Accelerator-Driven System Design FLUKA Exercise

Adonai Herrera-Martínez and Yacine Kadi
AB Dept. ATB/EET
European Organization for Nuclear Research (CERN)
CH-1211 Geneva 23, Switzerland

Introduction

The goal of this exercise is to give a basic on how to run to Monte Carlo code FLUktuierende KAskade (FLUKA) [1]. This is a general-purpose tool for simulating the interaction of radiation with matter, covering an extended range of applications, from high-energy physics (e.g. study of the ATLAS detector at CERN) to medical and radiation physics (e.g. calculation of doses to astronauts by NASA). Namely, it has been thoroughly used in the design of ADS for waste transmutation and energy production. The development of the Energy Amplifier by the Emerging Energy Technologies group at CERN was performed simulating the nuclear cascade, produced by high-energy protons (occurring mainly in the spallation target of the device), and coupling its results with EAMC [2], to accurately transport the low-energy neutrons and calculate the burn-up evolution, using the latest point-wise cross-section libraries.

The present exercise is a simplified version of a model used in the spallation target design performed within Task #2 of the EURopean Isotope Separation On-Line Radioactive Ion Beam Facility Design Study (EURISOL DS) [3]. For the sake of comparison, the final document of this study may be downloaded from Reference [4].

The input defines a cylindrical target, where several incident particles, their energies and target materials can be varied, in order to study the effect of these parameters in the general neutronics of the system.

The output files will show the particle distributions, energy deposition, fission density and particle energy spectrum in the target, allowing the students to extract useful conclusions concerning the design of the system.

Model Layout

Initially, the beam particles considered are protons with a kinetic energy of 1GeV, following a Gaussian distribution with a standard deviation of 5.0 mm in both x and y directions. The target is 80 cm long and 20 cm radius (Figure 1), composed of Hg, with a density of 13.55 g/cm^3. Figure 1 illustrates an artistic view of the target, with a 20 cm long, 2 cm radius vacuum along the proton line in order to reduce the possible particle backscattering through the front cap.

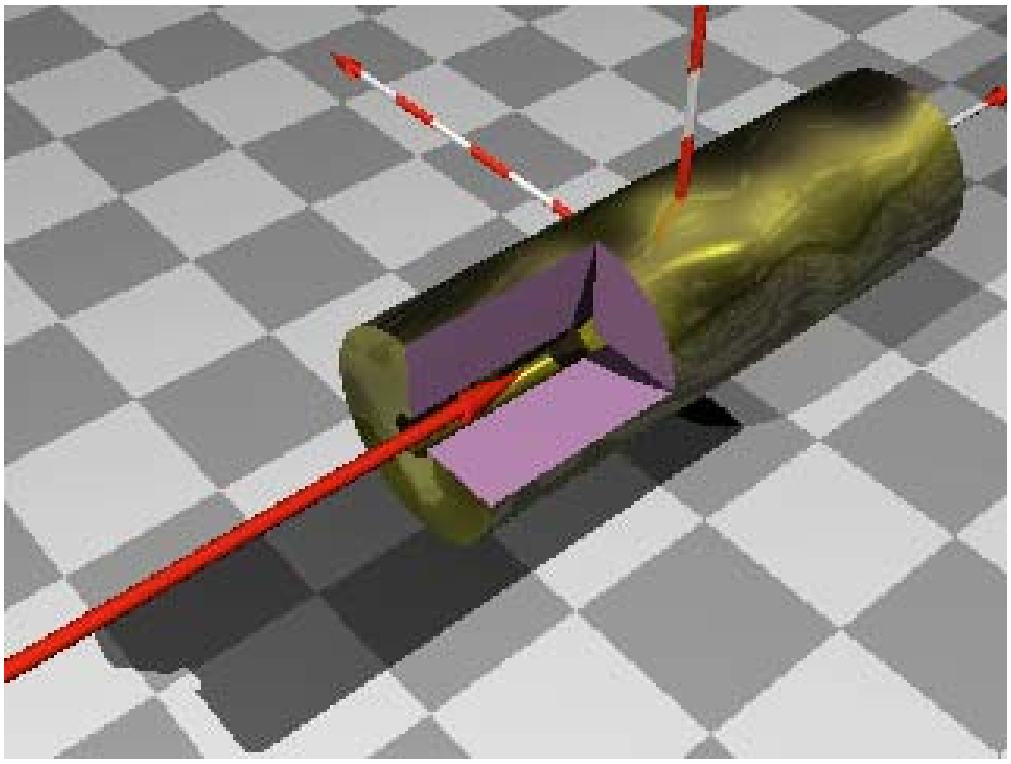


Figure 1. 3-D view of the target used in the exercise.

Generalities about FLUKA input

FLUKA reads user input from an ASCII "standard input" file with extension .inp. The input consists of a variable number of "commands" (called also "options"), each consisting of one or more "lines" (called also "cards" for historical reasons). The typical structure of a FLUKA input file is the following:

- Titles and comments for documentation purposes (optional, but recommended)
- Definition of the particle source (mandatory)

- Description of the problem geometry (solid bodies and surfaces, combined to partition space into regions) (mandatory)
- Definition of the materials (mandatory unless pre-defined materials are used)
- Material assignments (correspondence material-region, mandatory)
- Definition of the requested “detectors”. Each of these is a phase space domain (region of space, particle direction and energy) where the user wants to calculate the expectation value of a physical quantity such as dose, fluence, etc. Various kinds of detectors are available, corresponding to different quantities and to different algorithms used to estimate them (“estimators”). Detectors are optional, but one at least is expected, at least in the production phase
- Definition of biasing schemes (variance reduction technique, optional)
- Definition of problem settings such as energy cut-offs, step size, physical effects not simulated by default, particles not to be transported, etc. (optional)
- Initialisation of the random number sequence (mandatory if an estimation of the statistical error is desired)
- Starting signal and number of requested histories (mandatory)

In addition, special commands are available in FLUKA for more advanced problems involving magnetic fields, time-dependent calculations, writing of history files (so-called "collision tapes"), transport of optical photons, event-by-event scoring, calling user-written routines, etc. These options are expected to be requested only by users having some previous experience with the more common commands: therefore they will be mostly ignored in this beginner's guide.

The general structure of the FLUKA command lines or cards (except for the geometry commands) contains:

- one keyword,
- six floating point values (called WHATs),
- one character string (called SDUM)

Not necessarily all WHATs and SDUMs are used. In some cases, a command line can be followed by a line of text (for instance a filename path or a title). Any line having an asterisk (*) in the first position is treated as a comment. All lines (commands, text strings and comments) are echoed on the standard output (the file with extension .out). In case of problems, it is a good idea to check how every line has been printed in the standard output.

More information at: *http://www.fluka.org/manual/Toc.html*

FLUKA Input Cards

This document briefly defines the input cards related with the present exercise, only explaining their most relevant options. Therefore, in order to fully understand the effect of each command in the FLUKA input, the user should read the definition of each one of them from the online manual:

http://www.fluka.org/manual/sect/s010/text.html

The command explanation follows the order of the exercise example, starting by the title and beam definition, followed by the geometry, material assignment and detector cards. The so-called WHAT's (from 1 to 6) are the command options, with a trailing text field called SDUM.

Title and Beam Definition

```
TITLE
EURISOL Multi-MW TARGET
BEAM       -1.0000    0.0         0.0       -2.35000  -2.35000   1.000     PROTON
BEAMPOS      0.0      0.0       -50.0         0.0       0.0
```

WHAT-1 is the BEAM card defines the kinetic energy of the primary particles, whereas WHAT-4&5 define the FWHM (2.35*sigma of the PROTON beam). BEAMPOS defines the initial position and direction cosines of the beam particles.

GeometryDefinition (GEOBEGIN – GEOEND)

```
GEOBEGIN                                                               COMBINAT
          MERCURY TARGET AND FISSILE ENVELOPE FOR HEAVY ION PRODUTION
**AAA*IIII---------+---------+---------+---------+---------+---------+
* Black hole
  SPH    1       0.0       0.0       0.0      340.
  SPH    2       0.0       0.0       0.0      330.
* Target Cylinders
  ZCC    3       0.0       0.0        2.
  ZCC    4       0.0       0.0     20.00
*
* Planes for delimiting the Regions
  XYP    5        0.0
  XYP    6       20.0
  XYP    7       80.0
  END
* REGIONS
* Black Hole & General Vacuum
  001          +1      -2
  002          +2      -4
* Mercury Target
  003     OR   +4      -3      +6      -5
          OR   +4      +7      -6
* Beam Vacuum
  004          +3      +6      -5
* Front-End Vacuum
  005          +4      +5      +2
* Back-End Vac
  006          +4      -7      +2
  END
GEOEND
```

FLUKA uses a Boolean approach to define geometries (similar to MCNP), first declaring the primitive bodies SPH for sphere, XYP for XY infinite planes, ZCC for infinite cylinders in the Z direction etc. The first (consecutive) numbers in the line definitions are the name of the body (in our case, ranging from 1 to 7). The trailing numbers in the body definition state some geometrical parameters for each body (such as its centre, radius etc.).

The Boolean operations (addition and subtraction) between bodies are defined in the second part of the GEOBEGIN. The initial number (i.e. 001, 002…. 006) defines the name/number of each spatial region. The positive or negative numbers show the inclusion or exclusion of a body, for each region. The OR's indicate the addition of several sub-regions to form one region. For example, region 003 is formed by ZCC 4 minus ZCC 3 between XYP 6 and 5, plus ZCC 3 between XYP 7 and 6.

Material Definition and Assignment

```
*23456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
*CARD     WHAT-1    WHAT-2    WHAT-3    WHAT-4    WHAT-5    WHAT-6    SDUM
MATERIAL     82.000   207.138    11.340     3.000      .000      .000 LEAD
MATERIAL     83.000  208.9804     9.780     4.000      .000      .000 BISMUTH
…..
…..
MATERIAL     80.000   200.592    13.550    11.000      .000      .000 MERCURY
* COMPOUND MATERIALS
*
MATERIAL       .000      .000    10.500    14.000      .000      .000 PbBi
COMPOUND      0.550    4.0000    0.4500   3.00000     .0000   0.00000 PbBi
*
*
* Assign materials cards
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
ASSIGNMAT      1.000     1.000     1.000      .000      .000      .000
ASSIGNMAT      2.000     2.000     2.000      .000      .000      .000
ASSIGNMAT     11.000     3.000     3.000      .000      .000      .000
ASSIGNMAT      2.000     4.000     6.000     1.000      .000      .000
*
```

The element materials are defined using the MATERIAL card, where WHAT-1 defines the Z number of the element, WHAT-2 defines its mass number, WHAT-3 defines the material's density and WHAT-4 assigns it a number. Finally, the SDUM option declares the name of the material. Notice the upper line, which is commented using an * as first character, defining the alignment of each one of the option cards (WHAT's).

The COMPOUND cards (always combined with a MATERIAL card) define a material composed by several elements. In the example above, a mixture of bismuth and natural lead is defined as PbBi, presenting and isotopic composition of 0.55 Bi and 0.45 Pb. Its density is declared in the preceding MATERIAL card as 10.5 g/cm3.

The assignment of materials to each region is performed using the ASSIGMAT command, where WHAT-1 shows the number of the material and WHAT-2&3 define the range of regions on which that material is applied.

Detector Cards (RESNUCLE USRBIN, USRBDX, USRTRACK)

```
*23456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
* RESIDUAL NUCLEI
*
RESNUCLE     3.0     -50.0                            3.0   100279.63 TarResNuc
*
* Userbins (two dimensional plots)
USRBIN     11.00       8.0    -41.00      100.0              110.0    Neutrons
USRBIN                        -50.00      200.0              320.0      &
*
* USRTRACK: Fluence Estimators; Neutron flux
USRTRACK   -1.0        8.0     -71.0        3.0    100279.63  120.0   Tar-N-Flux
USRTRACK  1.0000       1.E-9                                            &
*
* USRBDX: User Boundary Crossings; Radial Neutron flux
USRBDX     99.0        8.0    -21.000       3.0       7.0   10053.096 Rad-N-Flux
USRBDX    1.0000       1.E-9     120.0                          6.0     &
```

The ultimate objective of this type of the simulations is to understand the effect of radiation interaction with matter. These results are mainly obtained by defining some “detectors”, which will give the user an idea on the real interaction. There are several types of detectors, although only four are used in this exercise.

In order to compute the residual nuclei, the RESNUCLE card is used. Note that only static results are obtained, thus no pile-up of elements or on-line decay is accounted for. WHAT-1 defines the type of products to be scored, WHAT-2 defines the logical output file where the results will be dumped (explained in the post-processing), WHAT-5 defines the region for which the residual nuclei are followed, WHAT-6 declares the volume (in cm^3) of that region and SDUM states the name of the detector.

The USRBIN card produces the 3-D mapping of a certain parameter. In the example above, the R-PHI-Z (WHAT-1 defines the type of binning) neutron flux (WHAT-2 states the particle to be studied, given by the FLUKA number of that particle) is dumped on a file with the extension 41 (WHAT-3, logical output file). The following cards define the R (or X), PHI (or Y) and Z geometrical dimensions of the detector.

The USRTRACK card scores the energy spectrum of a certain particle, inside a region. Thus, WHAT-2 defines the particle type, WHAT-4 defines the region where the particles shall be tracked and WHAT-4 declares the volume of the aforementioned region. WHAT-5 and the WHAT-1&2 in the second card define respectively the number of energy bins, the lower and the higher thresholds of the detector.

Finally, the USRBDX card also scores the energy distribution of a certain particle type, but only those crossing the boundary surface between to regions (WHAT-4&5).

Running FLUKA

At this point, the only card necessary to finalise the FLUKA input is the declaration of the number of particles to run, which will have an impact on the running time and the statistical error of the Monte Carlo simulation. WHAT-1 in the START card declares the number of primary particles which be simulated (1,000 in the following example).

```
* The first number is the number of events to be processed
START     1.000E+03 1.0000E+9                                1.000E+07
* End the run
STOP
```

FLUKA Exercise

The previously explained FLUKA input should already be copied in each computer. To facilitate the exercise to the students who are unfamiliar with Unix environments, the commands to type in the command line (i.e. **>**) are also included ***using this font***.

First, get into the working directory:

>cd fluka_exercise

And open the FLUKA input with a text editor (e.g. kwrite):

>kwrite fluka_example.inp &

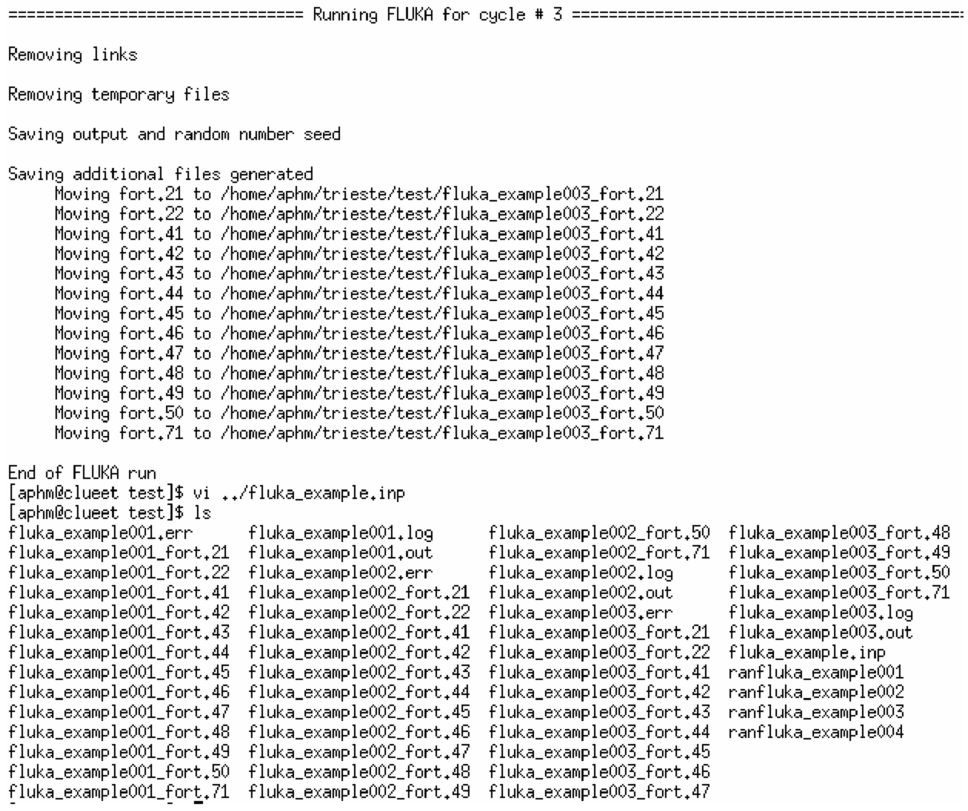
Using this editor, one can modify the input and save the changes. Once the changes are done, FLUKA is run by typing:

>rfluka –M3 –N0 fluka_example &

where fluka_example is the name of the fluka input (without the .inp extesion!)

After a while, and if everything run properly, we should obtain a whole set of results, to list these results (list command):

>ls

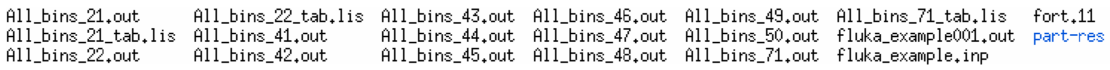


```
=============================== Running FLUKA for cycle # 3 ==========================================

Removing links

Removing temporary files

Saving output and random number seed

Saving additional files generated
     Moving fort.21 to /home/aphm/trieste/test/fluka_example003_fort.21
     Moving fort.22 to /home/aphm/trieste/test/fluka_example003_fort.22
     Moving fort.41 to /home/aphm/trieste/test/fluka_example003_fort.41
     Moving fort.42 to /home/aphm/trieste/test/fluka_example003_fort.42
     Moving fort.43 to /home/aphm/trieste/test/fluka_example003_fort.43
     Moving fort.44 to /home/aphm/trieste/test/fluka_example003_fort.44
     Moving fort.45 to /home/aphm/trieste/test/fluka_example003_fort.45
     Moving fort.46 to /home/aphm/trieste/test/fluka_example003_fort.46
     Moving fort.47 to /home/aphm/trieste/test/fluka_example003_fort.47
     Moving fort.48 to /home/aphm/trieste/test/fluka_example003_fort.48
     Moving fort.49 to /home/aphm/trieste/test/fluka_example003_fort.49
     Moving fort.50 to /home/aphm/trieste/test/fluka_example003_fort.50
     Moving fort.71 to /home/aphm/trieste/test/fluka_example003_fort.71

End of FLUKA run
[aphm@clueet test]$ vi ../fluka_example.inp
[aphm@clueet test]$ ls
fluka_example001.err      fluka_example001.log      fluka_example002_fort.50  fluka_example003_fort.48
fluka_example001_fort.21  fluka_example001.out      fluka_example002_fort.71  fluka_example003_fort.49
fluka_example001_fort.22  fluka_example002.err      fluka_example002.log      fluka_example003_fort.50
fluka_example001_fort.41  fluka_example002_fort.21  fluka_example002.out      fluka_example003_fort.71
fluka_example001_fort.42  fluka_example002_fort.22  fluka_example003.err      fluka_example003.log
fluka_example001_fort.43  fluka_example002_fort.41  fluka_example003_fort.21  fluka_example003.out
fluka_example001_fort.44  fluka_example002_fort.42  fluka_example003_fort.22  fluka_example.inp
fluka_example001_fort.45  fluka_example002_fort.43  fluka_example003_fort.41  ranfluka_example001
fluka_example001_fort.46  fluka_example002_fort.44  fluka_example003_fort.42  ranfluka_example002
fluka_example001_fort.47  fluka_example002_fort.45  fluka_example003_fort.43  ranfluka_example003
fluka_example001_fort.48  fluka_example002_fort.46  fluka_example003_fort.44  ranfluka_example004
fluka_example001_fort.49  fluka_example002_fort.47  fluka_example003_fort.45
fluka_example001_fort.50  fluka_example002_fort.48  fluka_example003_fort.46
fluka_example001_fort.71  fluka_example002_fort.49  fluka_example003_fort.47
```

The files fluka_example00?_fort.(21, 22, 41 etc.) are the logical output number previously defined in the detector cards. The next step is to put together the results from different runs so as to obtain the statistical errors (since each run is performed with a completely independent random seed). To do this, we will run a shell script, which will make our life easier.

>put_together_res.scrpt

What this script does is to sequentially run a set of programs, not only to sum up the results from the run, but also to convert the binary results into a more readable format. By the end of this process, our working directory should contain the following result files (use the ls command again):



```
All_bins_21.out      All_bins_22_tab.lis  All_bins_43.out  All_bins_46.out  All_bins_49.out  All_bins_71_tab.lis   fort.11
All_bins_21_tab.lis  All_bins_41.out      All_bins_44.out  All_bins_47.out  All_bins_50.out  fluka_example001.out  part-res
All_bins_22.out      All_bins_42.out      All_bins_45.out  All_bins_48.out  All_bins_71.out  fluka_example.inp
```

First, we will plot the USRBIN results. These will show colour maps of the 2D distribution of a certain parameter (e.g. neutron flux). Again, we shall use a shell script to make our life easier in obtaining the results:

>bin_prod.scrpt

This will launch a whole set of paw windows, creating .eps files, which can be later visualised typing:

>for f in *.eps; do display –rotate 90 $f; done

Figure 2 illustrates the neutron flux distribution simulated in the example. There should be 9 colour plots equivalent to this one, each one representing the particle distribution for each USRBIN card.

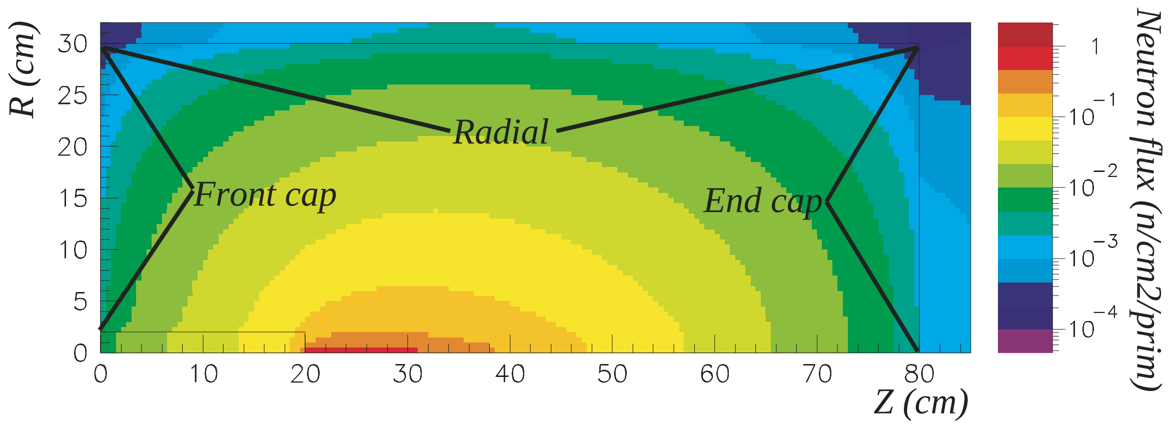


Figure 2. Neutron flux (neutrons/cm^2/primary) distribution for a 1 GeV proton beam.

The next step is to plot the neutron energy spectrum in the target from files 71, 21 and 22. For example, All_bins_71_tab.lis holds the spectrum information for the USRTRACK 71 (neutron spectrum inside the target. To plot it, we will run a script using gnuplot, which is a Linux standard application for producing graphs.

>plot_spectra.scrpt All_bins_71_tab.lis

We are now ready to visualise the neutron energy spectrum with:

>kghostview plot*.ps

The neutron energy spectrum in the target should follow a distribution typical of a fast system, such the one in Figure 3, where the peak of the distribution lays at ~300 keV due to the multiple scattering process of the neutrons in Hg.

The following step is to plot the residual nuclei in the target. For this, we have to transform the binary file All_bins_50.out into ASCII format and select the values to plot (i.e. nuclei as a function of atomic or mass number). By using plot_resnuclei.py (a python script):

>plot_resnuclei.py All_bins_50.out

we will produce 3 files: ***All_bins_50_sum.lis***, ***Z_residual_50.txt*** and ***A_residual_50.txt***. The first one contains all the information, in text (ASCII) format, of the residual nuclei per cm^3 and primary particle, including the isotopic distribution. The students can visualise through it using ***kwrite***.

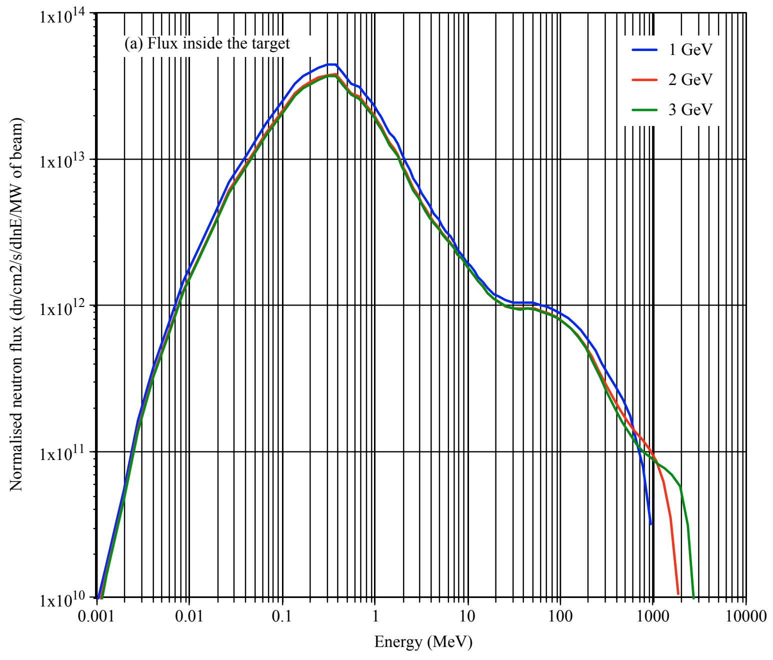


Figure 3. Neutron energy spectrum (neutrons/cm^2/s/dlnE/MW of beam) for different proton energies, inside the Hg target.

After browsing through the isotopic composition of the residual nuclei, the students may plot the distribution as a function of atomic number (A) and mass number (Z) by typing these commands:

```
>plot_residual.scrpt Z_residual_50.txt
>kghostview plot_Z_residual_50.ps
>plot_residual.scrpt A_residual_50.txt
> kghostview plot_A_residual_50.ps
```

These plots should produce an isotopic distribution similar to Figure 4 (a) and (b):

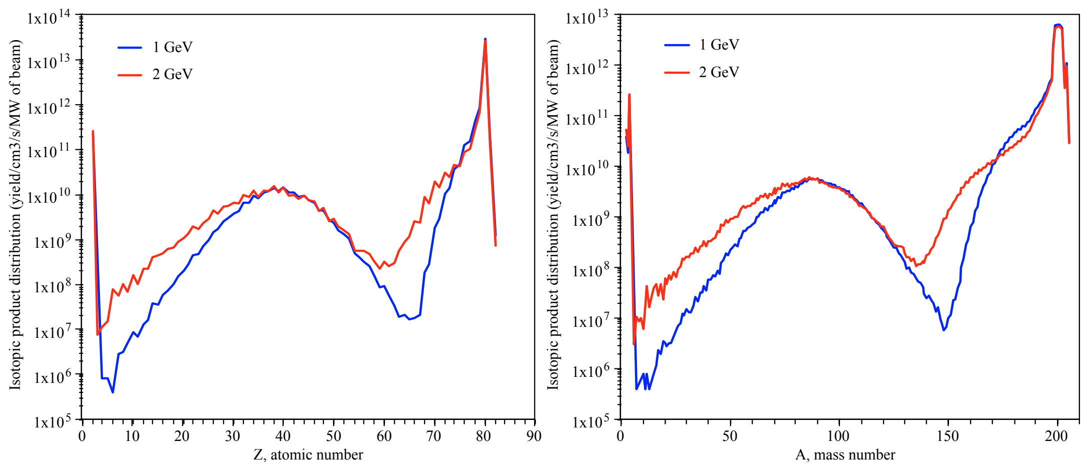


Figure 4. Residual nuclei distribution in the Hg target (nuclei/cm3/mW of beam) for 1 and 2 GeV protons.

Finally, FLUKA produces a text output of the run called ***fluka_example001.out***. This file contains useful information on the loading of the input file, the statistics of the run and about the general physics undergone. At the end of this file (open with ***kwrite fluka_example001.out***) we can extract, for example, the neutron yield:

```
Number of secondaries generated in inelastic interactions per beam particle:
 Prompt radiation      Radioactive decays
   5.1901E+01 (100.%)   0.0000E+00 (100.%)
   1.5600E-01 ( 0.3%)   0.0000E+00 ( 0.0%) 4-HELIUM
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) 3-HELIUM
   1.6000E-02 ( 0.0%)   0.0000E+00 ( 0.0%) TRITON
   3.9000E-02 ( 0.1%)   0.0000E+00 ( 0.0%) DEUTERON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) HEAVYION
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) OPTIPHOT
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) RAY
   3.5500E+00 ( 6.8%)   0.0000E+00 ( 0.0%) PROTON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) APROTON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) ELECTRON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) POSITRON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) NEUTRIE
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) ANEUTRIE
   1.7448E+01 (33.6%)   0.0000E+00 ( 0.0%) PHOTON
-> 3.0368E+01 (58.5%)   0.0000E+00 ( 0.0%) NEUTRON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) ANEUTRON
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) MUON+
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) MUON-
   0.0000E+00 ( 0.0%)   0.0000E+00 ( 0.0%) KAONLONG
   1.0300E-01 ( 0.2%)   0.0000E+00 ( 0.0%) PION+
   7.1000E-02 ( 0.1%)   0.0000E+00 ( 0.0%) PION-
```

In our case, 1 GeV protons interacting with the previously defined Hg target produce approximately 30 neutrons per primary proton. The neutron yield strongly depends on the type of primary particles, the energy of these particles and the target material. Therefore, the students should now modify the following parameters:

- energy of the primary particle
- type of target material
- type of primary particle

This exercise may be performed by using the shell commands ***mkdir*** *dirname* (to create a directory dirname), ***cp*** *filename dirname/* (to copy a file [filename] into a directory [dirname]), ***cd*** *dirname* (to change directory [into dirname]). Once the directories are set and fluka_example.inp copied into each one of them, the students should edit the example file and change the aforementioned parameters, one by one, filling up tables 1 and 2.

Table 1 will show the evolution of the neutron yield as a function of the proton energy and target material. The materials used are already defined (C, Al, Fe, Ta, Hg, PbBi and natural U) in the input file; hence only the ASSIGNMAT card of region 3 needs to be changed. The students should choose 4 of them (for example, Al, Ta, Hg and nat

U) and run for the proton energies suggested in Table 1, by modifying WHAT-1 in the BEAM card of the input file.

Table 1. Neutron yields for different materials and proton energies.

		Proton energy (GeV)					
		0.1	0.25	0.5	1.0	1.5	2.0
Target material	C						
	Al						
	Fe						
	Ta						
	Hg						
	PbBi						
	Nat U						

Likewise, Table 2 should be filled with the neutron yields of different source particles (protons, deuterons, helium-4 and electrons). The name of the primary particles should be changed in the SDUM field of the BEAM card, using FLUKA's particle name (on-line manual: http://www.fluka.org/manual/Toc.html, section 5).

The data in these tables will show that, for energies below 1 GeV, the increase in neutron yield as a function of the energy follows an exponential pattern, whereas for energies above ~1.5 GeV the increase is reduced, roughly to a linear increase. GNUPLOT can be used to plot this data, producing results similar to Figure 5.(a), where the neutron yield evolution as a function of the energy is illustrated.

Figure 5.(b) shows the evolution of the neutron yield efficiency by normalising the previous results over the incident particle energy. This procedure effectively allows us to infer the optimum energy of neutron production through spallation.

Table 2. Neutron yields for projectile particles and incident energies.

		Energy of the projectile (GeV)					
		0.1	0.25	0.5	1.0	1.5	2.0
Projectile particle	Proton						
	Deuteron						
	Helium-4						
	Electron						

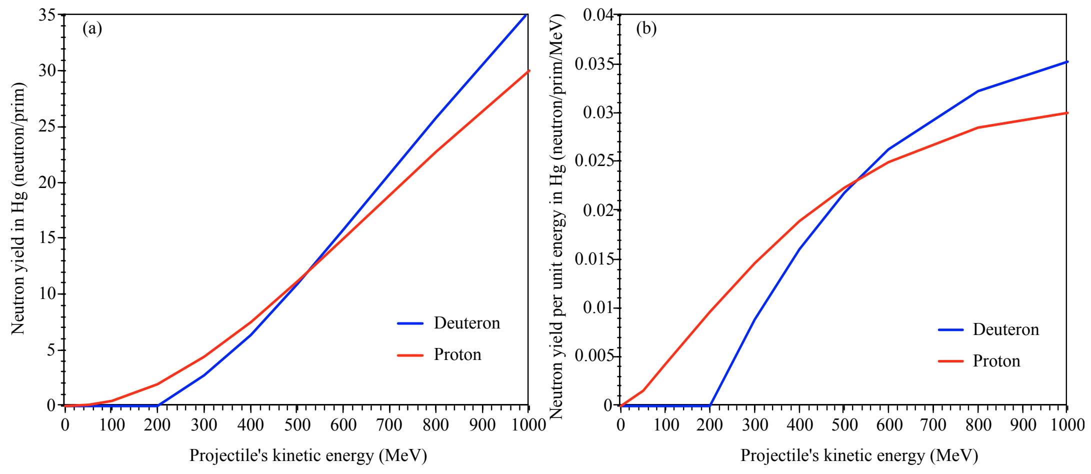


Figure 5. Neutron yield in Hg for different types of projectile.

Conclusion

We have now finished this brief introduction to the use of FLUKA. You should now be able to configure basic models and extract a wide range of results. For this purpose, the scripts used in the previous exercises are included in this document as annexes.

For more in-depth information on FLUKA and the physics behind it, visit the official website at *http://www.fluka.org*.

References

[1] *"FLUKA: Status and Prospective for Hadronic Applications"*, A. Fassò, A. Ferrari, J. Ranft, P.R. Sala, invited talk in the Proceedings of the MonteCarlo 2000 Conference, Lisbon, October 23--26 2000, A. Kling, F. Barao, M. Nakagawa, L. Tavora, P. Vaz eds., Springer-Verlag Berlin, p. 955-960, 2001. Also: *"Electron-photon transport in FLUKA: Status"*, A. Fassò, A. Ferrari, P.R. Sala, invited talk in the Proceedings of the MonteCarlo 2000 Conference, Lisbon, October 23--26 2000, A. Kling, F. Barao, M. Nakagawa, L. Tavora, P. Vaz eds., Springer-Verlag Berlin, p. 159-164, 2001. For additional information: *http://www.fluka.org*

[2] Yacine Kadi, *"The EA-MC Monte Carlo Code Package"*, in Proc. of the Fifth International Meeting on Simulating Accelerator Radiation Environment — SARE-5: Models and Codes for Spallation Neutron Sources, OECD Headquarters; Paris, France (July 2000).

[3] *"EURISOL DS; EURopean Isotope Separation On-Line Radioactive Ion Beam Facility Design Study"*, EC – FP6 Research Infrastructure Action-Structuring the European Research Area, Project Contract no. 515768 RIDS.

[4] *"Preliminary Study of the Liquid Metal Proton-to-Neutron Converter"*, A. Herrera-Martínez, Y. Kadi, EURISOL DS/TASK2/TN-05-01, http://eurisol-hg-target.web.cern.ch/eurisol-hg-target/default.htm

Annexes (Scripts used in the Exercise)

put_together.scrpt

```
#!/bin/bash
# Script to recover the outputs of FLUKA and convert them automatically


mkdir part-res

# Create the files with the path to the results
for file in *001_fort.?[0-9]; do
     numb=`echo $file | cut -d. -f2`
     ls -1 *.$numb > bin_${numb}.txt
     echo >> bin_${numb}.txt
     echo "All_bins_${numb}.out" >> bin_${numb}.txt
done

# Put the results together
for file in bin_2*.txt; do
     ~/FLUKA-files/Executables/usxsuw < ${file}
     numb=`echo $file | cut -d. -f1 | cut -d_ -f2`
     mv All_bins_${numb}_sum.lis ./part-res/
     #usrbdx.r 1 All_bins_${numb}_sum.lis bdx_${numb}
done

for file in bin_4*.txt; do
     ~/FLUKA-files/Executables/usbsuw < ${file}
done

for file in bin_7*.txt; do
     ~/FLUKA-files/Executables/ustsuw < ${file}
     numb=`echo $file | cut -d. -f1 | cut -d_ -f2`
     mv All_bins_${numb}_sum.lis ./part-res/
     #usrbdx.r 1 All_bins_${numb}_sum.lis track_${numb}
done

for file in bin_5*.txt; do
     ~/FLUKA-files/Executables/usrsuw < ${file}
done

mv *00[1-5]* ./part-res/
mv ./part-res/*001.out .
mv bin*.txt ./part-res/
mv *sum.lis ./part-res/
```

bin_prod.scrpt

```
#!/bin/bash

ulimit -d 1000000

tmp=`pwd`


for file in All_bins_4[1-9].out
do

part=`echo $file | cut -d_ -f3 | cut -d. -f1`


distance=target
scale="1.0"

minval=" "
maxval=" "
divn=3
mult=" "
binf=1


  exes="0.0 30.0 "
  maxexes=" 30"
  guays=" "
  maxguays=" "
  zets=" -10.0 90.0 "
  maxzets="100"
```

```
case "$part" in
  41)
     particle=Neutron
     units="Neutrons/cm2/prim"
     minval="1E-5"
     divn=4
   ;;
  42)
     particle=Primary
     units="Primaries/cm2/prim"
     minval="1E-8"
     divn=3
   ;;
  43)
     particle=Energydep
     units="GeV/cm3/prim"
     scale="1.0"
     minval="1E-7"
     divn=4
   ;;
  44)
     particle=NeutBalDens
     units="NeutrBal/cm3/prim"
     minval="1E-6"
     divn=4
   ;;
  45)
     particle=Proton
     units="Protons/cm2/prim"
     minval="1E-8"
     divn=3
   ;;
  46)
     particle=Photon
     units="Photons/cm2/prim"
     minval="1E-6"
     divn=4
   ;;
  47)
     particle=FissDens
     units="Fiss/cm3/prim"
     minval="1E-8"
     divn=4
   ;;
  48)
     particle=HE-FissDens
     units="HE-Fiss/cm3/prim"
     minval="1E-8"
     divn=4
   ;;
  49)
     particle=LE-FissDens
     units="LE-Fiss/cm3/prim"
     minval="1E-8"
     divn=4
   ;;
esac



# PGMDRV
 ~/trieste/exes/pgmdrv << target0
${distance}.geo
target0


# Inputs for pawlevbin

# 1.- PLOTGEOM file (def. PLOTGEOM.STORE): (return)
# 2.- Swap plotgeom axis (def=n)? (return)
# 3.- Rotate geometry ? (y,n), def n (return)
# 4.- In the geometry file -> X_min - X_max (def. provided)
# 5.- Y_min - Y_max (def. provided)
# 6.- Bin file: WRITE THE NAME OF THE FILE!!!!            <----------
# 7.- Density file: (???, return) NOW, NEG:  Scaling Factor  <----------
#     for TRADE [ 6.25E15 to conv to n/s/mA/cm2, 1.E6 to W/s/cm3]
# 8.- Threshold density (def. provided): (???, return)
# 9.- Which binning  WRITE THE NUMBER OF THE BINNING!!!!     <----------
#10.- Xmin, Xmax (def. provided)                             <----------
#11.- NX (def. N_X_max)
#12.- Ymin, Ymax (def. provided)                             <----------
#13.- NY (def. N_Y_max)
#14.- Zmin, Zmax (def. provided)                             <----------
#15.- NZ (def. N_Z_max)
#16.- Bin file: (??, return)
```

```
pawlevbin << target1




${file}


${binf}
${exes}
${maxexes}
${guays}
${maxguays}
${zets}
${maxzets}

target1

#Call the input for PAW


# Input file for PAW

# 1.- Workstation type (?=HELP) <CR>=1 :  1
# 2.- exec $FLUPRO/flutil/pawlev
# 3.- Input file (<CR>=pawlevh) (return)
# 4.- Name of the Postscript output file
# 5.- Metafile type (PS-PORT=-111, PS-LAND=-112, EPS=-113) (<CR>=-112) (return)
# 6.- Title
# 7.- Macro : factx ? (<CR>=1) (return)
# 8.- Macro : facty ? (<CR>=1) (return)
# 9.- Macro : geog ? (<CR>=) (n)
#10.- min and max val. in matrix, Enter Min       <---------------
#11.- max val. in matrix, Enter Max               <---------------
#12.- enter n. of divisions per decade            <---------------


paw << target2
1
exec ~/trieste/scripts/pawlev

${particle}_usrbin.eps

${particle} in ${distance}, (${units})

${mult}
n
${minval}
$[maxval]
${divn}

target2

done
```

plot_spectra.scrpt

```
#!/bin/bash
# Script to automatically plot the USRTRACK and USRBDX Spectra

if [ .$1 != "." ]; then

dataf=$1
outname=`echo $dataf | cut -d_ -f3`

gnuplot<< EOF

set title 'Neutron Flux Energy Spectrum'
set xlabel 'E (MeV)'
set ylabel 'Neutron flux (dn/dlnE/cm2/primary)'

plot '$dataf' us (sqrt(\$1*\$2*1E6)):(\$3*sqrt(\$2*\$1*1E6)/1E3) with lines title
'neutron flux',\
 '' us
(sqrt(\$1*\$2*1E6)):(\$3*sqrt(\$2*\$1*1E6)/1E3):(\$3*sqrt(\$2*\$1*1E6)/1E3*\$4*1E-2)
with yerr title 'err'

set logscale
set term post eps enhanced color
set output 'plot_${outname}_log.ps'
replot
```

```
set term x11
set out
replot

EOF

else
echo "Arg 1: Filename (4 column file, where column 1 and 2 contain the lower and upper
boundaries fo the bin,"
echo "column 3 holds the flux values and column 4 show the errors -> The script shows
a spectrum plot using gnuplot"
fi
```

process_resnuc.py

```
#!/usr/bin/python
# Read and plot the residual nuclei from a ninary file

import sys, math, os, shutil, glob, string, re

if len(sys.argv)!=2:
        print
        print >> sys.stderr, "Syntax: "+sys.argv[0]+" $1=binary resnuc file"
        print "Date: 13/04/05"
        print "Author: Adonai Herrera Martinez"
        print
        sys.exit(0)


ft=open('tmp.txt','w')
ft.write(sys.argv[1]+"\n \n")
ft.close()
os.system('usrsuw < tmp.txt')
os.remove("tmp.txt")

name=sys.argv[1].split('.')[0]
detect=name.split('_')[-1]

try:
        fileinp=open(name+"_sum.lis",'r')
except Exception:
        print >> sys.stderr, "ERROR processing  "+sys.argv[1]
        sys.exit(1)

fileout1=open('Z_residual_'+detect+'.txt','w')
fileout2=open('A_residual_'+detect+'.txt','w')


for line in fileinp:
        cutline=line.split()
        if not cutline: continue
        if cutline[0] in 'Z:':
               fileout1.write(cutline[1]+"   "+cutline[2]+" "+cutline[4]+'\n')
        if cutline[0] in 'A:':
               fileout2.write(cutline[1]+"   "+cutline[2]+" "+cutline[4]+'\n')

fileinp.close()
fileout1.close()
fileout2.close()
```

plot_residual.scrpt

```
#!/bin/bash
# Script to automatically plot the RESNUCLEI distribution

if [ .$1 != "." ]; then

dataf=$1
outname=`echo $dataf | cut -d. -f1`
echo $outname
gnuplot<< EOF

set xlabel 'Atomic Number (Z) or Mass Number (A)'
set ylabel 'Residual nuclei (nuclei/cm3/primary)'

plot '$dataf' us 1:2 notitle

set logscale y

set term post eps enhanced color
set output 'plot_${outname}.ps'
replot
```

```
set term x11
set out
replot

EOF

else
echo "Arg 1: Filename (3 column file, where column 1 is the atomic or mass number,
column 2 contains the quantity of residual nuclei"
echo " and column 3 holds the relative errors (%) -> The script shows a nuclei
distribution plot, using gnuplot"
fi
```