Installing and Running

FLUKA Beginner's Course

How to download and install FLUKA

Two ways of downloading the FLUKA software:

- From the FLUKA website http://www.fluka.org
- From NEA databank http://www.nea.fr through the liaison officer from your institute

It is mandatory to be registered as FLUKA user.
Follow the link:

http://www.fluka.org/download.html

After registration , using your user-id and password, you can proceed in downloading the latest official release version.

The currently available distribution files are: (YY at present = 2x)

fluka2011.YY-linuxAA.tar.gz (for g77 compiler, 32 bit mode)

fluka2011.YY-linux-gfor64bitAA.tar.gz (for gfortran compiler version 8.3, 64 bits)

fluka2011.YY-linux-gfor64bit-7.4-AA.tar.gz (for gfortran compiler version 7.4, 64 bits)

fluka2011.YY-mac-gfor64bit-7.3-AA.tar.gz (for Mac, gfortran compiler version 7.3, 64 bits)

fluka2011.YY-mac-gfor64bitAA.tar.gz (for Mac, gfortran compiler version 8.2, 64 bits)

fluka-2011.YY.i686.rpm (rpm, 32 bit, for g77 compiler)

fluka-2011.YY.x86_64.rpm (rpm, 64 bit, gfortran-8.3)

How to download and install FLUKA from tar file

Choose the tar file according to your operating system/compiler version and download it. In the following instructions we assume you are using flukaXXXXAA.tar.gz (for g77 compiler, 32 bit mode)

The following commands, issued from a terminal/console window, will create a directory flukagfor under your home directory and install FLUKA.

```
cd                                  # changes directory to your home
mkdir flukagfor                     # creates a directory called FLUKA
cd flukagfor                        # changes to the FLUKA directory
tar zxvf  path-to-download/fluka2011XXXXAA.tar.gz .
                                    # expands the FLUKA package
# set FLUPRO environment variable
export FLUPRO=$HOME/flukagfor   # sets FLUPRO in bash shell or similar
or  setenv FLUPRO $HOME/flukagfor   # sets FLUPRO in tcsh shell or similar
make                                # compiles a FLUKA executable and
                                      auxiliary programs
```

How to download and install FLUKA – g77/gfortran

The installation for g77 and gfortran versions follow the same procedure. However:

For gfortran, be careful to the compiler version (gfortran –version)

And: tell the system that we are using gfortran , either with

```
    # set FLUFOR environment variable
    export FLUFOR=gfortran  # sets FLUFOR in bash shell or similar
or  setenv FLUFOR gfortran  # sets FLUFOR in tcsh shell or similar
```

Or

Choose a name for the installation directory containing "gfor" (as in this course)

How to download and install FLUKA from rpm

On systems supporting rpms you can install FLUKA via the rpm distribution file (fluka-20XX.YY.i686.rpm or fluka-20XX.YY.x86_64.rpm).

Some Linux distributions offer graphical rpm installers; alternatively, you can install the rpm directly from the command line:

```
# installing FLUKA using the RPM file
rpm -ivh path-to/fluka-20XX.YY.i686.rpm
```

or

```
dnf install path-to/fluka-20XX.YY.i686.rpm
```

Note: FLUKA is installed in the system directory tree (/usr/local) and hence one needs root privileges (or according permissions via sudo) for the installation.

$FLUPRO !!!!

The environmental variable FLUPRO must be set each time you compile or run Fluka

To make environment variable settings persistent on your computer, you can add the following lines in your shell configuration file (already done on the Linux machines used in this course).

bash users:

```
cd
emacs [or any editor] .bashrc
```

add the following:

```
export FLUPRO=${HOME}/flukagfor
export FLUFOR=gfortran                (only if distribution for gfortran is used)
export PATH=${PATH}:$FLUPRO:$FLUPRO/flutil
```

tcsh users:

```
cd
emacs [or any editor] .tcshrc
```

add the following:

```
setenv FLUPRO ${HOME}/flukagfor
setenv FLUFOR gfortran                (only if distribution for gfortran is used)
setenv PATH ${PATH}:$FLUPRO:$FLUPRO/flutil
```

The changes will be activated on the next login or if you type the command

```
source ${HOME}/.bashrc
source ${HOME}/.tcshrc
```

FLUKA release: main directory $FLUPRO

Main Library:

libflukahp.a (object collection)

Physics data files:

sigmapi.bin
elasct.bin
brems_fin.bin
cohff.bin
gxsect.bin
neuxsc-ind_260.bin
nuclear.bin
fluodt.dat
e6r1nds3.fyi
jef2.fyi
jendl3.fyi
xnloan.dat
Fad/*
DDS/*

Basic Scripts: (in $FLUPRO/flutil)

rfluka
lfluka
fff

Random Number seed

random.dat

Important Directories

flukapro/	all FLUKA commons
usermvax/	user routines
flutil/	general utilities

Working directory

- It is strongly recommended to reserve the $FLUPRO directory for FLUKA installation only.
- Simulations shall be run within separate working directories
- The FLUKA code and scripts take care of retrieving all information, provided the environmental variable FLUPRO is set!
- you can check with

  env | grep FLUPRO

Available Documentation

- fluka2011.manual ASCII version of the manual (easy to edit)
- FM.pdf current version of the FLUKA manual
- CERN-2005-10.pdf official reference for FLUKA (manual not up to date)
- or navigate the manual, online version (www.fluka.org)

- or (when using FLAIR) press F1 to get an interactive manual

- or the FAQ available at:
  http://www.fluka.org/fluka.php?id=faq&mm2=3
- or the archive of fluka-discuss:
  http://www.fluka.org/MailingList.html
- Release notes

Input example

- FLUKA is driven by the user almost completely by means of an input file (.inp) which contains directives issued in the form of
  DATA CARDS
- The standard release provides a simple case to test the installation: example.inp
- Different examples are used along this course, which will be varied in different ways for didactic reasons
- We will start with a minimum input file and after each lecture we will enhance our example with more and more functionality
- It is strongly recommended that for every exercise you create a subdirectory *i.e.*, example_running, ex_Geometry1… where all the necessary input and output file will be stored
- For better clarity before starting a new exercise you will get the solution of the previous one, to be picked up at the course website:
  https://indico.cern.ch/event/753612/

Prepare the working space

- We don't want to run inside the $FLUPRO directory, therefore:
- Go to your home directory and create a subdirectory named example_running:

```
cd
mkdir example_running
cd example_running
```

- Get the source example file from the usb pen (copy example_running.inp files to your subdirectory: **example_running**)

UNITS and Coordinates

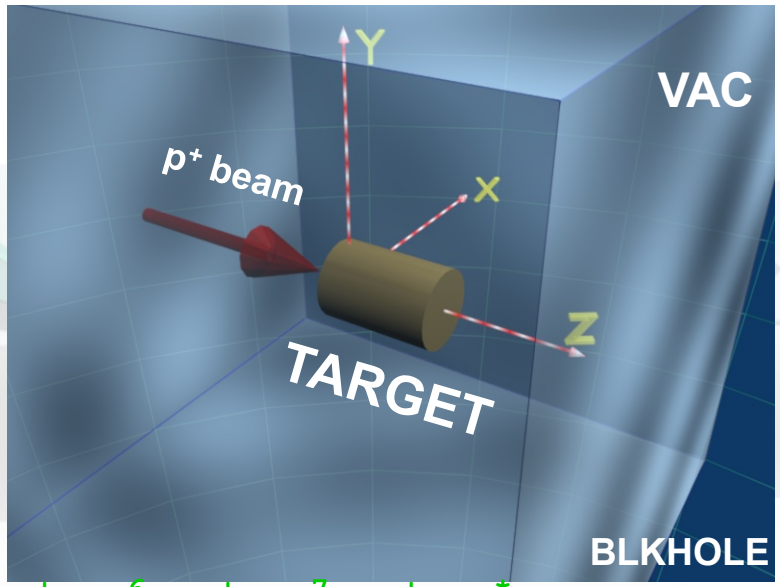
- FLUKA units:
  - Length: cm
  - Mass: g
  - Energy: GeV
  - Time: s

- FLUKA coordinate system:
  - Right-handed Cartesian system
  - By default, the primary beam is directed along the z axis, going in the positive direction (can be changed by user)

A simple example

```
TITLE
FLUKA Course Exercise
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....*
DEFAULTS                                                          NEW-DEFA
BEAM             -3.5      -0.8      -1.7       0.0       0.0       1.0PROTON
BEAMPOS           0.0       0.0      -0.1       0.0       0.0       0.0
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....*
GEOBEGIN                                                          COMBNAME
    0    0                    Cylindrical Target
SPH BLK 0.0  0.0  0.0  10000.
* vacuum box
RPP VOI -1000. 1000. -1000. 1000. -1000. 1000.
* Lead target
RCC TARG 0.0 0.0 0.0 0.0 0.0 10. 5.
END
* Regions
* Black Hole
BLKHOLE 5    +BLK -VOI
* Void around
VAC     5    +VOI -TARG
* Target
TARGET  5    +TARG
END
GEOEND
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....*
ASSIGNMA    BLCKHOLE    BLKHOLE
ASSIGNMA      VACUUM        VAC
ASSIGNMA        LEAD     TARGET
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....*
RANDOMIZ          1.0
START            10.0       0.0
STOP
```

Geometry



Now let's test the installation

After you have created your standard FLUKA we can run the first example:

Script that runs fluka

No. of previous cycle (default is 0)

No. of Last cycle (default is 5)

$FLUPRO/flutil/rfluka -e $FLUPRO/flukahp -N0 -M1 example_running

Specifies the executable name: if it is flukahp in $FLUPRO (default) then it can be omitted

Name of the input file. It must be a file named ****.inp (one must omit the .inp when specifying the file name)

What rfluka does:

It creates a temporary subdirectory: $PWD/fluka_nnnn
($PWD means the current directory)
where nnnn is the system process-id assigned to FLUKA. There all necessary logical links are established and output files are written.

elasct.bin → $FLUPRO/elasct.bin
fluodt.dat → $FLUPRO/fluodt.dat
fort.1 → **../ranexample_running001**
fort.11 → **example_running001.out**
fort.12 → libec_thihecufealw_10t.pemf
fort.15 → **example_running001.err**
fort.16 → "geometry scratch"
fort.2 → **ranexample_running002**
neuxsc.bin → $FLUPRO/neuxsc-ind_260.bin
nuclear.bin → $FLUPRO/nuclear.bin
sigmapi.bin → $FLUPRO/sigmapi.bin
xnloan.dat → $FLUPRO/xnloan.dat

(for non-experts in fortran: fort.xx is the default file name for writing/reading in fortran, xx being a logical unit number. Can be substituted of course with a real name)

What rfluka does -II

- As described in the introduction to MonteCarlo,
  - FLUKA uses pseudo-random numbers to simulate physics processes
  - Many "histories", or "primary particles" are needed to reach a good statistical accuracy
  - Statistical errors can be derived as rms from "batches" of primaries
- rfluka takes care of running several "batches" or cycles,
  - numbering them for convenience and further use ,
  - and giving appropriate names to the output files : i.e. example_running002.out is output from input example_running.inp, 2nd cycle.
- How many cycles ? Defined by the -M and -N parameters: from cycle N+1 to cycle M
- The collection of these cycles is called a "run"
- The pseudo-random sequence is preserved by FLUKA + rfluka:
  - Initial random copied from $FLUPRO or generated (see lecture) as *raninp001*
  - At Nth cycle end (actually more often), random written to *raninp### , ###=N+1*
  - To be used as starting point for the next cycle

At the end of the FLUKA run:

- If everything is OK the temporary directory disappears
- And the relevant results are copied in the start directory:

- Removing links
- Removing temporary files
- Saving output and random number seed

by default you have example_running00n.log, example_running00n.out, example_running00n.err (n=cycle) and ranexample_running00m (seed for cycle m = n+1)

- Saving additional files from scoring requested by the user

User-defined scoring (see lecture)

```
Moving fort.33 to /home/username/work/ex_running/example_running001_fort.33
Moving fort.47 to /home/username/work/ex_running/example_running001_fort.47
Moving fort.48 to /home/username/work/ex_running/example_running001_fort.48
Moving fort.49 to /home/username/work/ex_running/example_running001_fort.49
Moving fort.50 to /home/username/work/ex_running/example_running001_fort.50
```

- End of FLUKA run

Checking FLUKA during the run

Look in the temporary directory:
a) Initialization phase ends when the *.err file is created.
b) Inside *.err file and (at the end of *.out file) the progress in the number of events is given in the line immediately following the one which starts by "NEXT SEEDS":

```
NEXT SEEDS:  C8888D     0      0      0      0      0 33B49B1      0      0      0
     1                9                9          0.0000000E+00      1.0000000E+30
                                                   0
NEXT SEEDS:  C88894     0      0      0      0      0 33B49B1      0      0      0
     2                8                8          5.0010681E-03      1.0000000E+30
                                                   0
NEXT SEEDS:  C8889A     0      0      0      0      0 33B49B1      0      0      0
     3                7                7          3.3340454E-03      1.0000000E+30
                                                   0
                                                  .....
```

EVENTS ALREADY COMPLETED

EVENTS TO BE COMPLETED

AVERAGE CPU TIME CONSUMED PER EVENT

Always open the output file

- The standard inp###.out file contains plenty of information
- If FLUKA crashes, gives hints on the reason
- It tells you how FLUKA interpreted your input cards → spot subtle errors
- It contains physics data used by LUKA
- It provides summary of the cycle: energy deposited, CPU time, particles produced….
- ➔ When setting up a simulation, it is a good practice to ALWAYS run a short test and check the output file
- → If something in the results puzzles you, ALWAYS check in the output file that the settings are what you meant to have.
- We'll show you examples all along the course

Output–Timing of the run- number of primaries

Use it to choose the number of primaries / cycle

Q: how many primaries?
A: as many as needed to reach a good statistical convergence

Q: what is a "reasonable" CPU time for a long cycle ?

A: less than one day, to be on the safe side for crashes

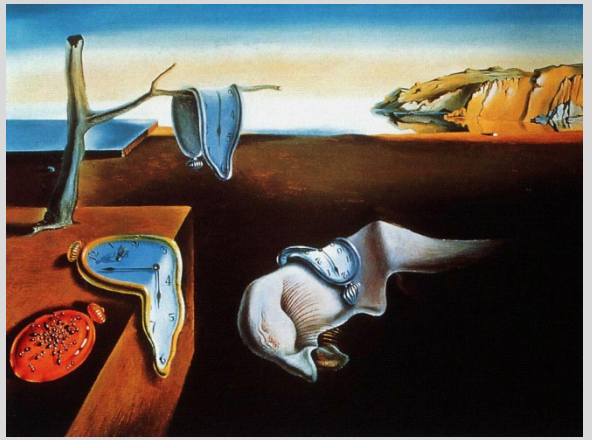
Q: in this example, how many primaries can be run in a 10h cycle?

A: 3600/6.8E-3 ≈ 5E5

Q: how many cycles?
A: minimum 5 to be able to calculate statistics

```
Total number of primaries run:          1000 for a weight of: 1.000000E+03
!!! Please remember that all results are normalized  per unit weight !!!
The main stack maximum occupancy was         81 out of      40000 available


Total number of inelastic interactions (stars):              1722
Total weight of the inelastic interactions (stars):  1.722000E+03


Total number of elastic interactions:                1582
Total weight of the elastic interactions:  1.582000E+03


Total number of low energy neutron interactions:             20821
Total weight of the low energy neutron interactions: 2.082621E+04

Total CPU time used to follow all primary particles:   6.843E+00 seconds of:

Average CPU time used to follow a primary particle:    6.843E-03 seconds of:

Maximum CPU time used to follow a primary particle:    4.699E-02 seconds of:

Residual CPU time left:                                1.000E+30 seconds of:
```

CPU time is not real time!



Complete the run

- add statistics by running more cycles:

- **$FLUPRO/flutil/rfluka –N1 –M5 example_running**

- **While it runs, have a look**

Output: Energy Balance

```
  3.5000E+00  (100.%) GeV available per beam particle divided into
Prompt radiation      Radioactive decays
  2.9309E-01  ( 8.4%)   0.0000E+00 ( 0.0%) GeV hadron and muon dE/dx
  1.1665E-01  ( 3.3%)   0.0000E+00 ( 0.0%) GeV electro-magnetic showers
  8.8952E-03  ( 0.3%)   0.0000E+00 ( 0.0%) GeV nuclear recoils and heavy fragments
  0.0000E+00  ( 0.0%)   0.0000E+00 ( 0.0%) GeV particles below threshold
  0.0000E+00  ( 0.0%)   0.0000E+00 ( 0.0%) GeV residual excitation energy
  1.1821E-03  ( 0.0%)   0.0000E+00 ( 0.0%) GeV low energy neutrons
  2.9282E+00  (83.7%)   0.0000E+00 ( 0.0%) GeV particles escaping the system
  1.6105E-02  ( 0.5%)   0.0000E+00 ( 0.0%) GeV particles discarded
  0.0000E+00  ( 0.0%)   0.0000E+00 ( 0.0%) GeV particles out of time limit
  1.3589E-01  ( 3.9%)                       GeV missing
```

Escaping the system: out of the geometry and going to other *blackholes (see lecture on geo).* If you find 100%..maybe something is wrong ..

Discarded particles (i.e. neutrinos).

Missing Energy: Calculated by difference:

- pure EM problems it should be 0;
- in hadronic problems it is the energy spent in endothermic nuclear reactions (≈ 8 MeV/n), or gained in exothermic (i.e. mostly neutron capture): it is -total Q.

Tips & Tricks

How to make a "clean" stop of FLUKA run

- Here "clean" means closing all files, writing scoring output and removing the temporary directory and files.
- In the temporary run directory:
  touch fluka.stop To stop the present cycle

- or kill -SIGTERM <process_id>
  the same id as in the fluka_xxxx
  or touch rfluka.stop To stop all remaining cycles
- The clean stop will occur at the next CPU-time check, *i.e.*, at the same time when printing the random number calls : see START card instructions (5th parameter) for the frequency of these checks!!

- If the check is never performed it means that the program has entered an infinite loop (probably a fault in user code)

MAC users

- A Mac version is available
- Users shall have gfortran installed.
- For the installation of the FLAIR graphical interface, see slides in the backup

Virtual machine installation

- A VM distribution based on Docker is available: https://flukadocker.github.io/F4D/
- The instructions provided allow you to:
  - Installing Docker
  - Generate your personal Docker image with FLUKA
  - Create your first FLUKA container
- There is also a list of known issues and instructions to update the FLUKA Docker image.

Thanks for your attention!

BACKUP

Flair Installation for Mac OSX

G. Battistoni, INFN Milano

System requirements (March 2018)

Example Instructions for Mac OS X High Sierra (10.13.3)

fink as manager for software installation

Flair installation

Present version: flair-2.3-0 and flair-geoviewer-2.3-0

Installation has to start from the flair*tgz packages to be locally compiled

Important: The crucial point is that it is necessary to have python installed by fink

Notice that in this case the relevant software goes into
/sw/lib
/sw/bin
etc.

Check:
1) there must exist /sw/bin/python (**ls /sw/bin/python***)
2) check with fink giving the command: **fink list python**
it should appear:
i python27 1:2.7.14-1 Interpreted, object-oriented language
i python27-shlibs 1:2.7.14-1 Interpreted, object-oriented language
(notice the initial i, which means “installed”)

Flair installation

If /sw/lib/python does not exist, and/or fink list python does not return the signal that python2.7 is installed give the command:
fink install pyhton 2.7
(one has to have administrator privileges)

After that perform the same checks described before

Also numpy has to be installed with fink.

Check with the command: **fink list numpy**
i numpy-py27 1.14.0-1 N-dimensional array package for Python2)

If not installed: **fink install numpy-py27**

Flair installation

The $PATH (or $path) env. variables must have /sw/bin with precedence with respect to /usr/bin

If in /sw/bin there is python2.7 but not just pyhton, then, as superuser, go to /sw/bin
and issue the command:
ln –sf pyhton2.7 python

For people accessing DICOM files with flair give also the command:
easy_install pydicom

Flair installation

After that download and expand flair and flair-geoviewer tgz files

Then follow all instructions in the README file contained in the flair-2.3 directory concerning tarball installation

In the same README there are also the instructions to compile and install flair-geoviewer