A customized FLUKA event by event output for low energy physics experiments. The case of FOOT

G. Battistoni, S.M. Valle
V. Patera

Introduction: a common need in many experiments

Typically there are experimental situations where MC is asked to produce data to be processed as real experimental data:

- event by event
- different particles in the same event,
- often in different detectors in the same apparatus
- each detector providing different kind of information: position, time, charge (energy release), ...
- Furthermore, people requires from MC the history of all detected particles in any event in order to verify the reliability of analysis and reconstruction algorithms

Goals of the work:

- provide a data structure satisfying these requirement and that can be used also by those who are not familiar with FLUKA

- provide data in a format/environment familiar for most people in a collaboration

Analysis Environment

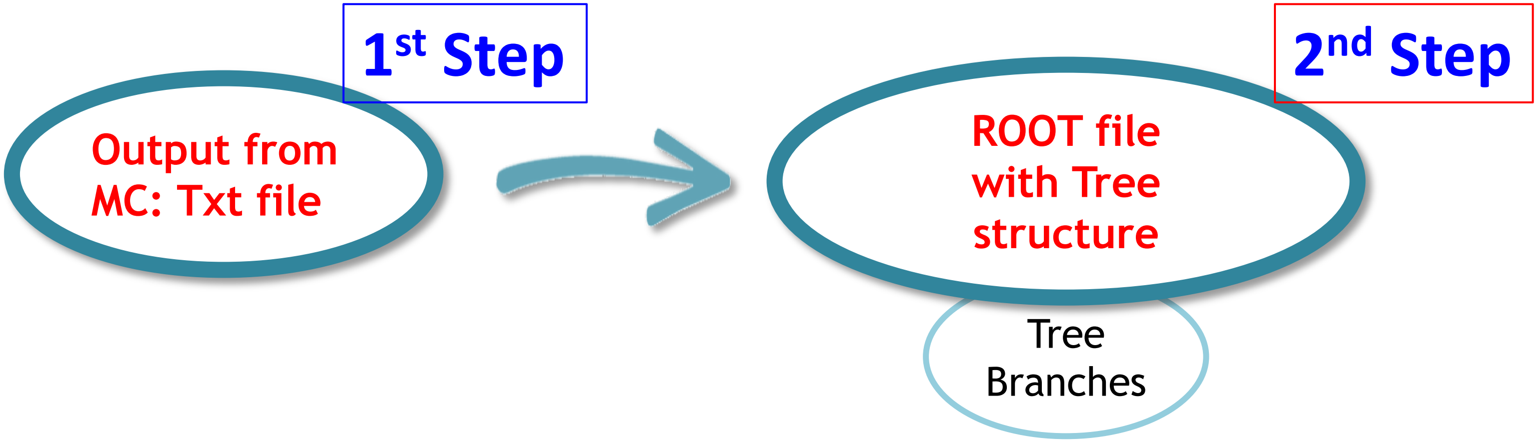
Based on ROOT

- Data Files are organized in a Root Tree event by event.
- For each event all relevant infos for each detectors are made available together with all kinematics and history of particle (primary + secondaries) participating to the event

Building our taylored MC Output in a Root Tree

We have configured some user routines of FLUKA to produce an "ad hoc" event-by-event output written as an ASCII file (*TXT.dat)

A simple and portable code reads these txt's and outputs ROOT files



FLUKA user routines used in FOOT - 1

mgdraw.inc : **custom include file** with array definitions and additional user variables; some parameters concerning geometry size, coord. etc.

usrini.f : Begin of run. It receives from data cards some user flags and possible thresholds to trigger data output. Recognizes and stores geometry names. **Writes run header on TXT file**

usrein.f : Begin of event. Zeroing of output arrays (defined in a user include file)

mgdraw.f (+ custom service routines) : managing the logic of the tree structure of event history. Entries used: mgdraw, sodraw, endraw, bxdraw, usdraw

mgdraw_lib.f : Not a standard FLUKA user routine. It contains the custom service routines that fill hits for every specific detector and for crossing borders as well

FLUKA user routines used in FOOT - 2

UpdateCurrentParticle.f: Not a standard FLUKA user routine. It manages the logic to recognize new created particles, beginning and end of history of each particle

magfld.f : reads the map of magnetic field and interpolates it at run time when tracking in a region with magnetic field on is requested

usreou.f : End of event. implements trigger logic for data output. **Writes output arrays on TXT file at the end of each event**

usrout.f : End of run. It does nothing important

parameters.inc: a custom include file with detector parameters that is automatically generated when producing the geometry with makeGeo

what there is inside parameters.inc

```
c     BEAM MONITOR PARAMETERS

      integer ncellBMN
      parameter (ncellBMN = 3)
      integer nlayBMN
      parameter (nlayBMN = 6)

c     VERTEX PARAMETERS

      integer nlayVTX
      parameter (nlayVTX = 4)
      integer xpixVTX
      parameter (xpixVTX = 928)
      integer ypixVTX
      parameter (ypixVTX = 960)
      double precision dxVTX
      parameter (dxVTX = 0.00207D+00)
      double precision dyVTX
      parameter (dyVTX = 0.00207D+00)
      ….
```

The compiling/linking script

Routines are stored in shoe/Simulation/ROUTINES

Compile+link scritpt is in shoe/Simulation: link_FOOT_mag.sh

```
#!/bin/sh
cd ROUTINES
$FLUPRO/flutil/fff  usrini.f
$FLUPRO/flutil/fff  usrein.f
$FLUPRO/flutil/fff  usreou.f
$FLUPRO/flutil/fff  usrout.f
$FLUPRO/flutil/fff  mgdraw.f
$FLUPRO/flutil/fff  magfld.f
$FLUPRO/flutil/fff  mgdraw_lib.f
$FLUPRO/flutil/fff  UpdateCurrentParticle.f

$FLUPRO/flutil/ldpm3qmd -m fluka usrini.o usrout.o usreou.o usrein.o  mgdraw.o m
agfld.o mgdraw_lib.o UpdateCurrentParticle.o -o fluka_FOOT_mag.exe

rm -rf  *.o
mv fluka_FOOT_mag.exe ../
cd ../
```

Compilation of routines

Linking of routines to create executable
shoe/Simulation/fluka_FOOT_mag.exe

Usage: source link_FOOT_mag.sh

A glimpse of FOOT user routines - 1

mgdraw.inc

As an example this is for instance the definition of all variables and arrays which constitute the «Particle Block»

```
....
 integer nump, maxnump
 parameter(maxnump=2000)

 integer idpa(maxnump), igen(maxnump)
 integer icha(maxnump), numreg(maxnump), iba(maxnump)
 integer idead(maxnump), jpa(maxnump)
 real vxi(maxnump),vyi(maxnump),vzi(maxnump)
 real vxf(maxnump),vyf(maxnump),vzf(maxnump)
 real px(maxnump),py(maxnump),pz(maxnump)
 real pxf(maxnump),pyf(maxnump),pzf(maxnump)
 real amass(maxnump), tempo(maxnump), tof(maxnump)
 real trlen(maxnump)
 common /particle_common/ vxi, vyi, vzi,
&     vxf, vyf, vzf,px, py, pz, pxf, pyf, pzf,
&     amass, tempo, tof, trlen, nump, idpa, igen,
&     icha, numreg, iba, idead, jpa
 .....
```

A glimpse of FOOT user routines - 2

usrein.f

As an example here you find the zeroing of all variables and arrays which constitute the «Particle Block» performed at the beginning of each event

```
 do ii = 1,min(nump,maxnump)
       idpa(ii)   = 0
       igen(ii)   = 0
       icha(ii)   = 0
       numreg(ii) = 0
       iba(ii)    = 0
       idead(ii)  = 0
       jpa(ii)    = 0
       vxi(ii)    = 0.
       vyi(ii)    = 0.
       vzi(ii)    = 0.
       vxf(ii)    = 0.
       vyf(ii)    = 0.
       vzf(ii)    = 0.
       px(ii)     = 0.
       py(ii)     = 0.
       pz(ii)     = 0.
       pxf(ii)    = 0.
       pyf(ii)    = 0.
       pzf(ii)    = 0.
       amass(ii)  = 0.
       tempo(ii)  = 0.
       tof(ii)    = 0.
       trlen(ii)  = 0.
c
       idfluka(ii) = 0   ! aux variables for particle latching
c
     end do
     nump = 0
```

A glimpse of FOOT user routines - 3

mgdraw.f

As an example here you find the point where, during a step in the transport of a particle, the energy deposition for the Start Counter is defined.

```
…
if( mreg.eq.nregSTC )then
      erawSTC = 0.
      IF ( MTRACK .GT. 0 )THEN
        do ii = 1,MTRACK
          erawSTC = erawSTC + dtrack(ii)
        end do
        IF ( LQEMGD )THEN
          RULLL  = ZERZER
          CALL QUENMG ( ICODE, MREG, RULLL, DTQUEN )
c
c     DTQUEN(MTRACK,1) e' il rilascio di energia quenchato nello start counter
c
          do ii = 1,mtrack
            equenchedSTC = equenchedSTC + dtquen(ii,3)
          end do
          equenchedSTC = equenchedSTC*abs_STC
        endif
      endif
      if(erawSTC.gt.0) then
        call score_STC(mreg,erawSTC,equenchedSTC,
   &     xtrack(0),ytrack(0),ztrack(0),xtrack(ntrack),ytrack(ntrack),
   &     ztrack(ntrack))
      endif
    endif
…
```

The score_STC routine, which actually fills the hit arrays for the Start Counter is in the file mgdraw_lib.f

A glimpse of FOOT user routines - 4

usreou.f

```
if(trigger) then
c
      write(outunit,*) ncase,nump,nSTC,nBMN,nVTX,nITR,nMSD,nSCN,nCAL,
    &      nCROSS
c
c   scrivo la banca delle particelle
c
      do ii = 1,nump
        write(outunit,*)idpa(ii), igen(ii), icha(ii),
    &         numreg(ii), iba(ii), idead(ii), jpa(ii), vxi(ii),
    &         vyi(ii), vzi(ii), vxf(ii), vyf(ii), vzf(ii), px(ii),
    &         py(ii),pz(ii),pxf(ii),pyf(ii),pzf(ii),amass(ii),
    &         tempo(ii),tof(ii),trlen(ii)
      end do
c
c   scrivo lo start counter
c
      do ii = 1,nSTC
        write(outunit,*) idSTC(ii),
    &         xinSTC(ii), yinSTC(ii), zinSTC(ii),
    &         xoutSTC(ii), youtSTC(ii), zoutSTC(ii),
    &         pxinSTC(ii), pyinSTC(ii), pzinSTC(ii),
    &         pxoutSTC(ii), pyoutSTC(ii), pzoutSTC(ii),
    &         deSTC(ii), alSTC(ii), timSTC(ii)
      end do
....
```

As an example here you find the point where, if the «trigger condition» is matched, the particle banck and hit arrays (here you see only Start Counter) are written onto the TXT file