Crossing regions: a quick overview

Yunsheng Dong

26/03/2021

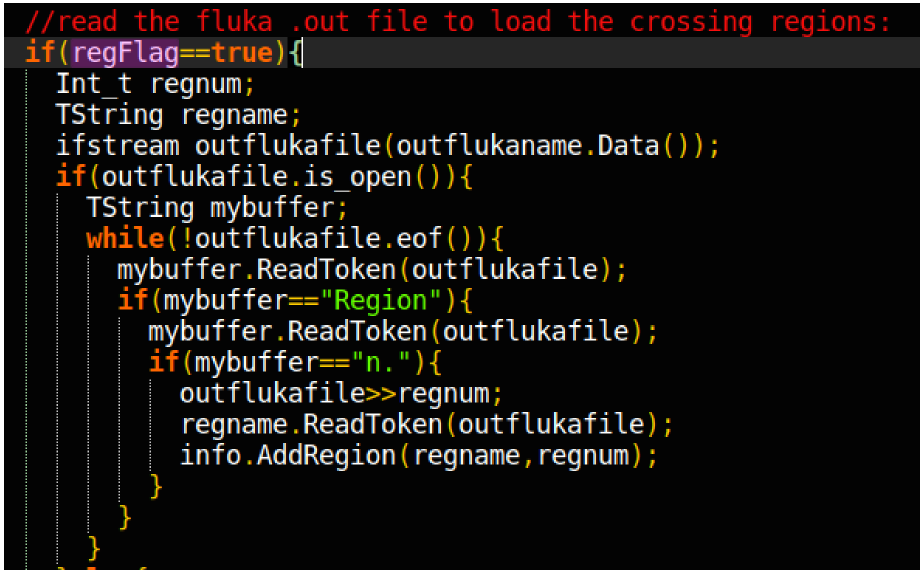
Crossing block of data

- **Information about the MC crossing block of data are present here:**

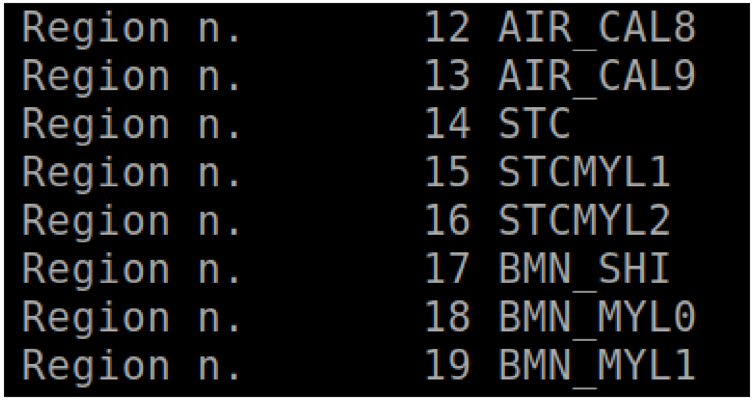
  https://docs.google.com/viewer?url=https%3A%2F%2Fagenda.infn.it%2Fevent%2F25103%2Fcontributions%2F127423%2Fattachments%2F78861%2F102103%2FFOOT_FLUKA.pdf

- This presentation will just show you how to retrieve the crossing region numbers to be used in the analysis softwares (avoiding hard coded numbers)

Txt2nturoot: input files



```
//read the fluka .out file to load the crossing regions:
if(regFlag==true){
  Int_t regnum;
  TString regname;
  ifstream outflukafile(outflukaname.Data());
  if(outflukafile.is_open()){
    TString mybuffer;
    while(!outflukafile.eof()){
      mybuffer.ReadToken(outflukafile);
      if(mybuffer=="Region"){
        mybuffer.ReadToken(outflukafile);
        if(mybuffer=="n."){
          outflukafile>>regnum;
          regname.ReadToken(outflukafile);
          info.AddRegion(regname,regnum);
        }
      }
    }
```



```
Region n.          12 AIR_CAL8
Region n.          13 AIR_CAL9
Region n.          14 STC
Region n.          15 STCMYL1
Region n.          16 STCMYL2
Region n.          17 BMN_SHI
Region n.          18 BMN_MYL0
Region n.          19 BMN_MYL1
```

For the Simulation guy:

- Add a FOOT.reg file in Reconstruction/level0/geomaps/expname/FOOT .reg with all the MC region numbers and names for the given simulation

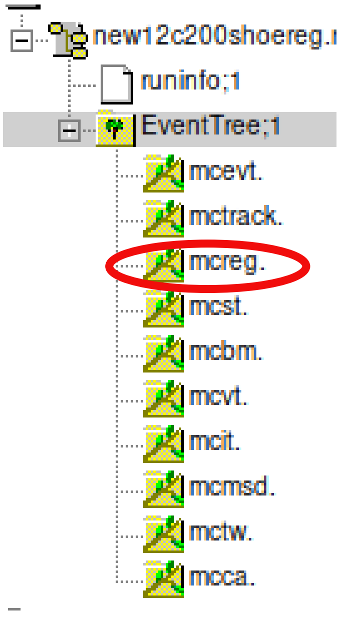
- Add a -reg flag in the ./Txt2NtuRoot command to enable the crossing regions

What is behind in the code:

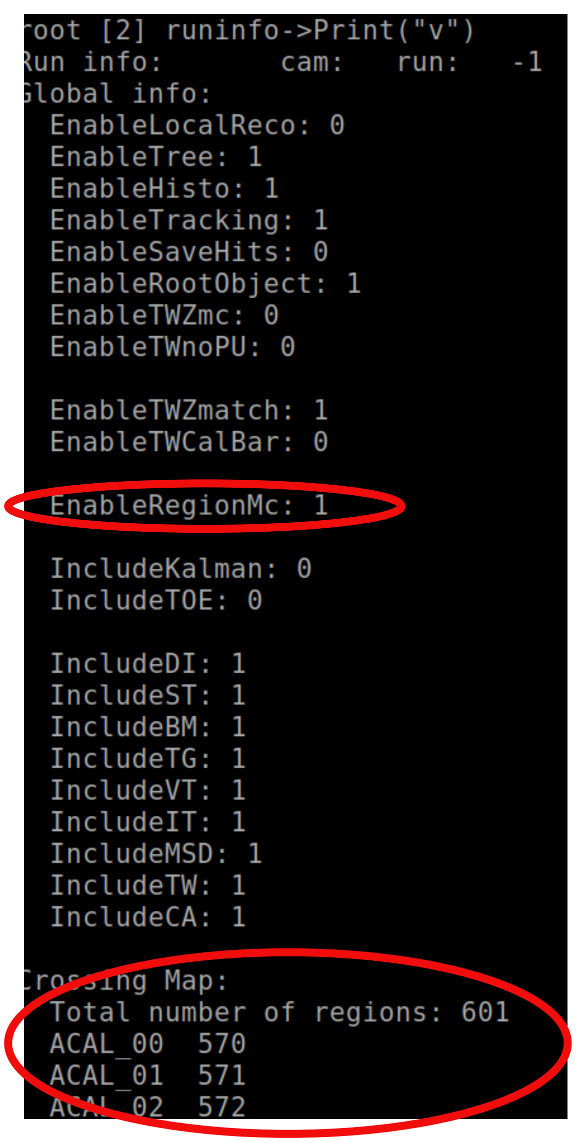
- -reg flag activates the m_enableRegionMc flag contained in GlobalPar

- Thanks to Chris, the region map is stored in Reconstruction/level0/geomaps/expname/FOOT .reg
  The region file is read and the crossing number and the region name are stored in a map contained in runinfo.

Txt2root: output file



```
new12c200shoereg.r
    runinfo;1
    EventTree;1
        mcevt.
        mctrack.
        mcreg.
        mcst.
        mcbm.
        mcvt.
        mcit.
        mcmsd.
        mctw.
        mcca.
```

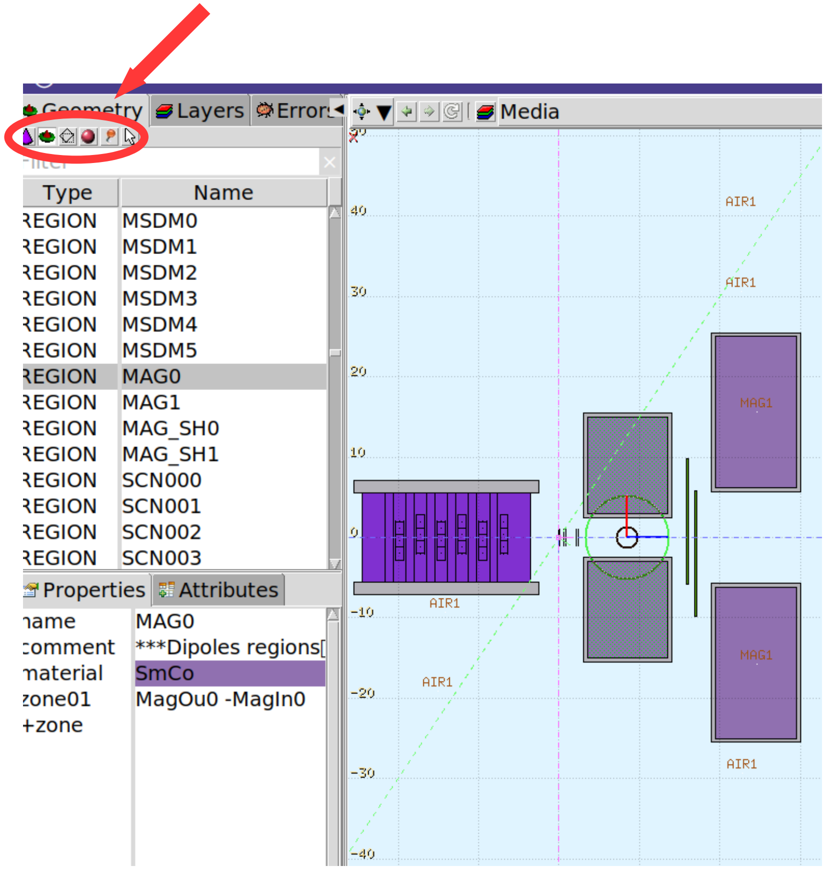


```
oot [2] runinfo->Print("v")
Run info:        cam:    run:    -1
lobal info:
  EnableLocalReco: 0
  EnableTree: 1
  EnableHisto: 1
  EnableTracking: 1
  EnableSaveHits: 0
  EnableRootObject: 1
  EnableTWZmc: 0
  EnableTWnoPU: 0

  EnableTWZmatch: 1
  EnableTWCalBar: 0

  EnableRegionMc: 1

  IncludeKalman: 0
  IncludeTOE: 0

  IncludeDI: 1
  IncludeST: 1
  IncludeBM: 1
  IncludeTG: 1
  IncludeVT: 1
  IncludeIT: 1
  IncludeMSD: 1
  IncludeTW: 1
  IncludeCA: 1

Crossing Map:
  Total number of regions: 601
  ACAL_00   570
  ACAL_01   571
  ACAL_02   572
```

- In the output file there should be a mcreg. Branch in the EventTree that contains all the crossing block of data
- And a runinfo with the global parameters and the crossing map.
- Type runinfo→Print() to check the presence of the crossing map
- Type runinfo→Print(“v”) or Print(“verbose”) to print also the crossing map

How to use the map: the fluka guy

Here the region box must be selected



Type	Name
REGION	MSDM0
REGION	MSDM1
REGION	MSDM2
REGION	MSDM3
REGION	MSDM4
REGION	MSDM5
REGION	MAG0
REGION	MAG1
REGION	MAG_SH0
REGION	MAG_SH1
REGION	SCN000
REGION	SCN001
REGION	SCN002
REGION	SCN003

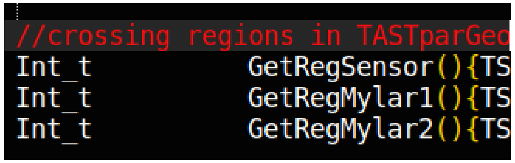
name	MAG0
comment	***Dipoles regions[
material	SmCo
zone01	MagOu0 -MagIn0
+zone	

- If you are familiar with fluka/flair, you can open the .inp file with flair and check the name of the region of interest.
- Then with with the region name you can retrieve the region number directly with runinfo→GetRegion() method

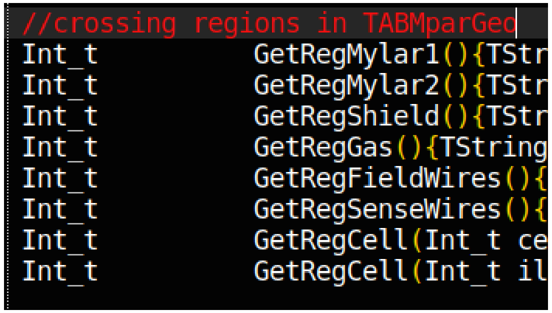
E.g.: I want to retrieve the region number of the first magnet:

- Open flair and click on the magnet and retrieve the region name (in this case "MAG0")
- To get the region name just use:
  TString regname="MAG0";
  runinfo→GetRegion(regname)
  If the region is present in the current mcfile, it will return the region number.
  Otherwise, it will return a -1 as output

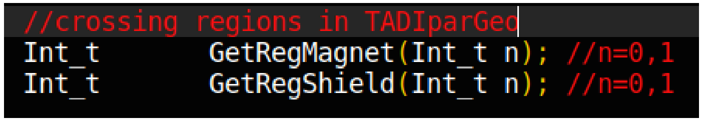
How to use the map: the shoe guy



```
//crossing regions in TASTparGeo
Int_t          GetRegSensor(){TS
Int_t          GetRegMylar1(){TS
Int_t          GetRegMylar2(){TS
```



```
//crossing regions in TABMparGeo
Int_t          GetRegMylar1(){TStr
Int_t          GetRegMylar2(){TStr
Int_t          GetRegShield(){TStr
Int_t          GetRegGas(){TString
Int_t          GetRegFieldWires(){
Int_t          GetRegSenseWires(){
Int_t          GetRegCell(Int_t ce
Int_t          GetRegCell(Int_t il
```



```
//crossing regions in TADIparGeo
Int_t        GetRegMagnet(Int_t n); //n=0,1
Int_t        GetRegShield(Int_t n); //n=0,1
```

- A more user friendly way to access to the fluka region numbers is to use the methods **GetReg*** added in the TA*parGeo files

  E.g.: the magnets are handled by TADIparGeo, and in TADIparGeo there are:
  GetRegMagnet(n) and GetRegShield(n) that provides the region number for the magnet and the shielding.
  To retrive the first magnet region you can use:
  pargeo->GetRegMagnet(0)

- The target region number can be retrieved from:
  TAGparGeo::GetRegTarget()

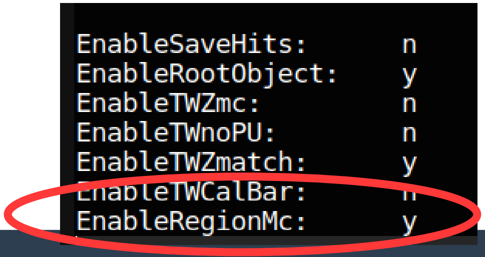
- The GetReg* methods have been developed based on what is currently written in the PrintRegions() methods

Other relevant info

- The region names (MAG0, STC, BMN_SHI) do not change among the different campaigns, only the region numbers can be different.
- if the PrintRegions() method of a detector needs to be changed, please change also the GetReg* methods accordingly
- If you want to use the crossings in a standalone analysis macro/code, be sure that the runinfo of the input mc file is correctly loaded:

```
TFile *f = new Tfile(inputnameFile.Data());

//open the file etc.

TAGrunInfo *p_runinfo=(TAGrunInfo*)(f→Get("runinfo"));
const TAGrunInfo runinfo(*p_runinfo);
gTAGroot.SetRunInfo(runinfo);

//From now on you can retrieve the crossing regions with both the fluka and shoe guy methods
```

- In order to analyze mc files that contains also the crossing block of data, be sure to correctly set the EnableRegionMC flag present in FootGlobal.par
- Thanks to Chris, the mc crossing block of data is propagated also in the DecodeMC output



```
EnableSaveHits:      n
EnableRootObject:    y
EnableTWZmc:         n
EnableTWnoPU:        n
EnableTWZmatch:      y
EnableTWCalBar:      n
EnableRegionMc:      y
```