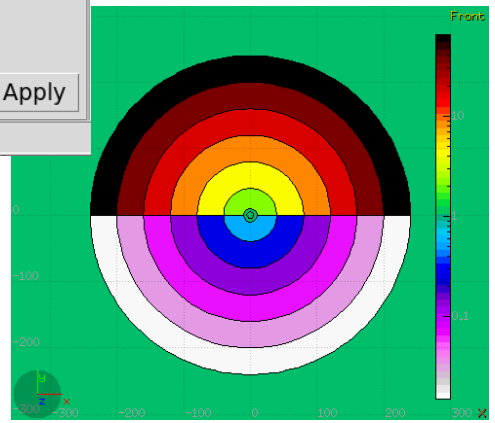
# Geometry tab – 5

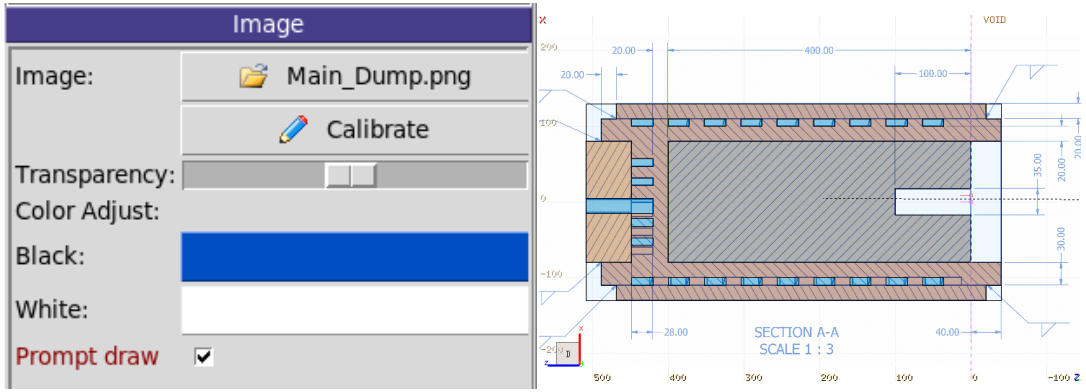
- Possible to add layers for better visualization:
  - Appearance (fonts, etc.)
  - Scoring (see Scoring-1 lecture)
  - Special quantities (e.g. region importance)
  - Background images (to help building geometry)



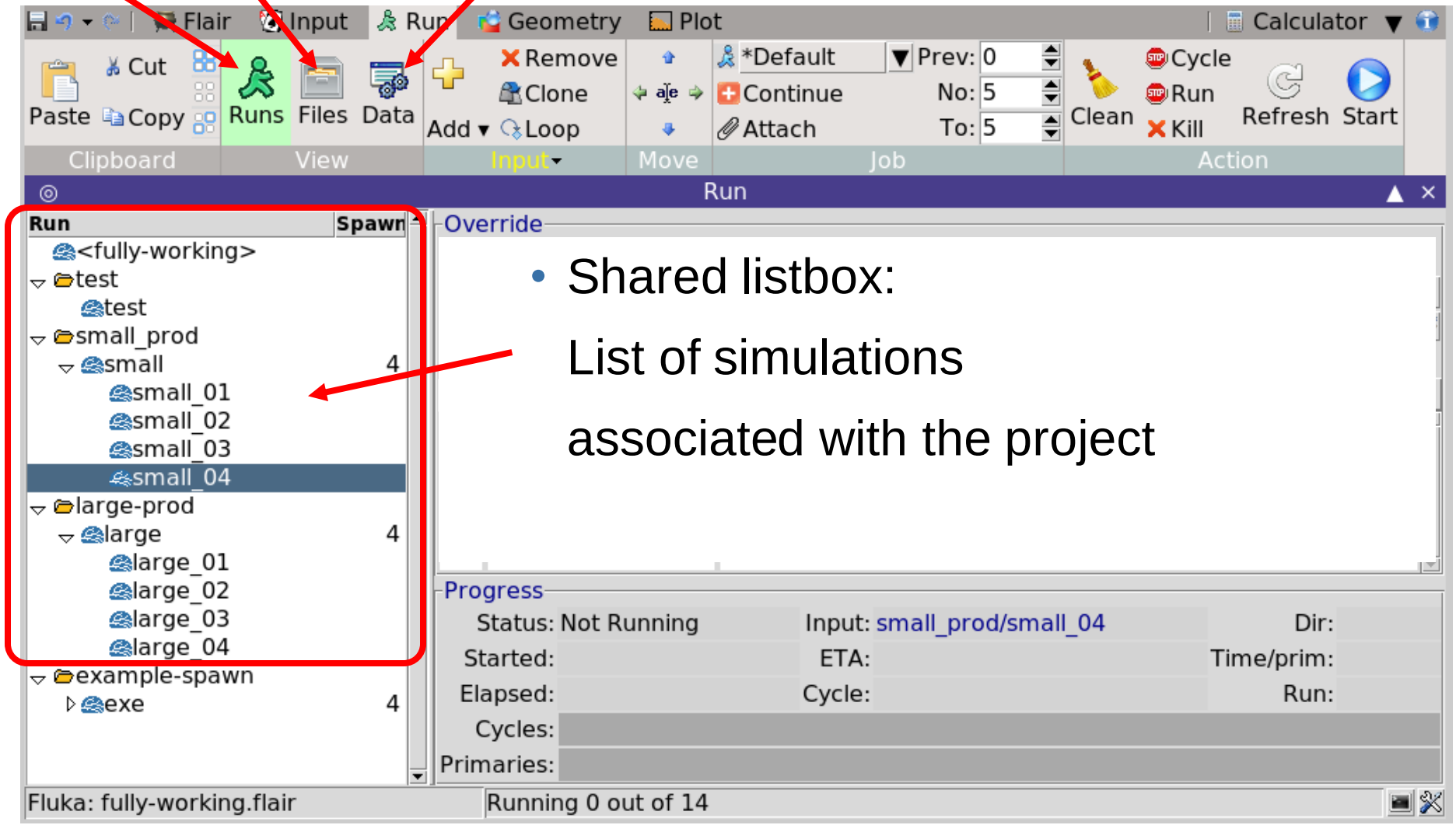





# Run tab

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



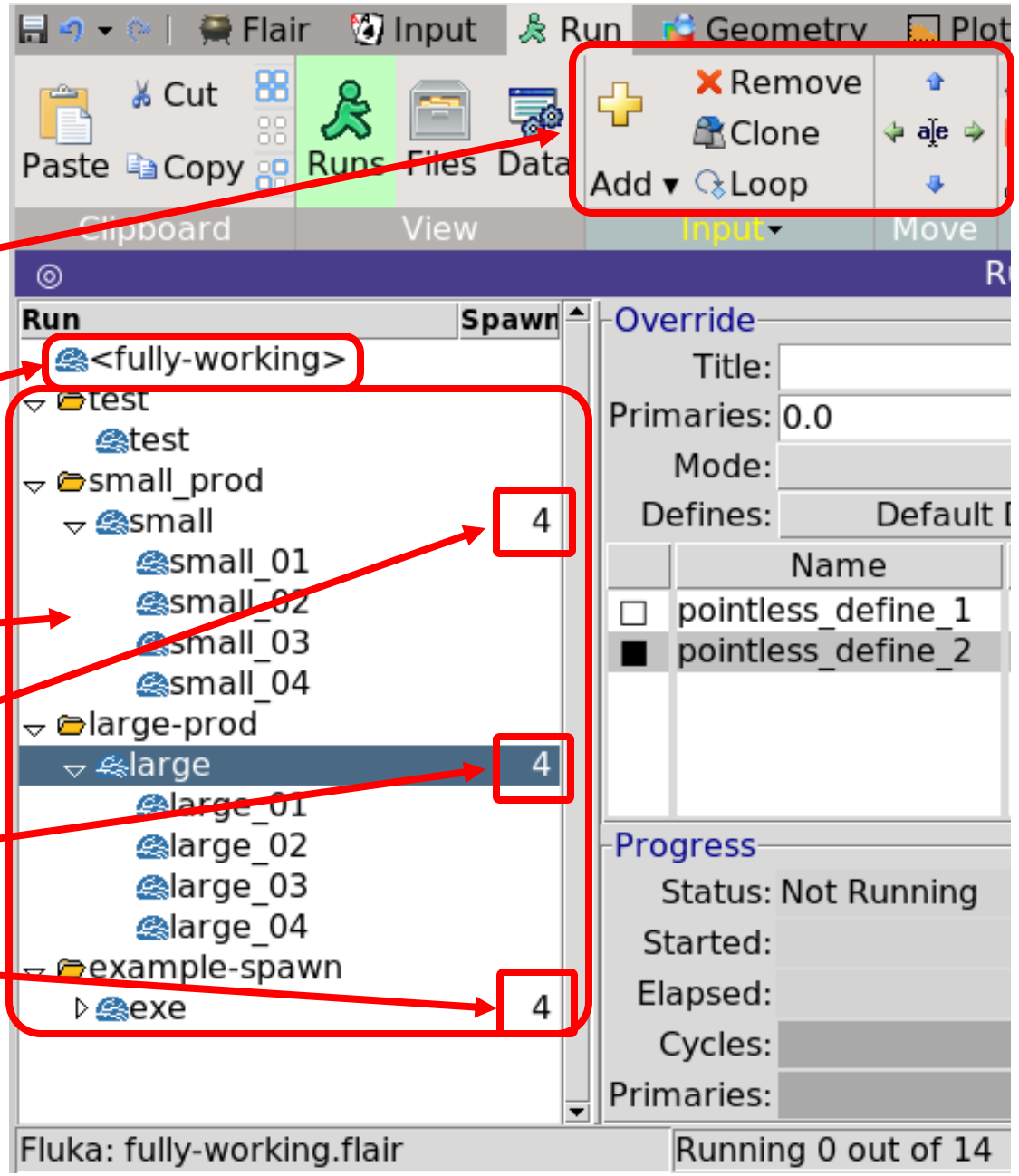

- Shared listbox:

  List of simulations

  associated with the project

# Run tab – Runs view – 1

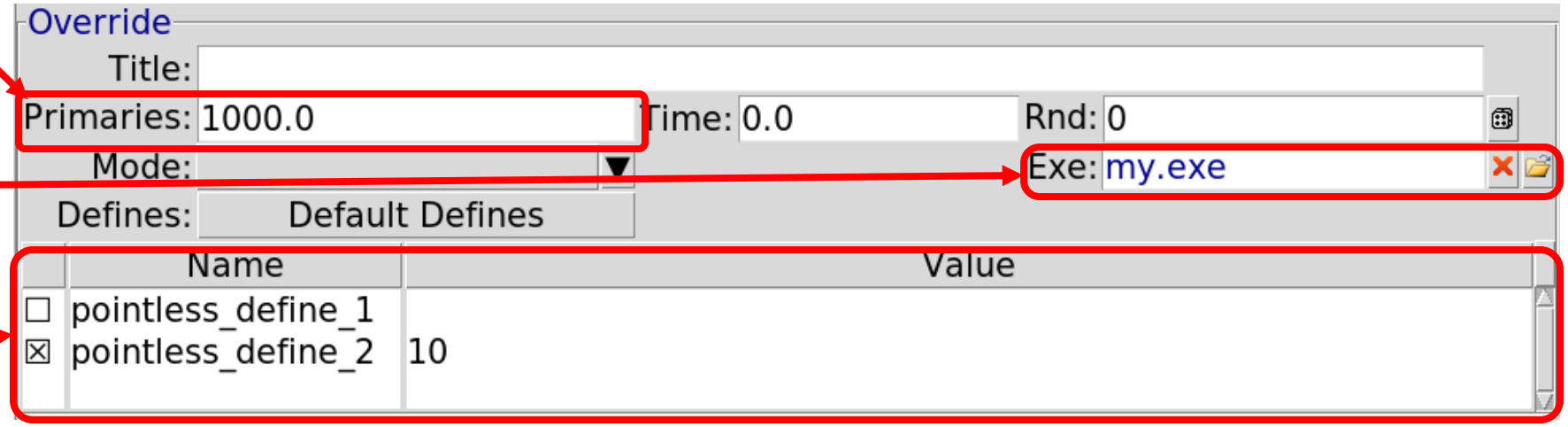
- Management of the various simulations
- Basic inputfile of the Flair project
- Different simulations associated with the Flair project
- Number of spawns



# Run tab – Runs view – 2

- Override of inputs
- Number of primaries
- Executable
- #define

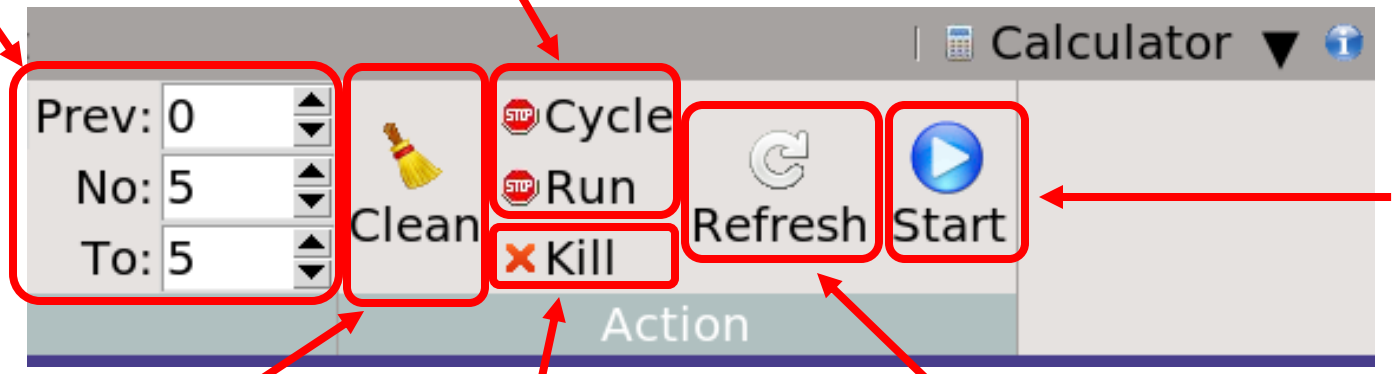
  Dedicated lecture



# Run tab – Runs view – 3

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

Cleanly stop the cycles/runs currently running



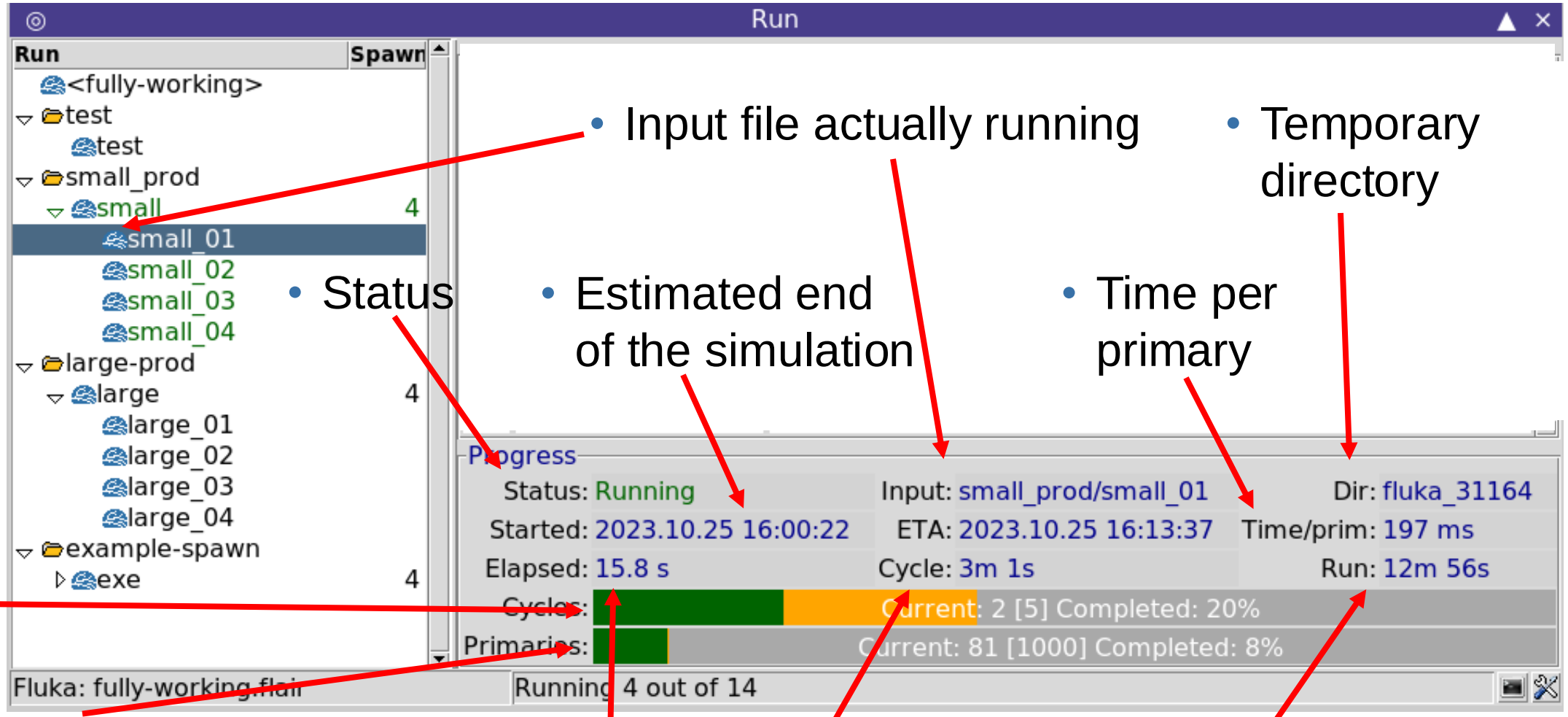

Start a simulation

Remove files from a previous simulation

Refresh the progress field

Kill the current simulations

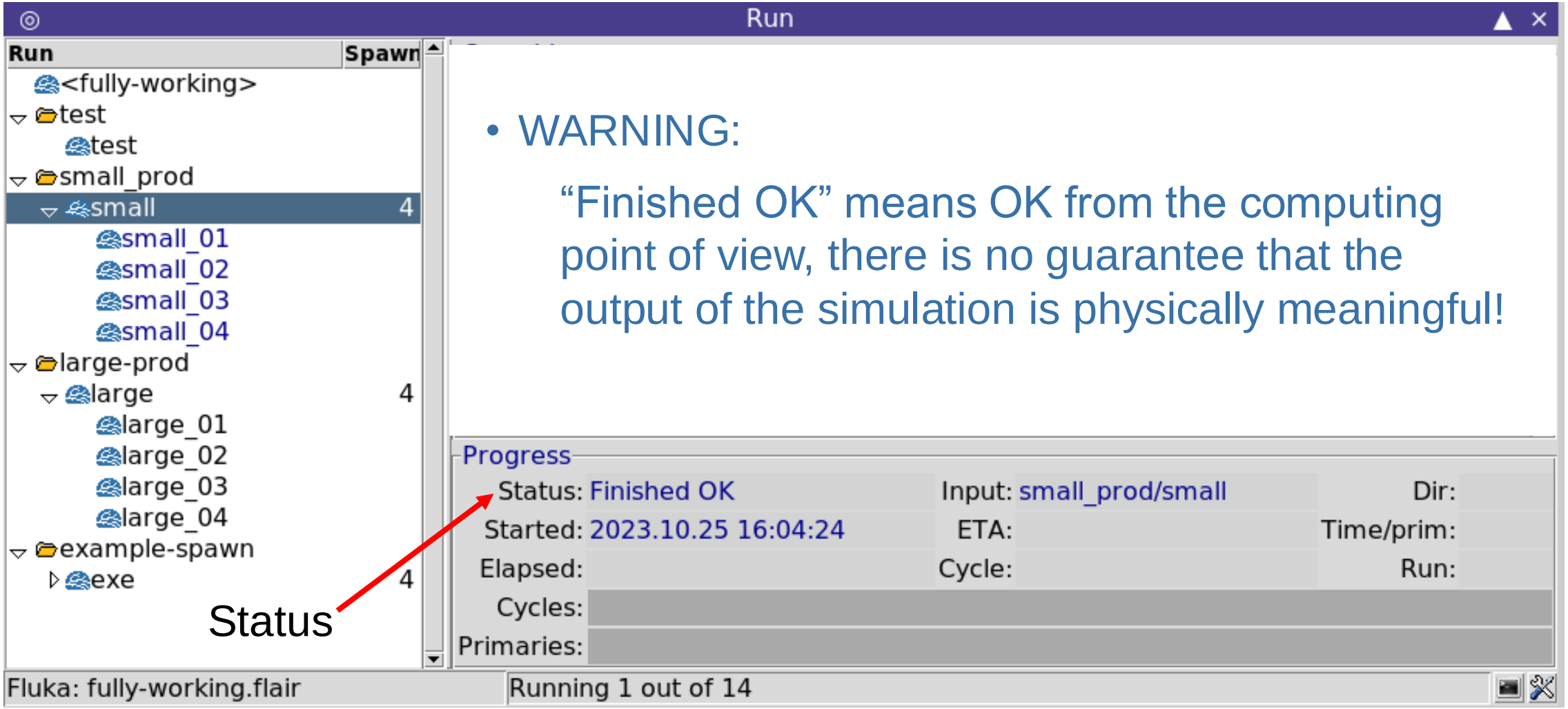
# Run tab – Runs view – 4




- 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 – 5

- At the end of the simulations…




- 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$
```

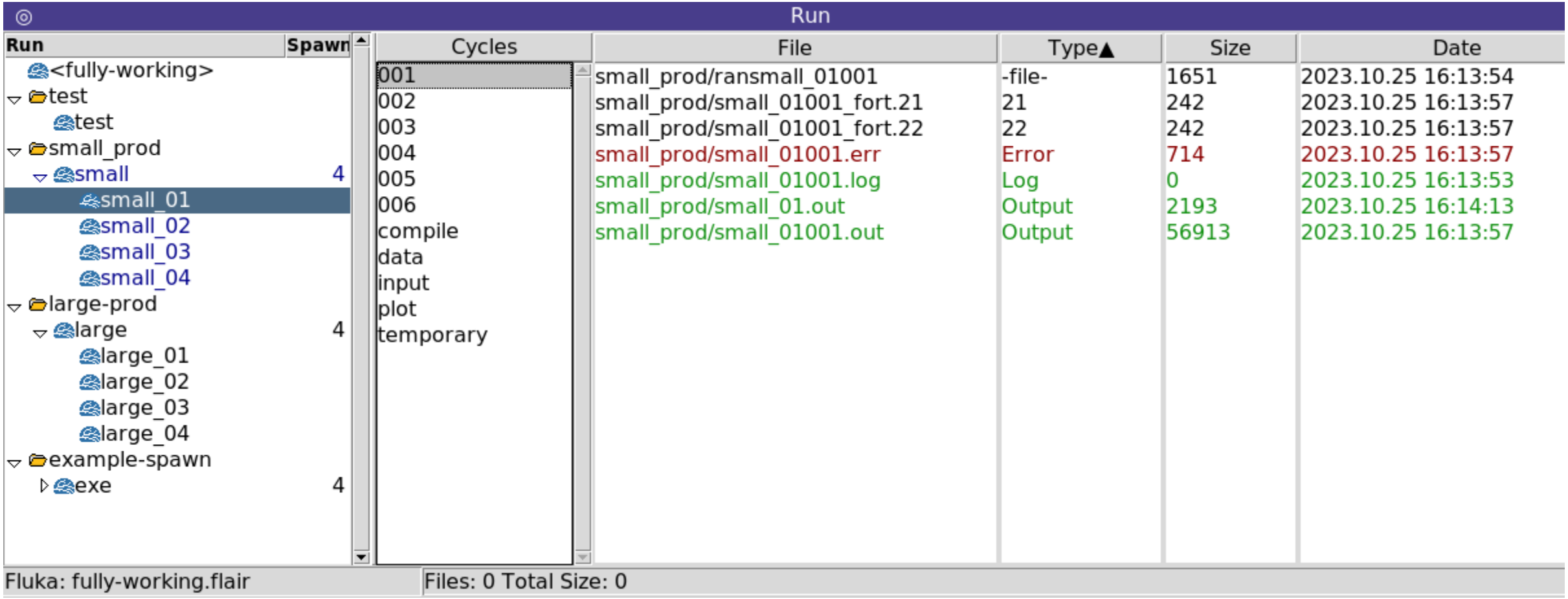
- Content of the working sub-directory

```
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.
ransmall_01004  small_01.out         small_01005.log      small_02004.
ransmall_01005  small_01001.err      small_01005.out      small_02004.
ransmall_01006  small_01001.log      small_01005_fort.21  small_02004_
ransmall_02001  small_01001.out      small_01005_fort.22  small_02004_
ransmall_02002  small_01001_fort.21  small_02.inp         small_02005.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$
```

`.inp` and `.out` files specific of each spawn

# Run tab – Files view – 1

- Generated files accessible via the Files view



# 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_fort.21`** |
| **`small_01001.log`** | **`small_01001_fort.22`** |
| **`small_01001.out`** | **`small_01.out`** |
| **`ransmall_01001`** | |

# 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_fort.21`** |
| **`small_01001.log`** | **`small_01001_fort.22`** |
| **`small_01001.out`** | **`small_01.out`** |
| **`ransmall_01001`** ← | 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 |
| 002 | small_prod/small_01001_fort.21 | 21 | 242 |
| 003 | small_prod/small_01001_fort.22 | 22 | 242 |
| 004 | small_prod/small_01001.err | Error | 714 |
| 005 | small_prod/small_01001.log | Log | 0 |
| 006 | small_prod/small_01.out | Output | 2193 |
| compile | small_prod/small_01001.out | Output | 56913 |
| data | | | |
| input | | | |
| plot | | | |
| temporary | | | |

- Spawn 2 Cycle 1

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

# Run tab – Files view – 7

- Spawn 1 Cycle 1

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

- Spawn 1 Cycle 6

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

- Random file for the next cycle is generated

# Run tab – Data view – 1

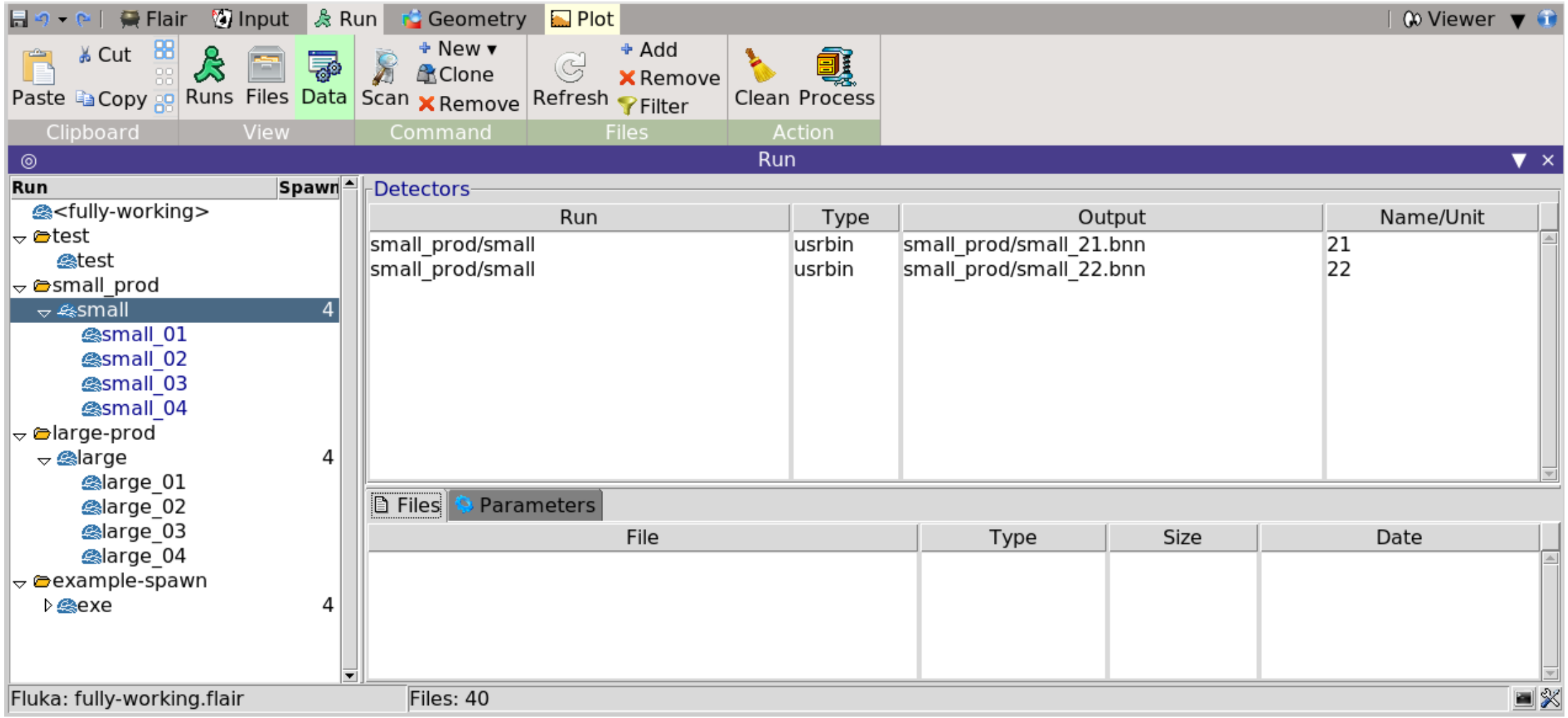
- All the generated files need to be merged to be analyzed

```
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
```

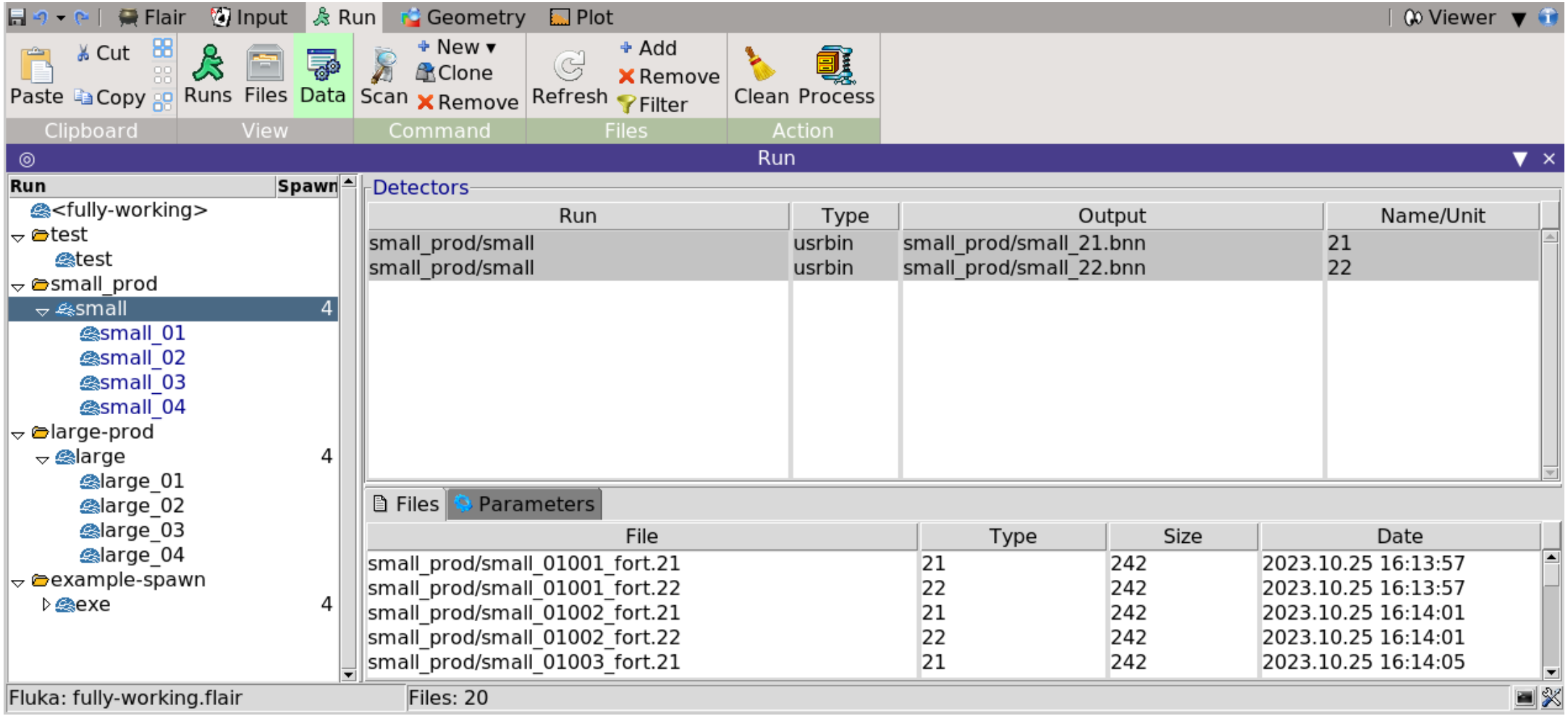
# Run tab – Data view – 2

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



# Run tab – Data view – 3

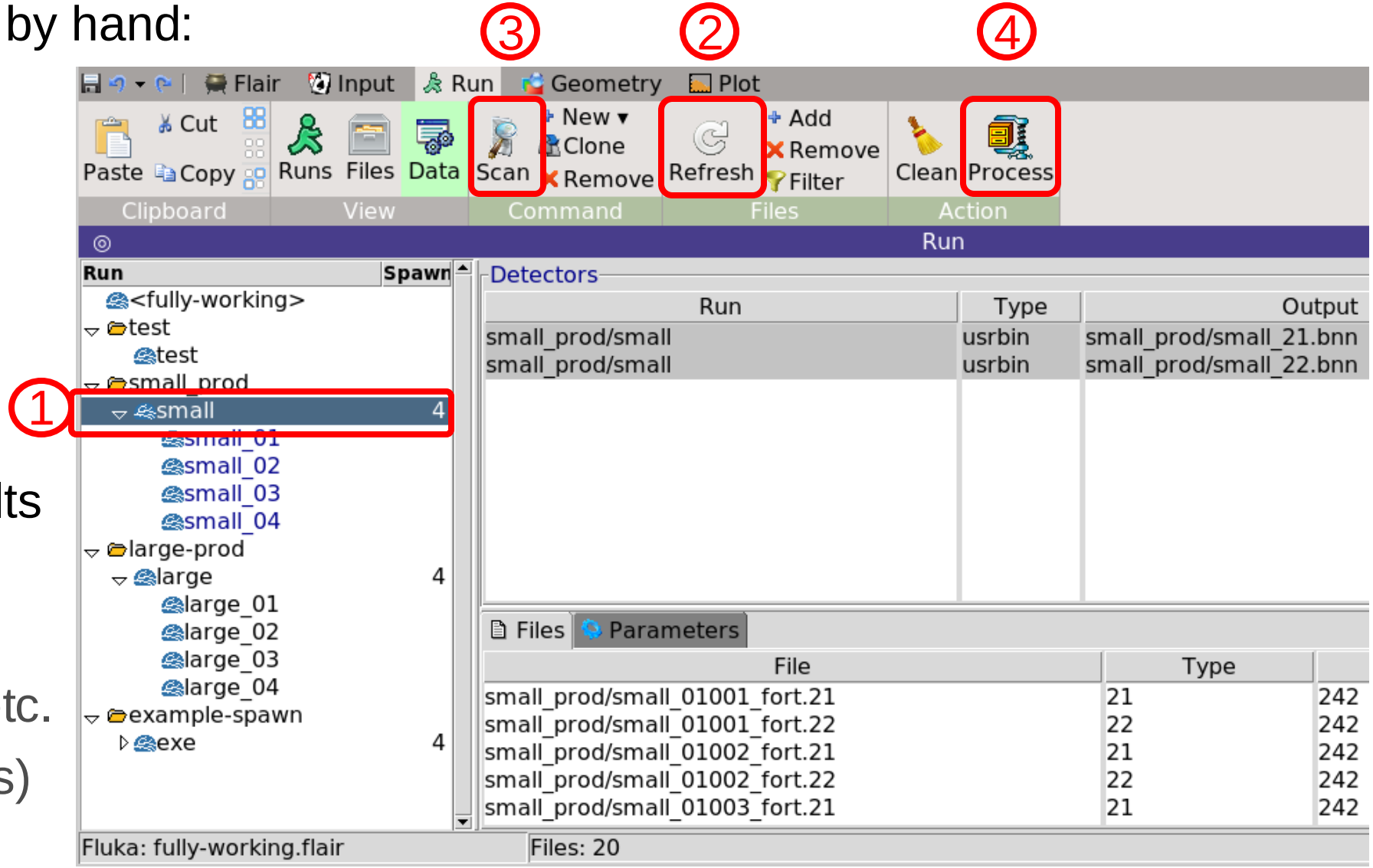
- Flair finds all the corresponding file (per spawn and per cycle)



# Run tab – Data view – 4

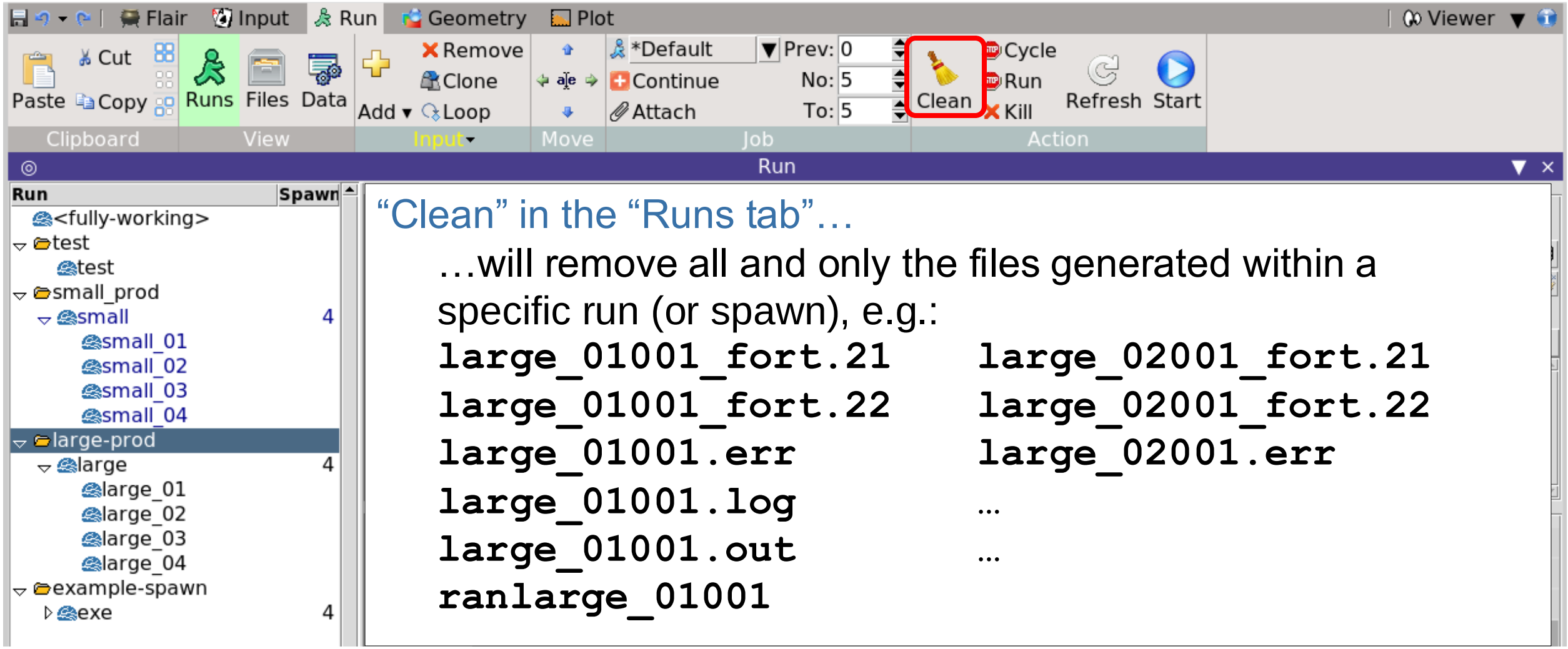
- Process can be forced by hand:
- 1-Select the run
- 2-Refresh
- 3-Scan
- 4-Process (merge)

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



# Run tab – Cleaning – 1

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



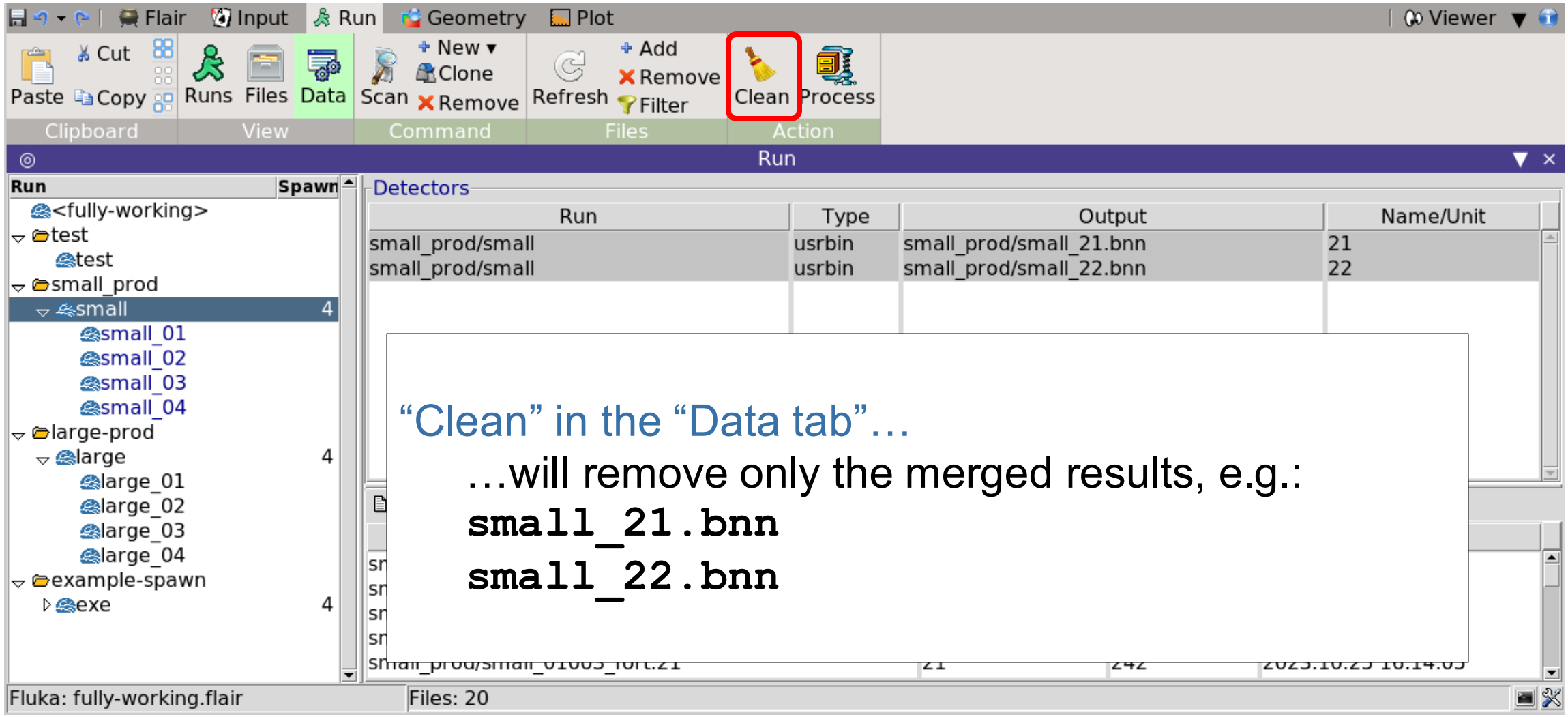

"Clean" in the "Runs tab"…

…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!




"Clean" in the "Data tab"…

…will remove only the merged results, e.g.:

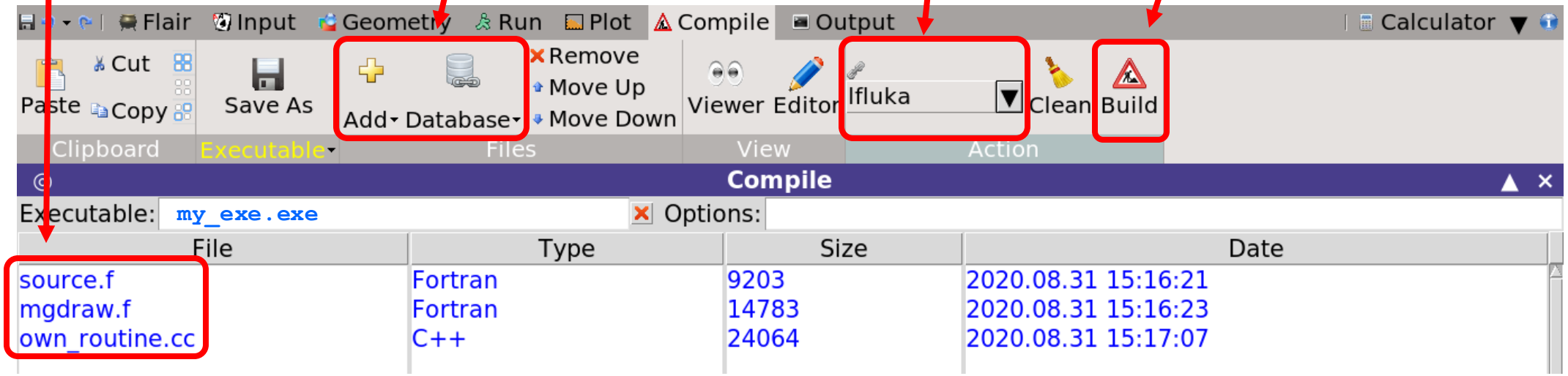
**`small_21.bnn`**

**`small_22.bnn`**

# Compile tab

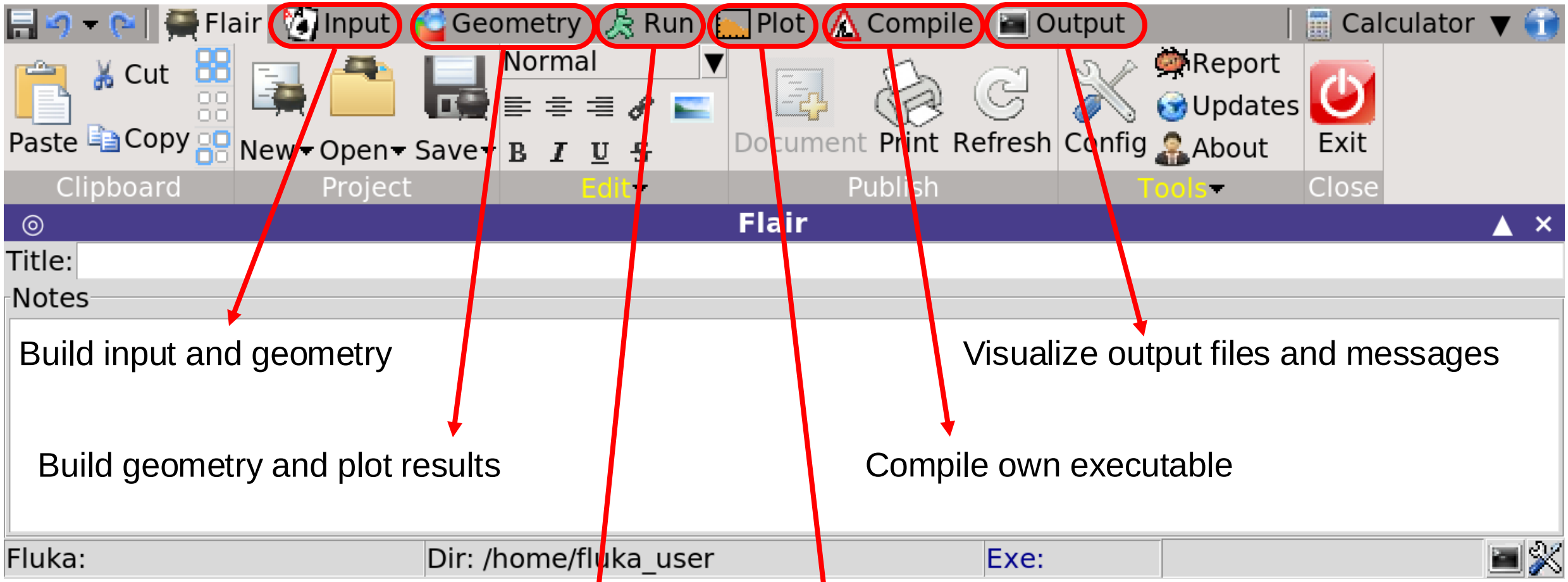
- Only very basic information is given here

- Routine to be compiled
- Add routines
- Select linker
- Build executable




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

# Do you remember slide 6?




Build input and geometry

Visualize output files and messages

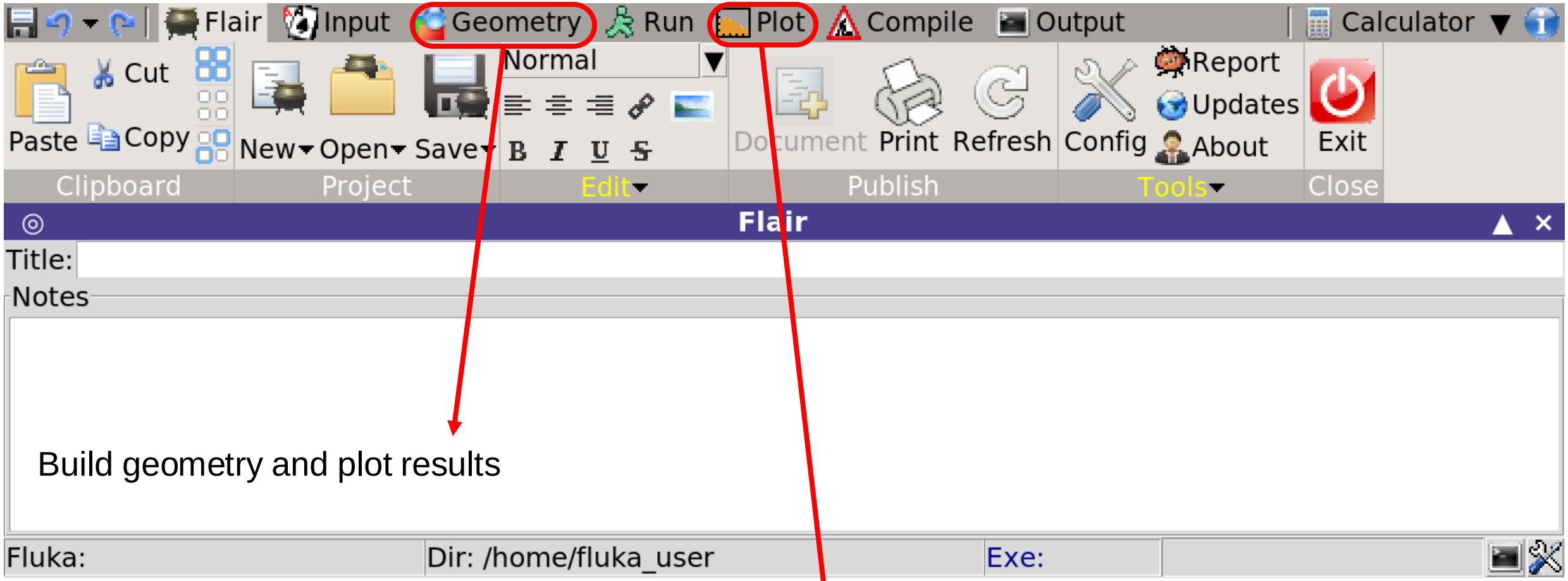
Build geometry and plot results

Compile own executable

Run and merge results

Plot results

# Do you remember slide 6?

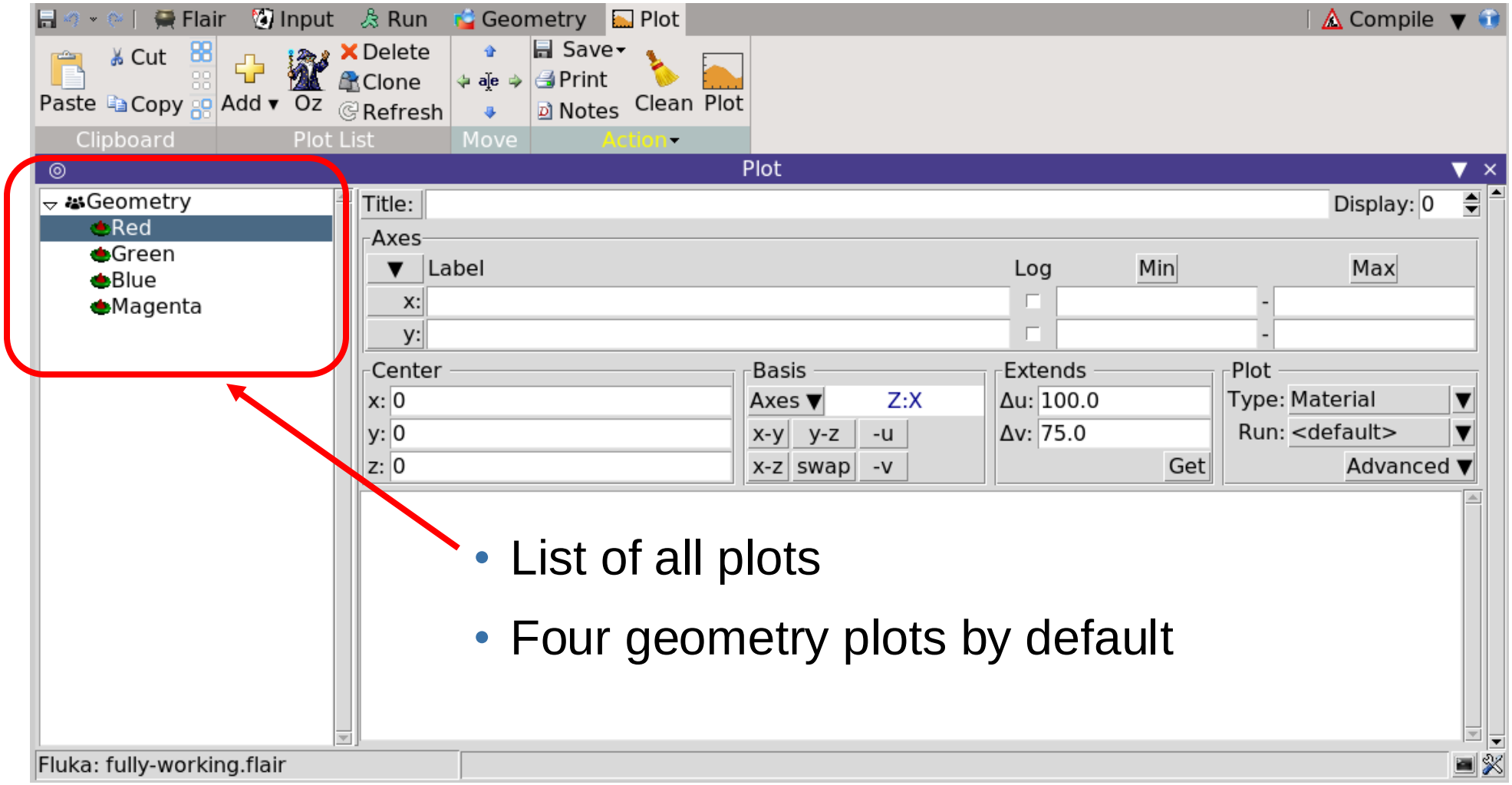



Build geometry and plot results

Plot results

# Plotting results in the Plot tab – 1

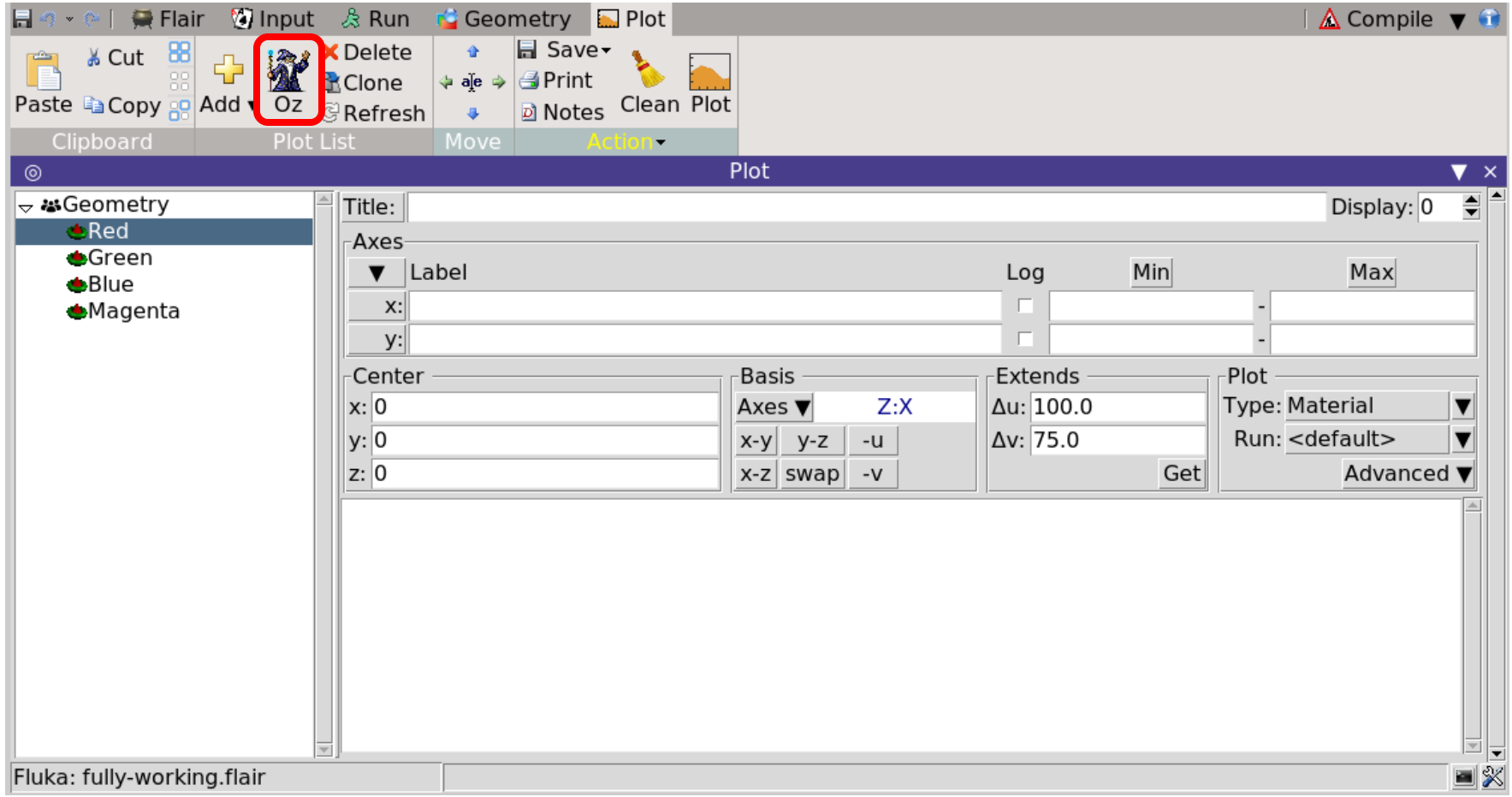
- Possible to plot geometry and all built-in scorings results




- List of all plots
- Four geometry plots by default

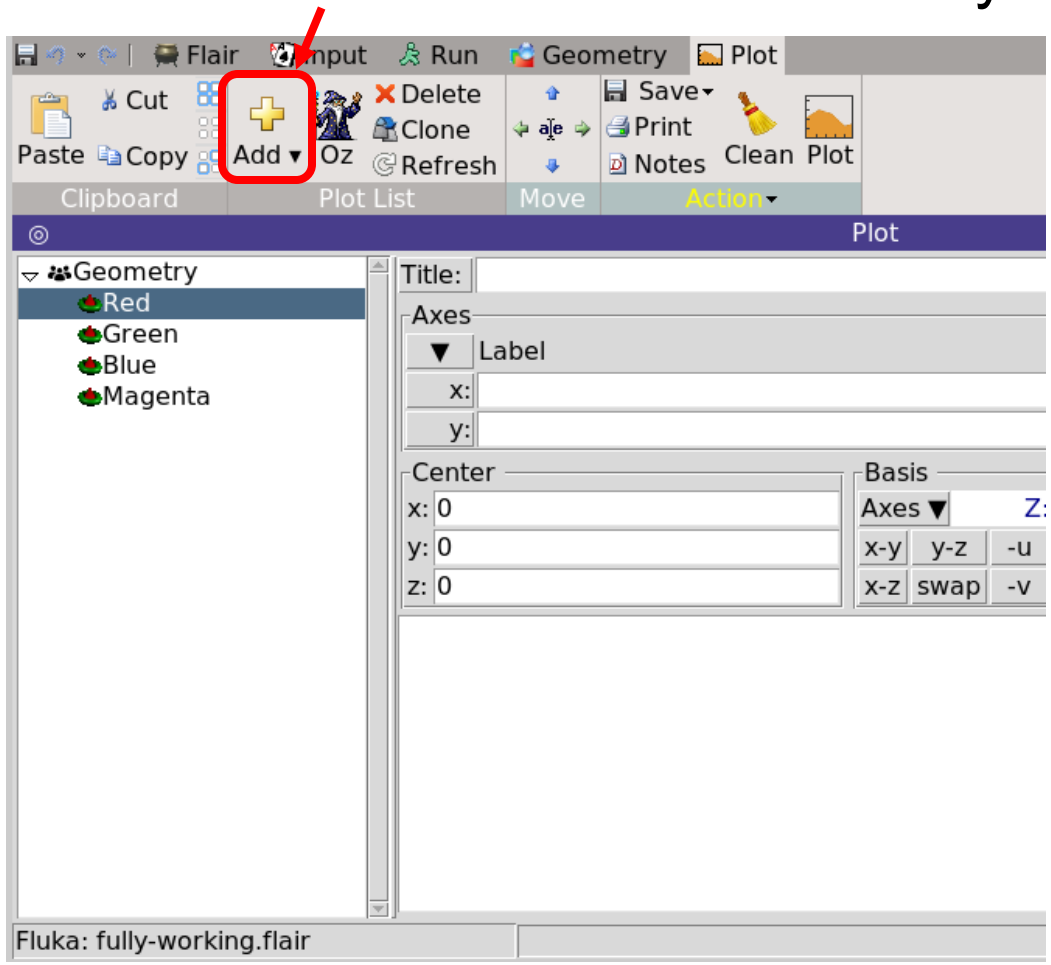
# Plotting results in the Plot tab – 2

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

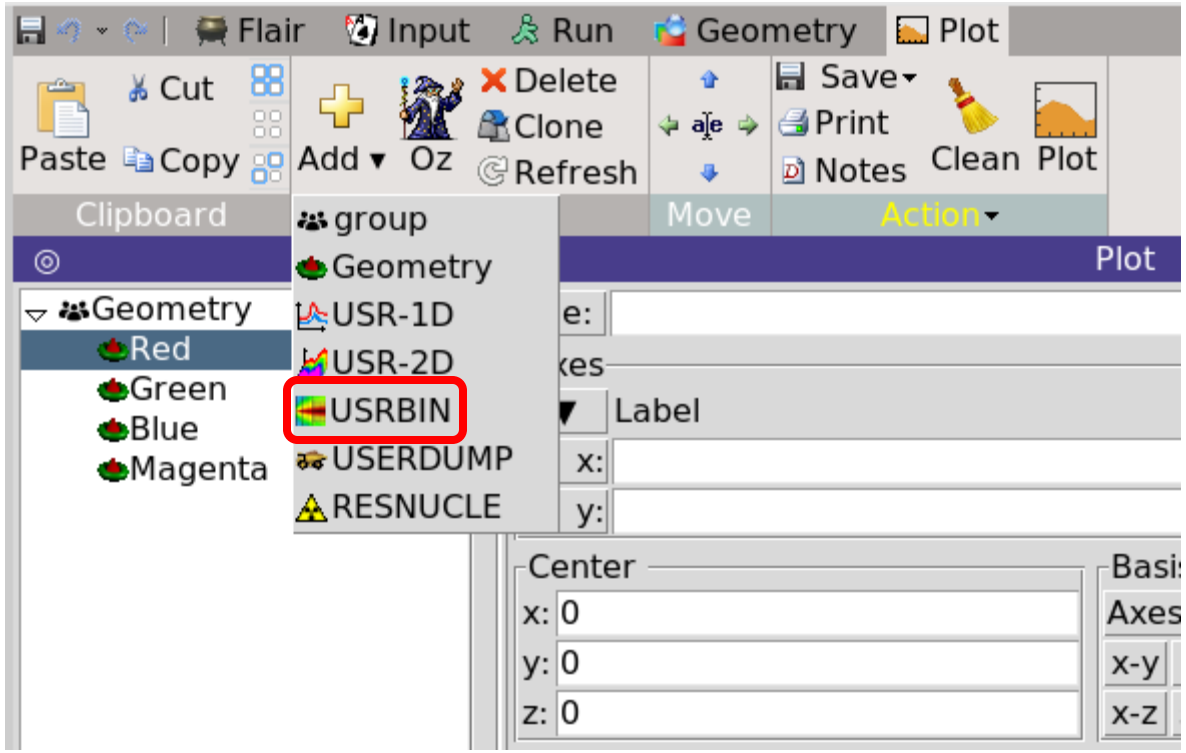


# Plotting results in the Plot tab – 3

- 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

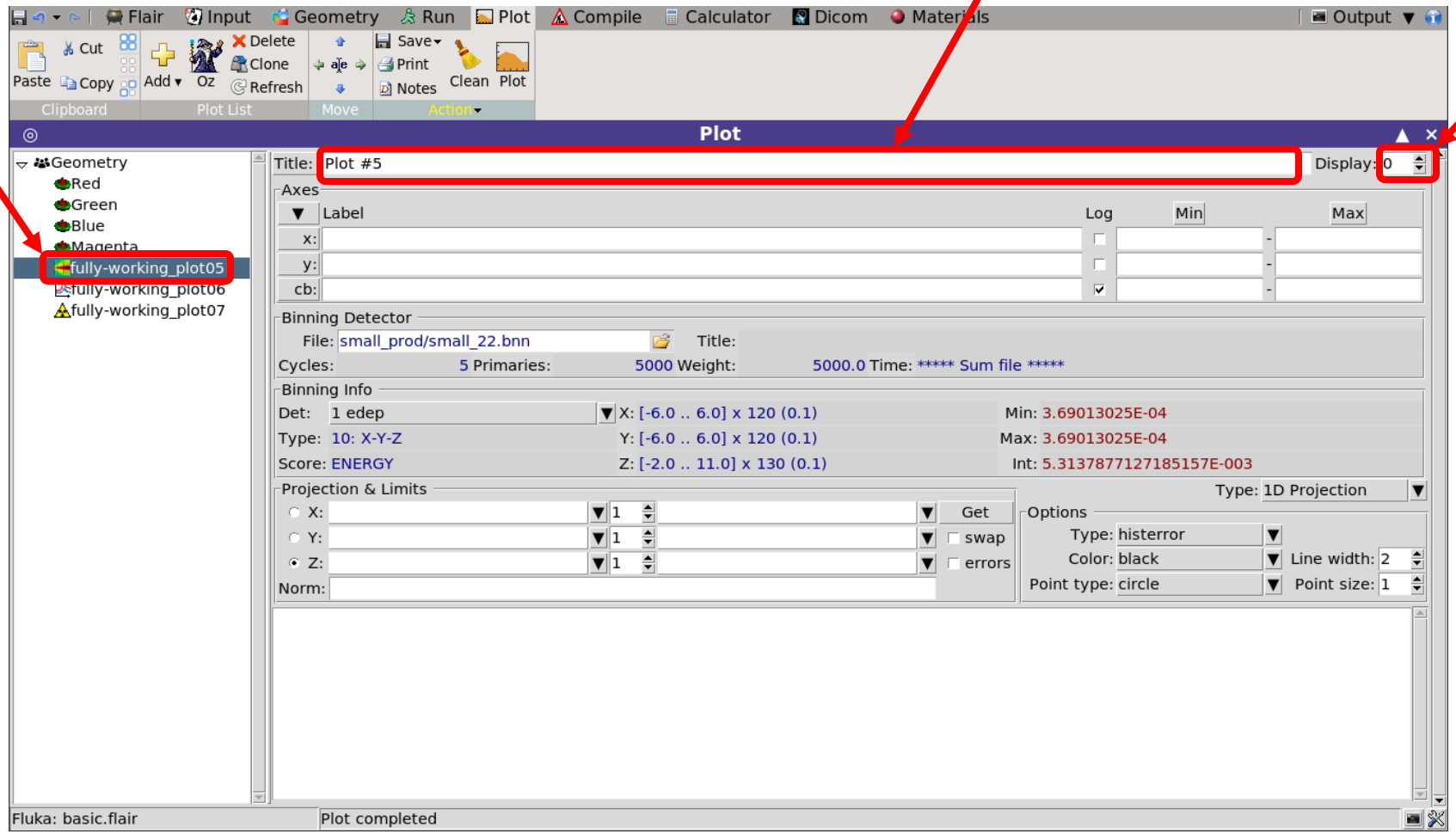



(here, we’ll see only USRBIN)



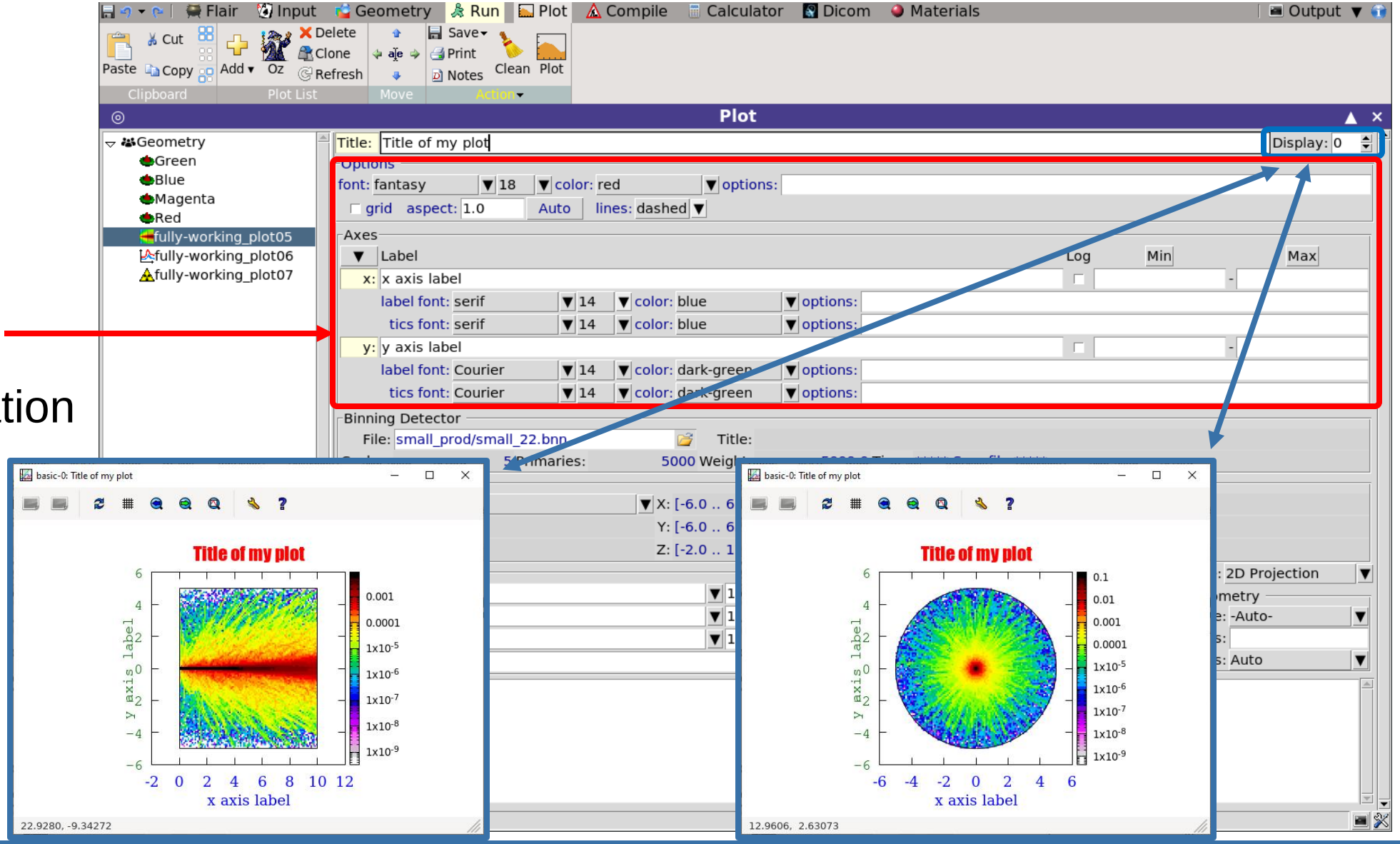
# Plotting results in the Plot tab – 4

- Name of the file that will be saved
- Title of the plot
- Display ID



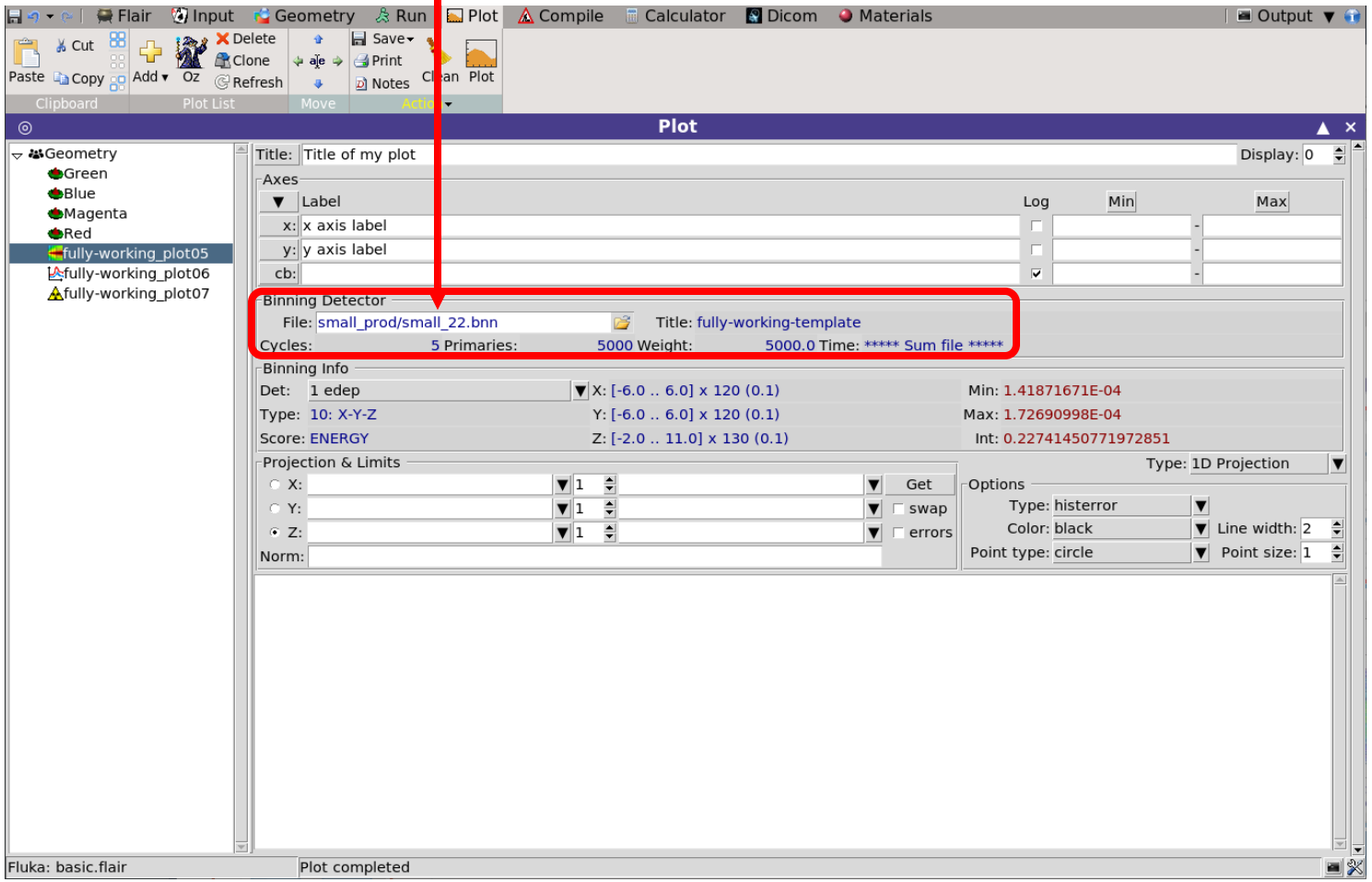
# Plotting results in the Plot tab – 5

- Plenty of options for plot customization



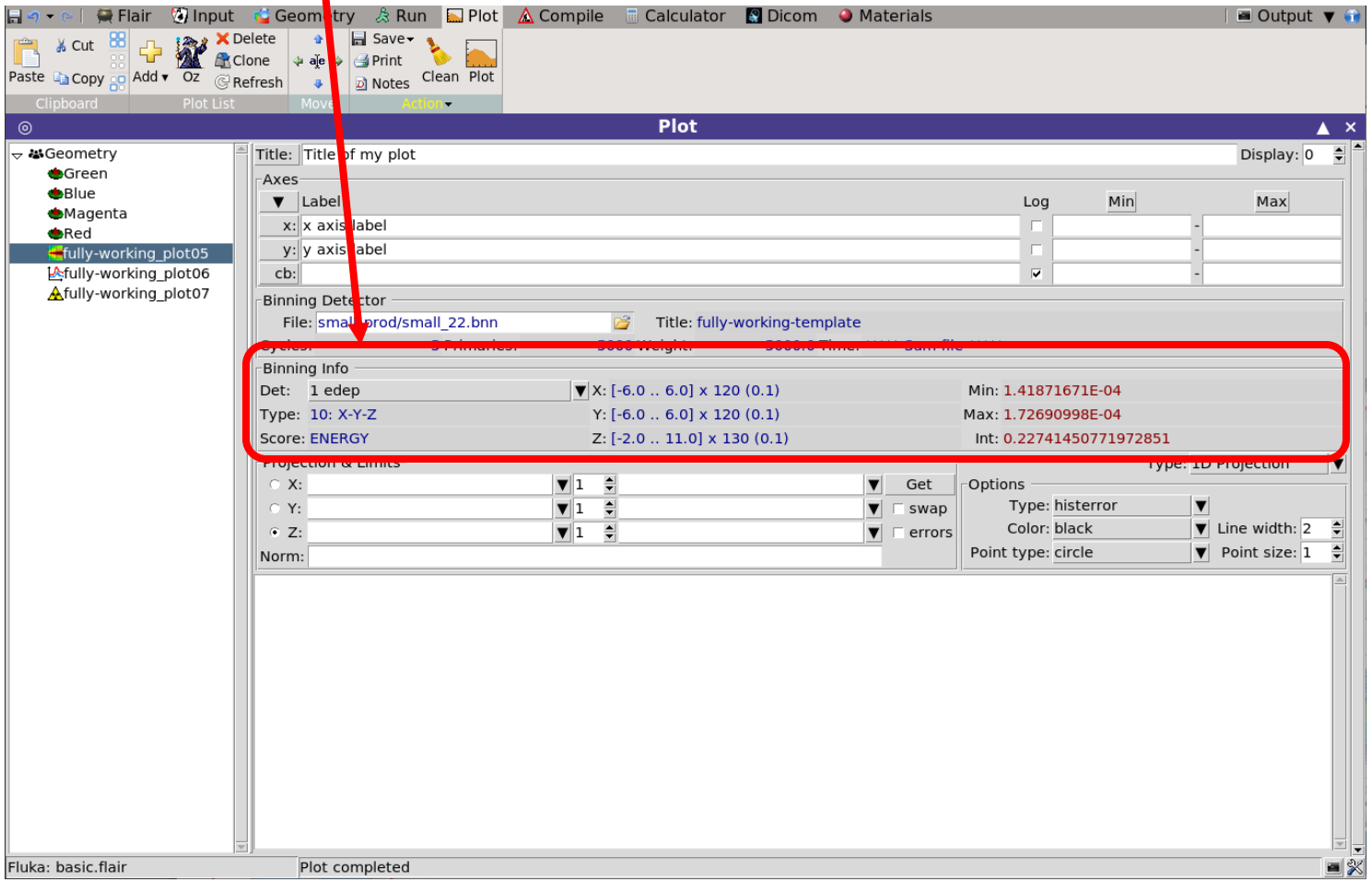
# Plotting results in the Plot tab – 6

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



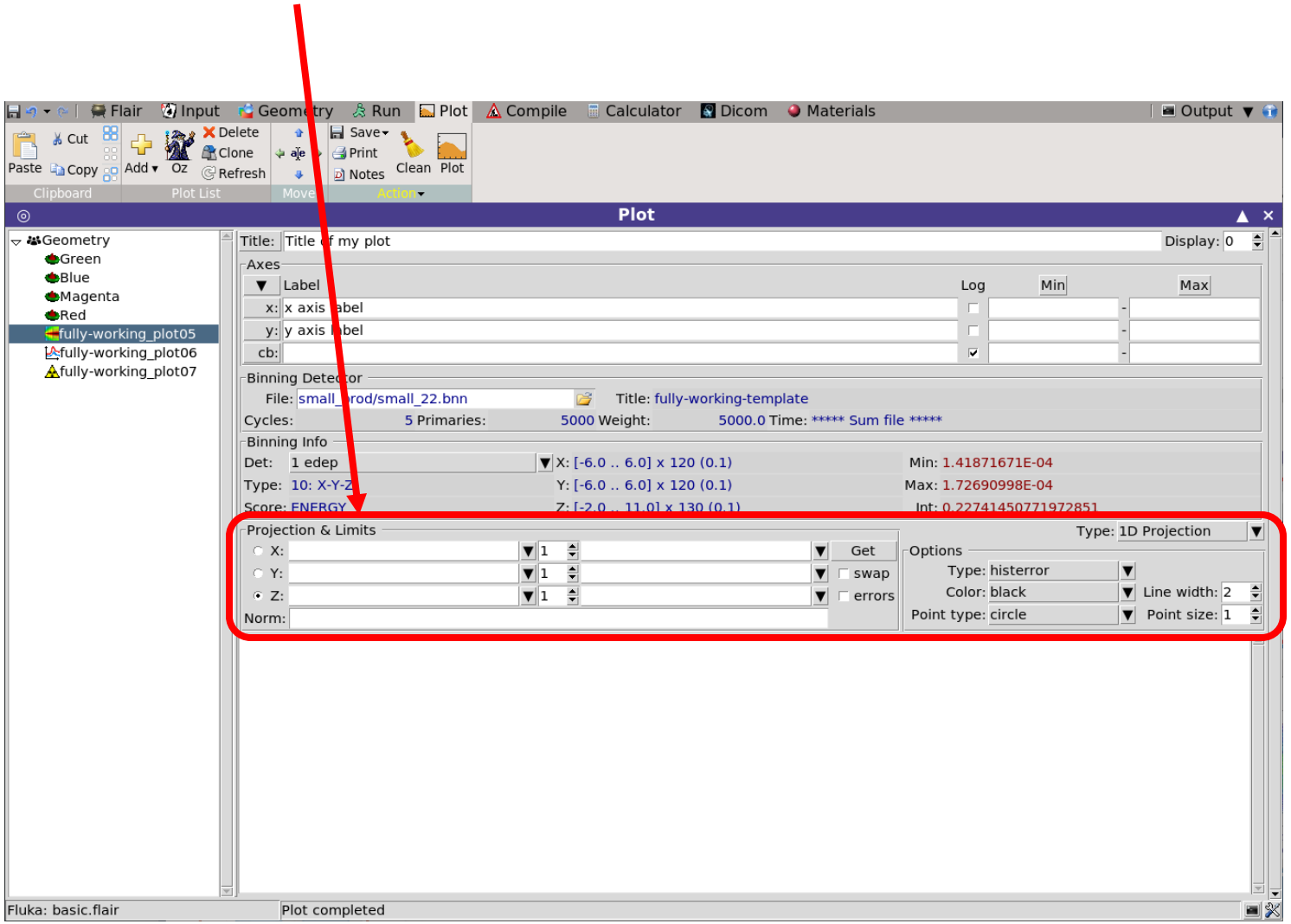
# Plotting results in the Plot tab – 7

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



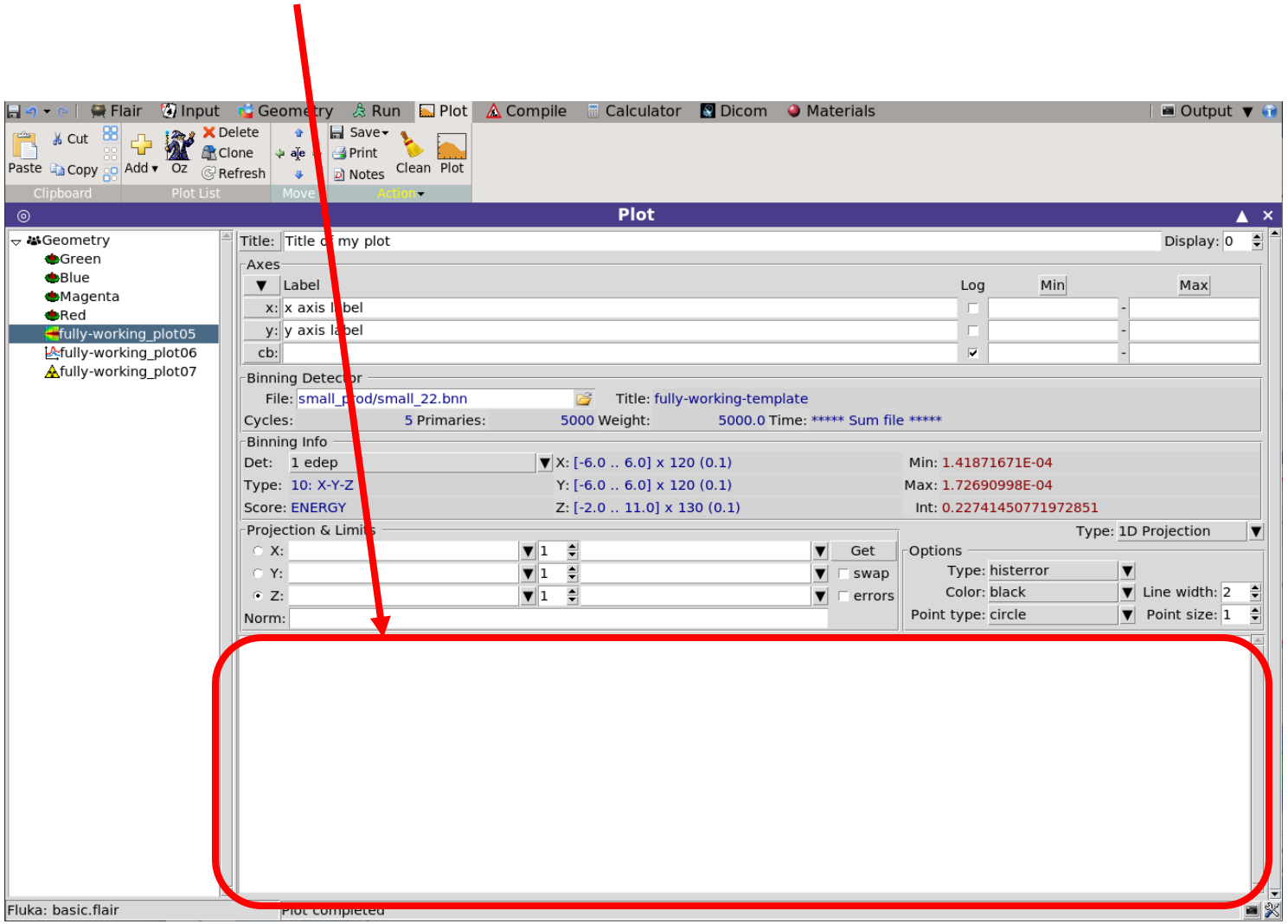
# Plotting results in the Plot tab – 8

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

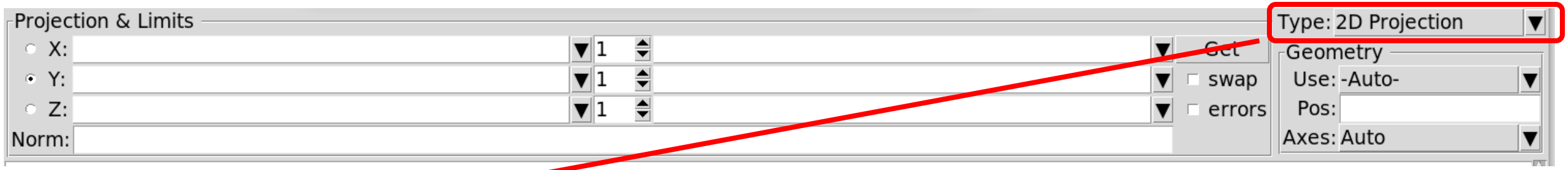


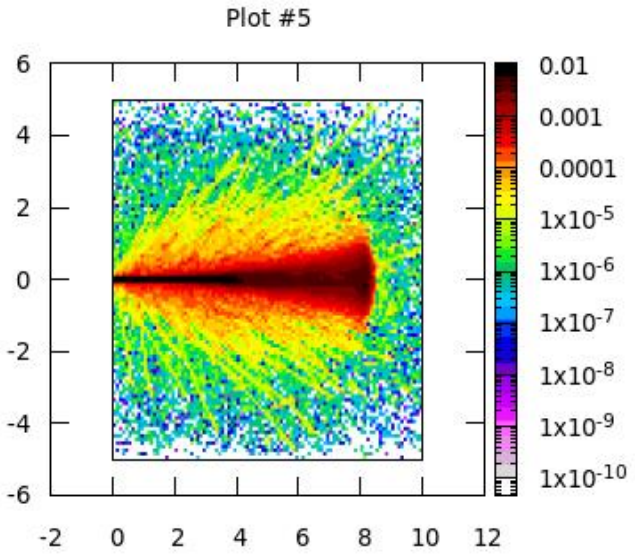
# Plotting results in the Plot tab – 9

- Additional plot customization
  - Gnuplot commands
  - Plot extents
  - Axis location
  - Label offsets
  - And much more….

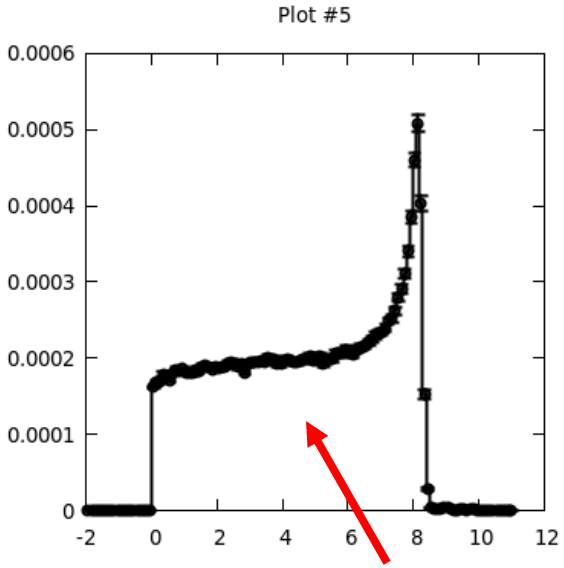


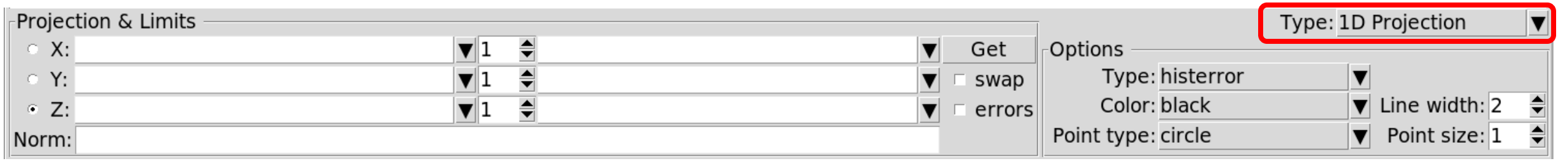
# Plotting results in the Plot tab – 10



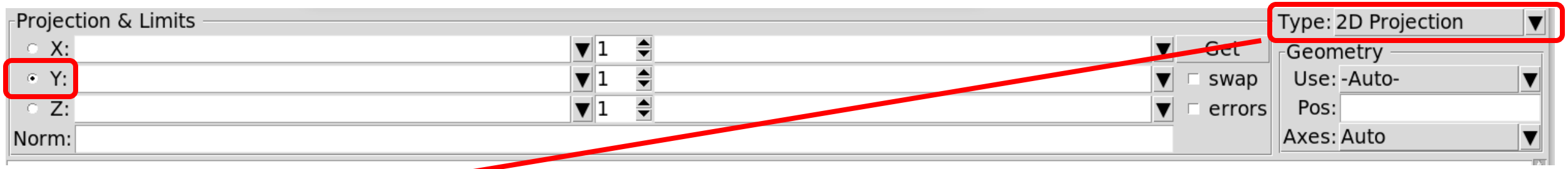



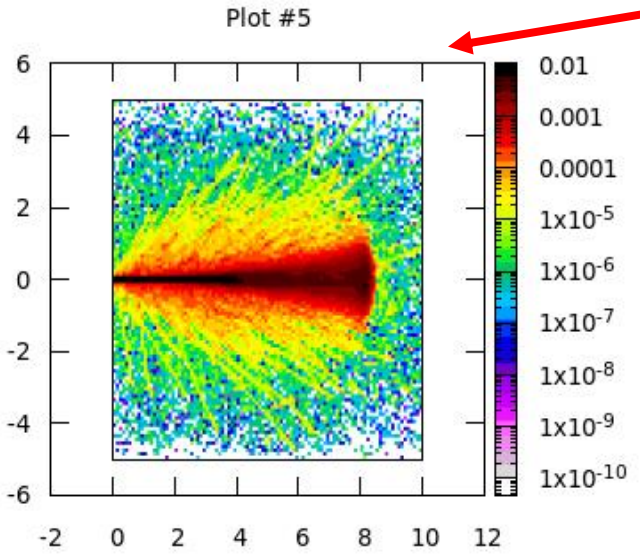

- 2D vs 1D projections



# Plotting results in the Plot tab – 11



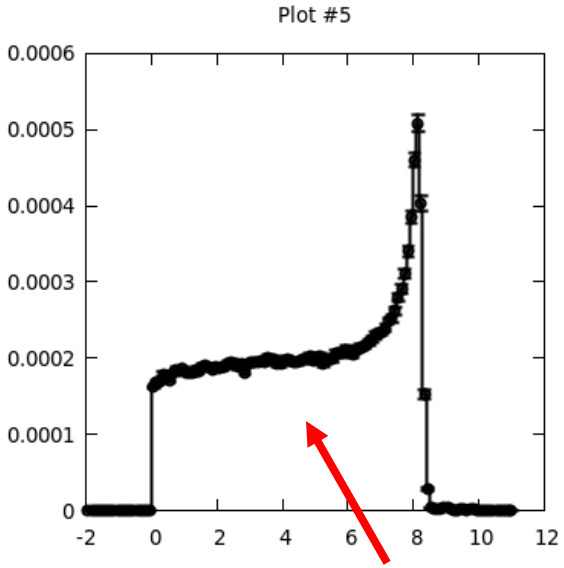



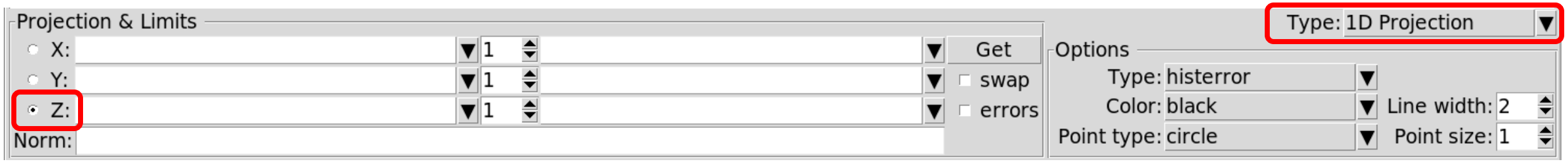

- The result is averaged over the selected coordinate

  i.e. a X-Z plot averaged over the Y coordinate

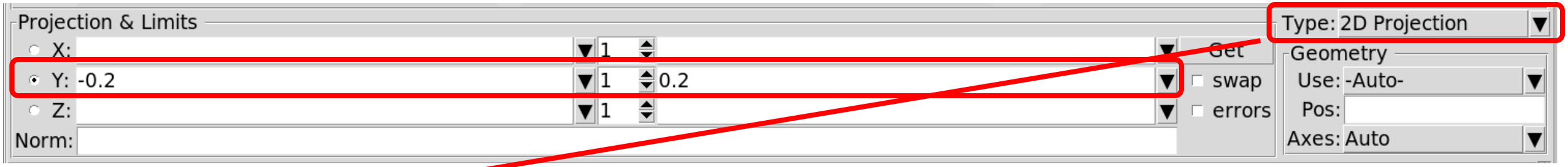
# Plotting results in the Plot tab – 12

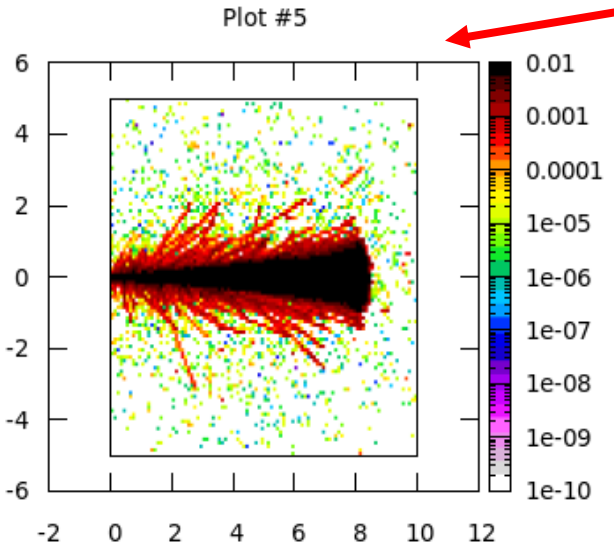
- The result is projected along the selected coordinate and averaged over the non-selected coordinates i.e. a projection along the Z axis



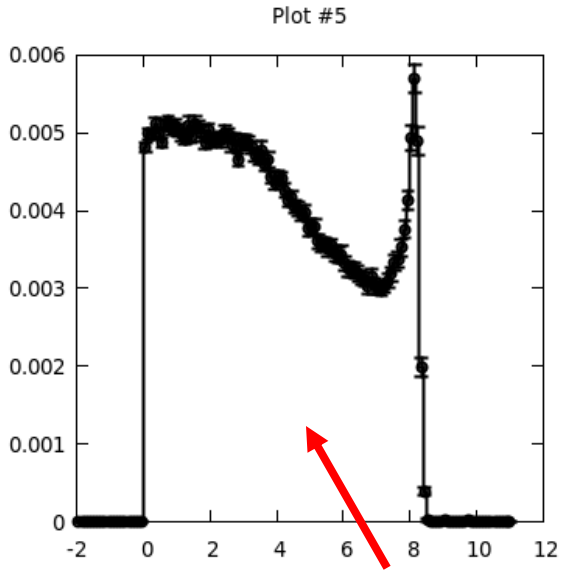



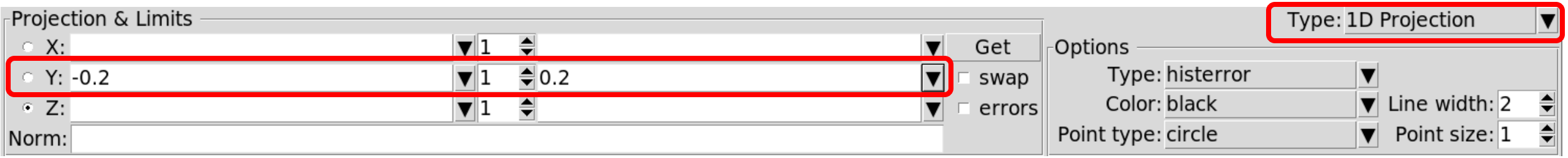
# Plotting results in the Plot tab – 13



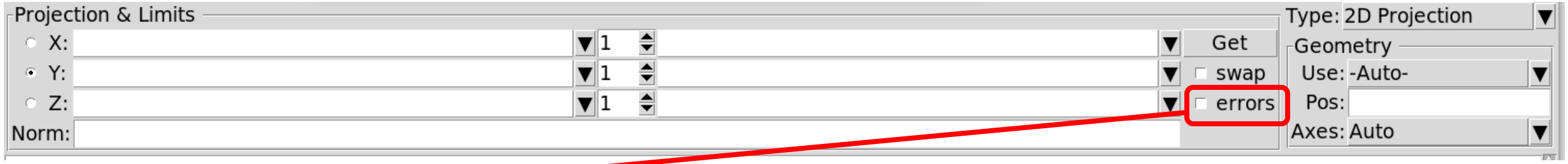



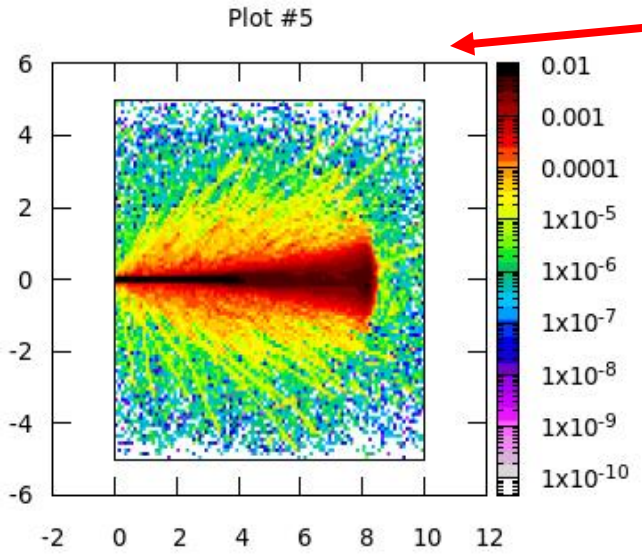

- Plot extension
- The results is averaged only within the specified limits



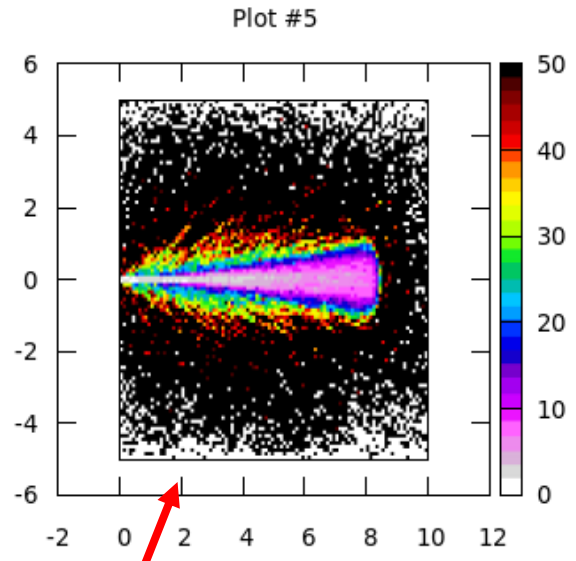
# Plotting results in the Plot tab – 14

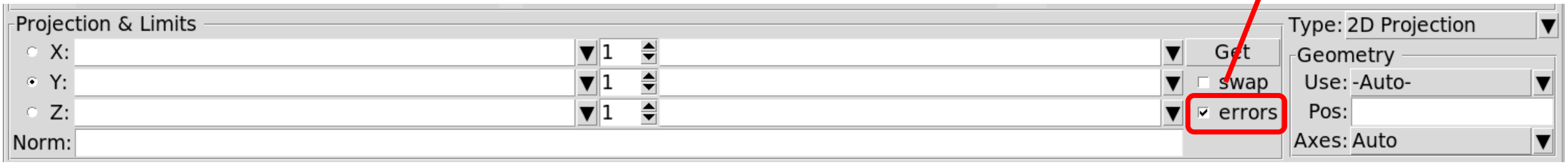



- Uncertainty

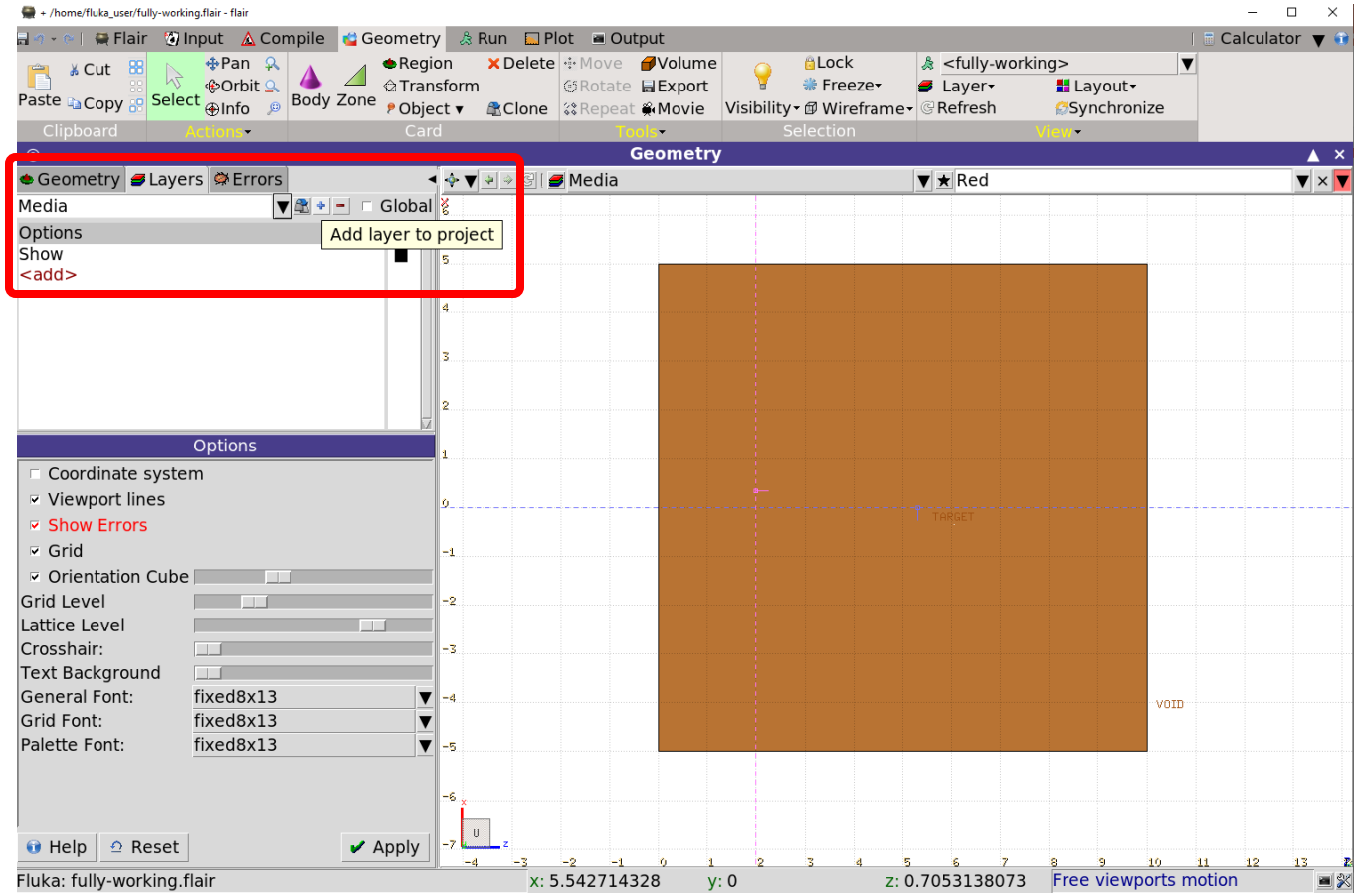
By ticking the “errors” box, it is possible to plot the statistical uncertainty in percent



# 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

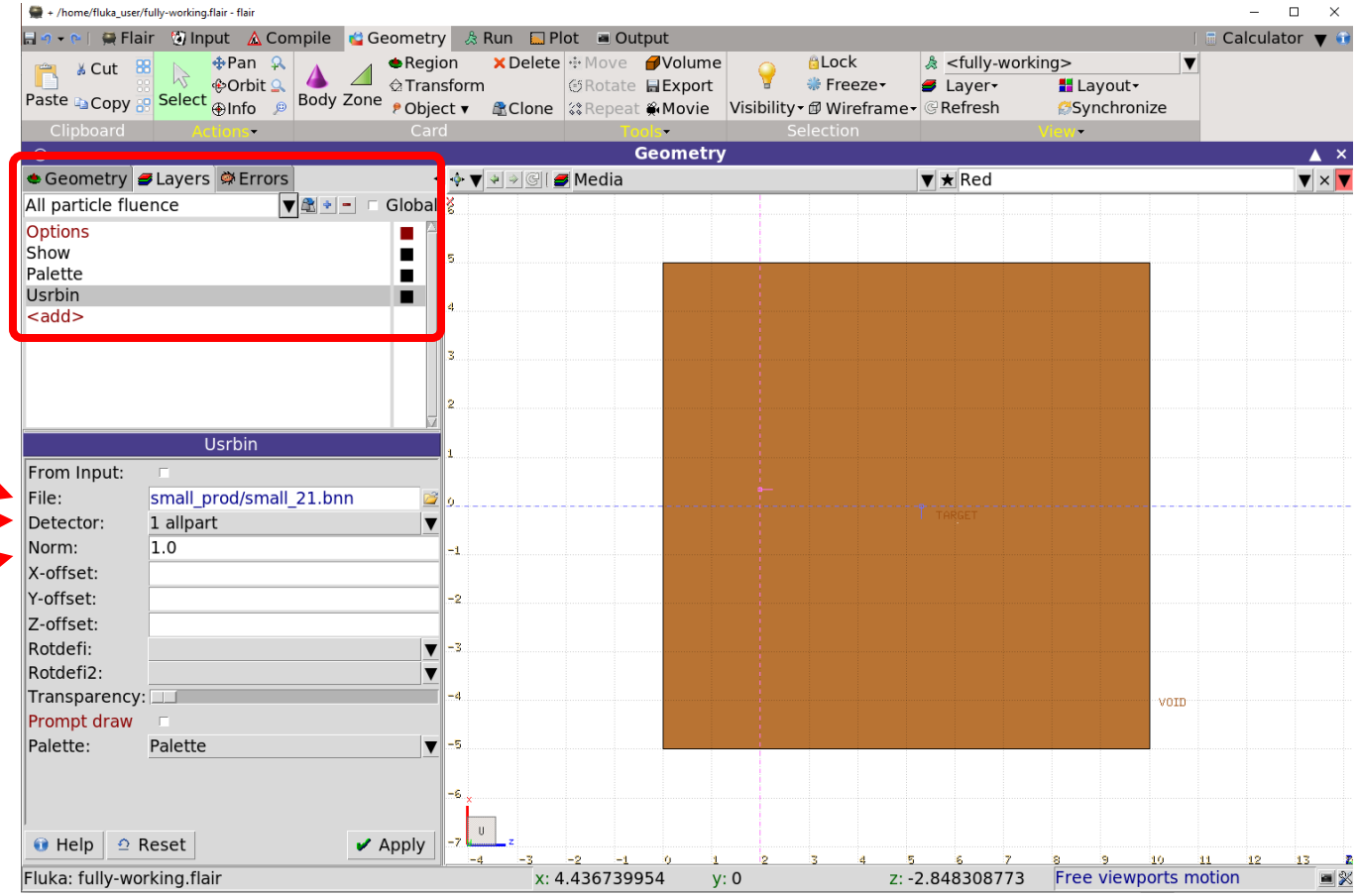
- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one

- <add> “Usrbin”

  (possible to add more than one)

- Select the file with the results
- Select the detector
- Play with normalization,

  palette and other options

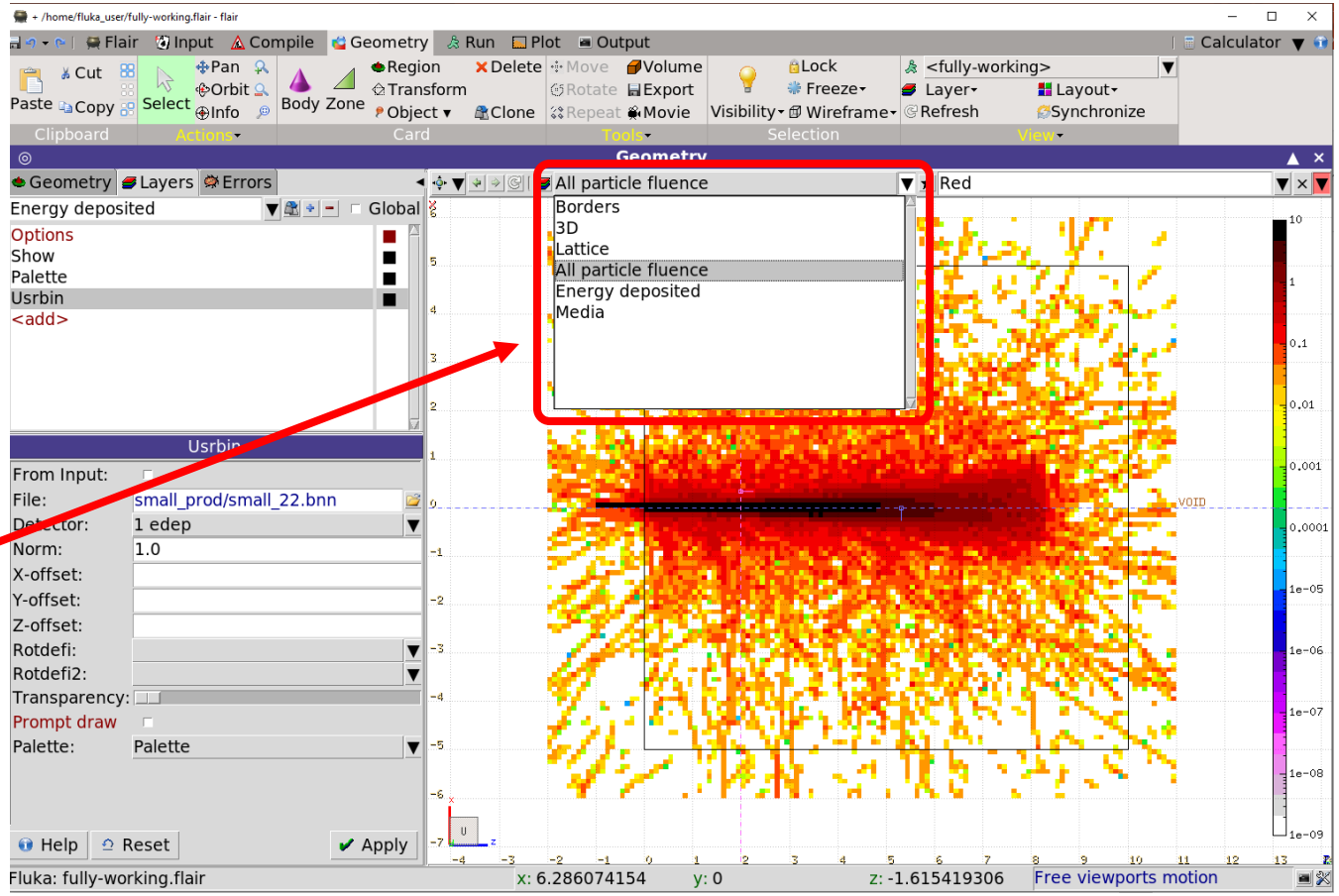


# Plotting results in the Geometry tab – 3

- 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 normalization,

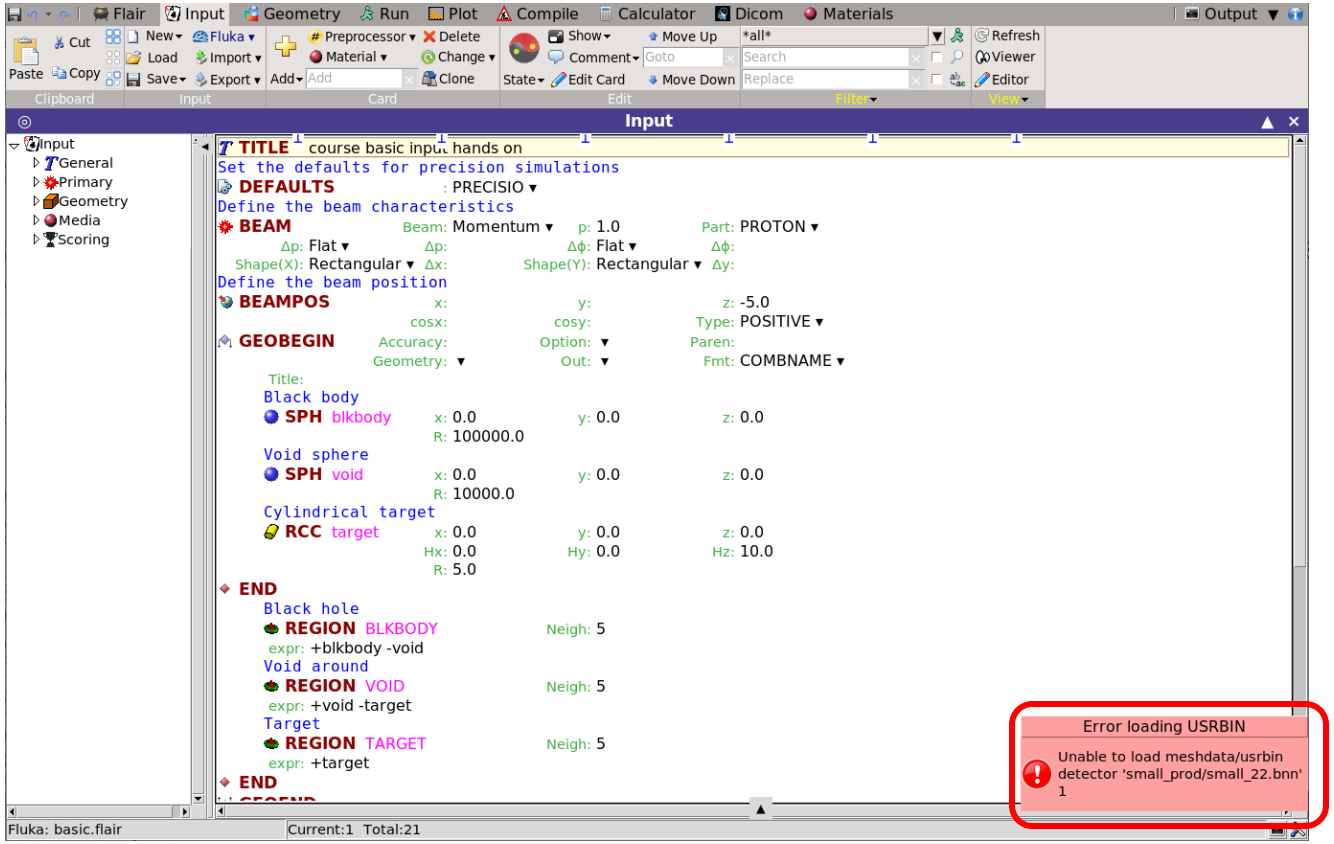
  palette and other options

- Select the layer to visualize



# 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!



# Summary of the work flow

- Create your **input** in the Input tab and Geometry tab (see future lectures)
- 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 (see future lectures)

# Time to do some practice!

- Let's start from the example file

  and run a simulation step by step

FLUKA