How to manage a run of FOOT simulation

G. Battistoni, S.M. Valle

How many events? How many FLUKA runs?

Typically, for the evaluation of detector performances we have been working so far with productions of the order of 10^7 primaries

Unpractical to make a single long run

Much better to produce k runs in parallel, considering more than 1 cycle per run. The size of k depends on the amount of available CPU cores

The number of primaries/cycle is something that depends on user's flavor and CPU speed. It is advisable to have cycles with duration not longer than a few hours.

How many events? How many FLUKA runs?

A common approach:
Copy the prototype of input file to k replicas (k will be the number of runs), paying attention to make different initializations of random number.

For instance: copy foot.inp to 16O_C2H4_200_1.inp, 16O_C2H4_200_2.inp, …, 16O_C2H4_200_20.inp

In each input file the **RANDOMIZE** data card has to be modified in such a way to put in WHAT(2) a different number (no rules or recommendation about this number, user's choice)

Personally I manage this by means of a simple code (**crea_input.cc**)
See on pcalien: **/disk4/Tutorial/Simulation/CreaInput/**

How many events? How many FLUKA runs?

Usage:

cp foot.inp 16O_C2H4_200.inp

./CreaInput/crea_input -n 20 -in 16O_C2H4_200 -out 16O_C2H4_200

This will create automatically 20 input files with different RANDOMIZE:

16O_C2H4_200_1R.inp
16O_C2H4_200_2R.inp
…
16O_C2H4_200_20R.inp

To be checked carefully in the input prototype!

1) Primary type (**BEAM** and **HI-PROPE** cards)
2) Primary energy **(BEAM** card)
3) Target material (for ex.: **ASSIGNMA Polyethy TARGET**)
4) Trigger flag to write events in the **USRICALL** card (2nd field: 6 means only events with ≥1 inelastic interaction in the target; 0 would mean an untriggered output)*
5) Number of primaries (**START** card)

- **Other output triggers exist but do not consider them (to be checked), other triggers can be studied, for instance an elastic event trigger has been implemented, but not yet committed.**
  Untriggered output means that ~98.5% of events are primaries not interacting in the target. They might interact in VTX, ITR, MSD or SCN and eventually die in CAL, producing there many particles ➞ Very large TXT file!

To be checked carefully in the input prototype!

1) There is an event debug flag in the **USRICALL** card: 1st field. Normally it should be 0. If >0 a verbose event debug output with many messages about the history of particle tracking is written on the *.log file
2) Heavy *log file. It is meaningful to do that only for single events. Otherwise it could be even too difficult to read the file
3) If the debug flag is <0, a file is created to allow the event display of tracks with Flair (advanced topic)

Launching the jobs

$FLUPRO/flutil/rfluka –e fluka_FOOT_mag.exe –N0 –M5 16O_C2H4_200_1R &

This will launch a run with 5 cycles for the input file 16O_C2H4_200_1R.inp

Depending on the number of available core it is possible to send several runs (each one for a different input file!!) in parallel

For each case a temporary fluka_xxxx directory will be created.

Progress of job can be checked looking at the *.out or *.err file in the temporary directory.
Tip: look for E+30 (grep «E+30» *.err)

Example: **380000** **620000** **620000** **1.0134497E-02** **1.0000000E+30**

No. of events processed

No. of events still to be processed

Average cpu time/event at that time

Tip:

- **to stop a cycle in an ordered way: in fluka_xxxx create fluka.stop (touch fluka.stop)**
- **To stop a run: touch rfluka.stop**

At the end of the job

As a first thing, before doing any other thing, check in the *.log files possible messages of error!!!!

If everything's OK you can proceed to build the root output file

For each cycle a *TXT.dat file wil be available in shoe/Simulation

Example: **ls -1 *TXT.dat**

16O_C2H4_200_10R001_TXT.dat
16O_C2H4_200_1R001_TXT.dat
16O_C2H4_200_2R001_TXT.dat
16O_C2H4_200_3R001_TXT.dat
16O_C2H4_200_4R001_TXT.dat
16O_C2H4_200_5R001_TXT.dat
16O_C2H4_200_6R001_TXT.dat
16O_C2H4_200_7R001_TXT.dat
16O_C2H4_200_8R001_TXT.dat
16O_C2H4_200_9R001_TXT.dat

ls -1 *TXT.dat > 16O_200.lis

will create a 16O_200.lis file containing the list of files to be processed to create the root output file

Inside the TXT file

Run header. This first line is unique for each TXT file.
It echoes the trigger conditions (written by **usrini.f**)

write(outunit,*) int(abs(fragtrig)), Ethrdep

```
 6   1.0000000000000000E-004
       96       136         1        12        47        11         0         0         0       626
```

…

This second line (event header) is written instead at the beginning of each event written onto output. For instance this example says that:

- **Event number of FLUKA is 96**
- **There are 136 particles in the particle data bank**
- **There is 1 hit in STC**
- **There are 12 hits in BM**
- **There are 47 hits in VTX**
- **There are 11 hits in ITR**
- **There are no hits in MSD, SCN and CALO**
- **There are 626 crossings**

After that all banks will follow (all these lines are written by **usreou.f**)

```
… write(outunit,*) ncase,nump,nSTC,nBMN,nVTX,nITR,nMSD,nSCN,nCAL,
  &       nCROSS
```

Processing the TXT files

The TXT files will be converted to a root tree by means of a code (**Txt2Root**):

1) The Txt2root code has to be compiled (the first time)
cd Simulation/TXT2ROOT
make

2) in shoe/Simulation
./TXT2ROOT/Txt2Root –in nome.lis -iL -out nome.root

This switch is used to say that nome.lis is a list of files

You may process a single TXT file:
./TXT2ROOT/Txt2Root –in TXT.dat –out nome.root
(notice that here you will not give -iL

Inside Txt2Root.cpp

```
//    loop sui file della lista ( if any)

  for(int idfile=0; idfile<numfiles;idfile++){
    cout<<endl<<"Now processing data from "<<infiles.at(idfile)<<"
file!"<<endl;
    ReadError = false;

    pfile = fopen(infiles.at(idfile),"r");

    nread= fscanf(pfile,"%d %lf\n",&fragtrig,&Ethreshold);
....
//    loop sugli eventi del file

   while((!feof(pfile))&&(!ReadError)){
...
nread= fscanf(pfile,"%d %d %d %d %d %d %d %d %d \n",&eve.EventNumber,
              &eve.TRn,&eve.STCn,&eve.BMNn,&eve.VTXn,&eve.ITRn,&eve.MSDn,
              &eve.SCNn,&eve.CALn,&eve.CROSSn);
...
```

Reads the run header

Reads the event header

Inside Txt2Root.cpp

```
nread = fscanf(pfile,
                "%d %d %d %d %d %d %d %lf %lf %lf %lf %lf %lf %lf %lf   %lf %lf %lf %lf %lf %lf
%lf %lf \n ",
                &eve.TRpaid[jj],&eve.TRgen[jj],&eve.TRcha[jj],
                &eve.TRreg[jj],&eve.TRbar[jj],&eve.TRdead[jj],
                &eve.TRfid[jj],&eve.TRix[jj],
                &eve.TRiy[jj],&eve.TRiz[jj],&eve.TRfx[jj],&eve.TRfy[jj],
                &eve.TRfz[jj],&eve.TRipx[jj],&eve.TRipy[jj],&eve.TRipz[jj],
                &eve.TRfpx[jj],&eve.TRfpy[jj],&eve.TRfpz[jj],&eve.TRmass[jj],
                &eve.TRtime[jj],&eve.TRtof[jj],&eve.TRtrlen[jj]);
```

This for example is the readout of the particle bank

All eve.* will be the leafs of the root tree