CosmosX manual

Release 0.7.0

COSMOS X development team

May 01, 2022

CONTENTS:

CHAPTER

ONE

WHAT IS COSMOS X?

1.1 What can we do with COSMOS X?

COSMOS is a Monte Carlo simulation package to simulate atmospheric air showers produced by high energy particles hitting the earth called Cosmic Rays. Original COSMOS was developed by Prof. Kasahara since 1970's and continued up to version 8 in 2010's. The history and references to the older versions are found in the legacy web page [cosmosLegacy] . Together with COSMOS, a detector simulation tool EPICS was also developed by Prof. Kasahara.

Originally COSMOS 9 was developed to integrate the functions of COSMOS and EPICS into a single package. To clarify this extended feature, the new COSMOS is released with a new name COSMOS X, meaning eXtended COSMOS. This document concentrates on the description of the COSMOS X and does not assume any experience of previous COSMOS or EPICS.

Using COSMOS X,

- Users can inject any particle available in the list (even unstable particles) with any angle and energy.
- Users can define the energy and composition distributions for injection using a simple text file.
- Users can switch hadronic interaction models.
- Users can access information of individual particle in the shower using a so-called *userhook* function.
- Users can define format and condition of output file(s).
- Users can define the electric field and magnetic field.
- Users can set non-atmospheric materials such as water and soil.
- Users can set non-earth environment such as the Sun.

Through these flexibilities, COSMOS X is suitable for various kinds of cosmic-ray studies.

1.2 Structure

This section describes the outline of the COSMOS X structure for users to have a rough idea of how to use COSMOS X. More practical explanations are found in Section 3 and Section 4.

1.2.1 General structure

When the source package is unpacked, users can find the following subdirectories under the main directory *CosmosX_zzzzz/* where *zzzzz* designates the version number. Usually users need to make their own codes based on (copy, paste and edit) the examples in the *Application/* directory.

- **Cosmos/** Functions to handle air shower development are contained.
- **Epics/** Functions to handle interactions in material are contained.
- **LibLoft/** Common functions used for air shower and material are contained.
- **Script/** Useful scripts are contained.
- **Application/** Useful sample codes for first usage are contained. Usage of the *Application/Example/FirstKiss/* sample is the first step of COSMOS and detail is given in Section 2.4.

1.2.2 User's flexibility: 3 user control files

In the *FirstKiss* example, users can find several files to control the simulation, among them users are required to edit three files, *primary*, *param* and *chook.f*.

- **Primary particles** (*primary* **file**) This file contains information of the type and energy of the primary particle. In case of the FirstKiss example shown in Listing 1.1 , mono-energetic nitrogen nuclei (isotope with mass number 14 and atomic number 7) with kinetic energy per nucleon (KE/n) 100 GeV are injected. More examples of primary definition are found in the *LibLoft/Data/Primary/* directory. Detail description of the primary file is given in Section 3.1.
- **Simulation setup** (*param* **file**) This file contains information to setup the simulation. An example (slightly modified) of FirstKiss is shown in Listing 1.2 . The file is a list of parameters with their names and values. This format is called `namelist` in FORTRAN and easily read with a single read function. (See FORTRAN textbook for more detail.) Information of the observation site, injection angle, hadronic interaction model and any options are specified in this file. Definitions and formats of all available parameters are listed in Section C.
- **Simulation control and Output (userhook)** You may want to access the information of individual particle during tracking as it is available in the popular detector simulation tools like GEANT. To access the event-by-event information (primary particle, end of shower, etc), micro physics of air shower development, to histogramming the intermediate information and handle the output, and to add any artificial effects, some userhook functions users can define in the *chook.f* file are very useful.

Listing 1.1: The FirstKiss example of the primary file.

```
#
# 'e-' 'MeV' 'KE/n' 'd' 0 / 24-12 Mg 22-11 Na
#
#-----------------------------------------------------------
  'iso 14 7' 'GeV' 'KE/n' 'd' 0 /
         100      1.
           0.         0.
```

Listing 1.2: The FirstKiss example of part of the param file (modified)

```
&PARAM
LatitOfSite = 30.110000
LongitOfSite = 90.53
YearOfGeomag = 2019.500

DepthList = 3000 4000 6000 10000 0
ASDepthList = 2000 4000.0 6000.0 8000.0 .0 .0

SeedFile = 'Seed'
InitRN = 300798 -3319907
PrimaryFile = 'primary'
CosZenith = (0.9, 0.9)
Azimuth = (0.0,0.0)
HeightOfInj = 100.0e3
DestEventNo = 1000 2

Generate ='em'
ThinSampling = F
IntModel = '"phits" 2.0 "dpmjet3" '
IncMuonPolari = T
KEminObs = 8*100e-6 ! 100 keV
LpmEffect = T
MinPhotoProdE = .152

BaseTime = 10.0
Cont = F
ContFile = ' '
CutOffFile = ' '
Ddelta = 5.00
DeadLine = ' '
DtGMT = 8.00
Freec = T
Hidden = F
Job = ' ' ! This is comment after input data
ObsPlane = 1
OneDim = 0
SkeletonFile = 'SkeletonParam '
SourceDec = 30.0
TimeStructure = T
Trace = 21    ! for display with detector
TraceDir = './'
WaitRatio = 0.01
Within = 99999
Za1ry = 'cos 1'
&END
```

An example of FirstKiss *chook.f* in Listing 1.3 shows how to output particle information when a particle arrives at the predefined observation altitude.

Listing 1.3: The FirstKiss example of a part of the chook.f file (modified).

```
#include "cmain.f"
#include "chookHybAS.f"
!!! #include "ctemplCeren.f" not needed now
!  *************************************** hook for Beginning of a Run
!  *

      subroutine chookBgRun
      implicit none
#include "Zmanagerp.h"

!            namelist output
      call cwriteParam(ErrorOut, 0)
!            primary information
      call cprintPrim(ErrorOut)
!            observation level information
      call cprintObs(ErrorOut)
      end

!     *********************************** hook for Beginning of  1 event
!     *
      subroutine chookBgEvent
      implicit none
      integer:: num, cumnum
      call cpEventNo(num, cumnum)
      write(*,*) '## ev', num
      end


!     ************************************ hook for observation
!     *
      subroutine chookObs(aTrack, id)
!
      implicit none
#include "Ztrack.h"
#include "Zcode.h"
      integer id  ! input.  1 ==> aTrack is going out from
!                                  outer boundery.
!                           2 ==> reached at an observation level
!                           3 ==> reached at inner boundery.
      type(track):: aTrack
!
!     For id =2, you need not output the z value, because it is always
!     0 (within the computational accuracy).
!
      if(id .eq. 2) then
!     output typical quantities.
```

(continues on next page)

(continued from previous page)

```
            write(*, '(3i3, 1p, 3g15.4)')
     *           aTrack%where,  !  observation level. integer*2.  1 is␣
↪highest.
     *           aTrack%p%code, !  ptcl code.  integer*2.
     *           aTrack%p%charge, !  charge,  integer*2
     *           aTrack%p%fm%p(4), ! total energy in GeV
     *           aTrack%pos%xyz%r(1), aTrack%pos%xyz%r(2) !  x, y in m
      endif
      end

!    ********************************** hook for end of 1 event
!    * At this moment, 1 event generation has been ended.
!    *
      subroutine chookEnEvent
```

A schematic explanation of the relationship between these files, functions and system functions is shown in Fig. 1.1. Here `cosmosLinuxGfort` is the name of executable file in case of the Linux environment.

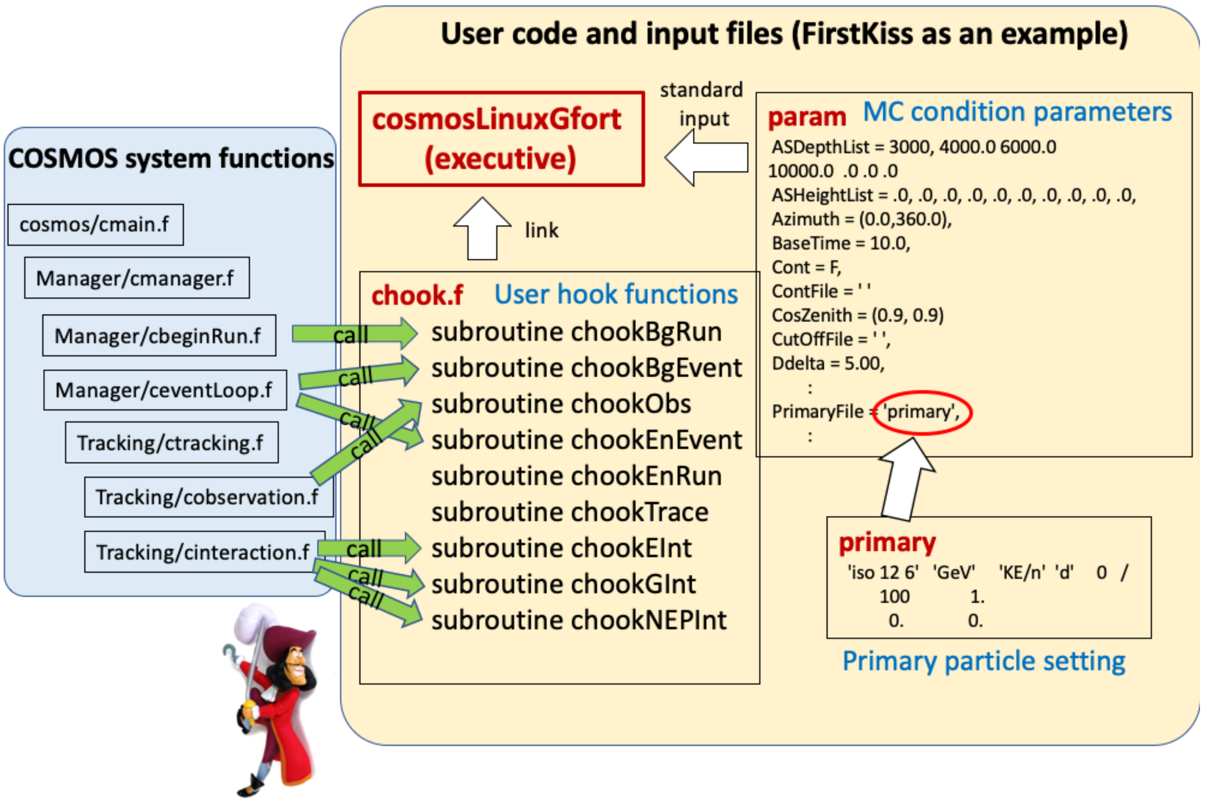

Fig. 1.1: Relation between the COSMOS system functions, userhook functions and parameter files.

Guides to more flexibilities such as arbitrary atmospheric profile, electric field, magnetic field, non-atmospheric material, non-earth sphere are given in Section 4.

1.2.3 What we can not do (now)?

- Simulation with arbitrary shape of medium
- User code with C++ (in development)

CHAPTER TWO

HOW TO USE COSMOS X FOR THE FIRST TIME?

2.1 Environment

COSMOS X is developed mainly under Linux and Mac OS X with Intel or gFortran compilers. Both conventional `make` and recent `cmake` are available for compilation, but the development team eventually unifies to `cmake`. List of successful combinations of OS version, distribution, compiler version are given in the official COSMOS web page,

http://cosmos.icrr.u-tokyo.ac.jp/COSMOSweb/

It is clear that the list is not complete. Any feedback (both success and failure) from the users to cosmos@icrr.u-tokyo.ac.jp are highly welcome.

2.2 Download

The source codes of COSMOS are packed in a single tar.gz file. First download the latest version from the COSMOS page introduced in Section 2.1 .

2.3 Installation

Installation procedure depends if cmake is available or not. The scripts are prepared for bash/zsh[1] .

- `gunzip CosmosX_zzzzz.tar.gz` to unpack the package, where `zzzzz` designates the version number.
- `tar xvf CosmosX_zzzz.tar` to unpack the tar archive.
- `cd CosmosX-zzzzz`, where is called *CosmosX* directory hereafter
    1. If you can use cmake
        - `less Doc/HowToBuildByCMake.txt` to know the procedure

[1] If you use (t)csh, please set the environment variables as follows instead of running *SetEnvironment.sh*.

```
setenv SHOWERMCTOP /somewhere/CosmosX-zzzzz
setenv LIBLOFT   $SHOWERMCTOP/LibLoft
setenv COSMOSTOP $SHOWERMCTOP/Cosmos
setenv HOST $HOSTNAME # if $HOST is not defined
```

   - `source Script/SetEnvironment.sh` to set environment variables. You can answer *yes* for 2 questions. The top directory of COSMOS X is assigned to the environment variable *$COSMOSTOP*.
   - `./Script/CompileLibraryByCMake.sh` to make library
   - if you find *lib/LinuxGfort/libshowermc.a* compilation is successful (in case of using Linux gFortran.)

2. if you can not use cmake (use legacy make)
   - `cd Cosmos`
   - `ls Site` to find a *site.configXXX* file fitting to your environment where *XXX* designates architecture and compiler
   - `cp Site/site.configXXX site.config` to copy a proper config file in the *Cosmos/* directory
   - `cd ..`, then repeat same in the *LibLoft* directory
   - move back to the *CosmosX* directory
   - `source Script/SetEnvironment.sh` to set environment variables. You can answer *yes* for 2 questions. The top directory of COSMOS X is assigned to the environment variable *$COSMOSTOP*.
   - TEMPORAL NOTE: if your meet an error message `-bash: ${yn,,}: bad substitution`, remove `,,` in *SetEnvironment.sh*.
   - `./Script/CompileByLegacyMake.sh` to make library
   - if you find *lib/LinuxGfort/libshowermc.a* compilation is successful (in case of using Linux gFortran.)

2.4 Test program (First Kiss)

The related files of the first example application are found in the *Application/Example/FirstKiss/* directory. Let us play with this example.

2.4.1 Compile and Run

1. If you can use cmake
   - move to the CosmosX directory

```
./Script/CompileExampleByCMake.sh ./Application/Example/FirstKiss
cd Application/Example/FirstKiss
```

2. if you can not use cmake

```
cd Application/Example/FirstKiss
make clean -f Makefile.legacy
make -f Makefile.legacy
```

- if you find `cosmosXXX` executable file, compilation is successful, where `XXX` depends on your system (`XXX` =LinuxGfort, for example)
- `./cosmosLinuxGfort < param` to start a simulation lasting for some minutes depending on the system

According to the primary and param files, this sample code simulates two showers of Nitrogen primary with kinetic energy of 100 GeV/n.

2.4.2 Track visualization

After a simulation, you can find trace files such as trace1 and trace2 for each primary injection. If you are in the *FirstKiss/* directory and be able to launch ROOT, type

```
cd Vis
root ReadTraceMacro.C
```

to visualize the shower image as shown in Fig. 2.1 . The ROOT code explains how to visualize the trace information. You can use your preferable graphical libraries. Because the size of the trace files gets large, to inactivate the trace output you can edit param file and change `Trace=0`.

Format of the trace file in case of `Trace=21` as found in the *FirstKiss* sample is:

```
x y z PID Ekin Q t
```

where `(x, y, z)` is the position of the particle in meter in the coordinate system with z-axis directing vertical (detector system, See Section B), `PID` is the particle identification code (Section E.1), `Ekin` is the particle kinetic energy in GeV, `Q` is the charge in e and `t` is the time in nsec. Options of the `Trace` value are summarized in Table 3.1.

Output of trace file is defined in *Cosmos/Tracking/cputTrInfo.f*. When you want to define the output of the trace file as you like, set the Trace value larger than 100 and edit the subroutine `chookTrace` in *chook.f*. Detail is given in Section 4.7.

2.4.3 Userhook output

As seen in Listing 1.3, the main code of *FirstKiss*, *chook.f*, contains a subroutine `chookObs(aTrack, id)` , which outputs particle information in air shower. As seen in Fig. 1.1, `chookObs(aTrack,id)` subroutine is called from the system subroutine `cobservation()` in *Cosmos/Tracking/cobservation.f* every time a particle reaches at the observation levels.The observation levels can be defined in the *param* file with a variable *DepthList* in $\mathrm{kg/m^2}$, or alternatively *HeightList* in m.

The variable `aTrack` given from the system subroutine has a structure type track defined in *Cosmos/include/Ztrack.h*. The member variables defined in the track structure are listed in Table 2.1 . To access these variables, as found in chook.f, we can use such as `aTrack%where` and `aTrack%p%mass`.

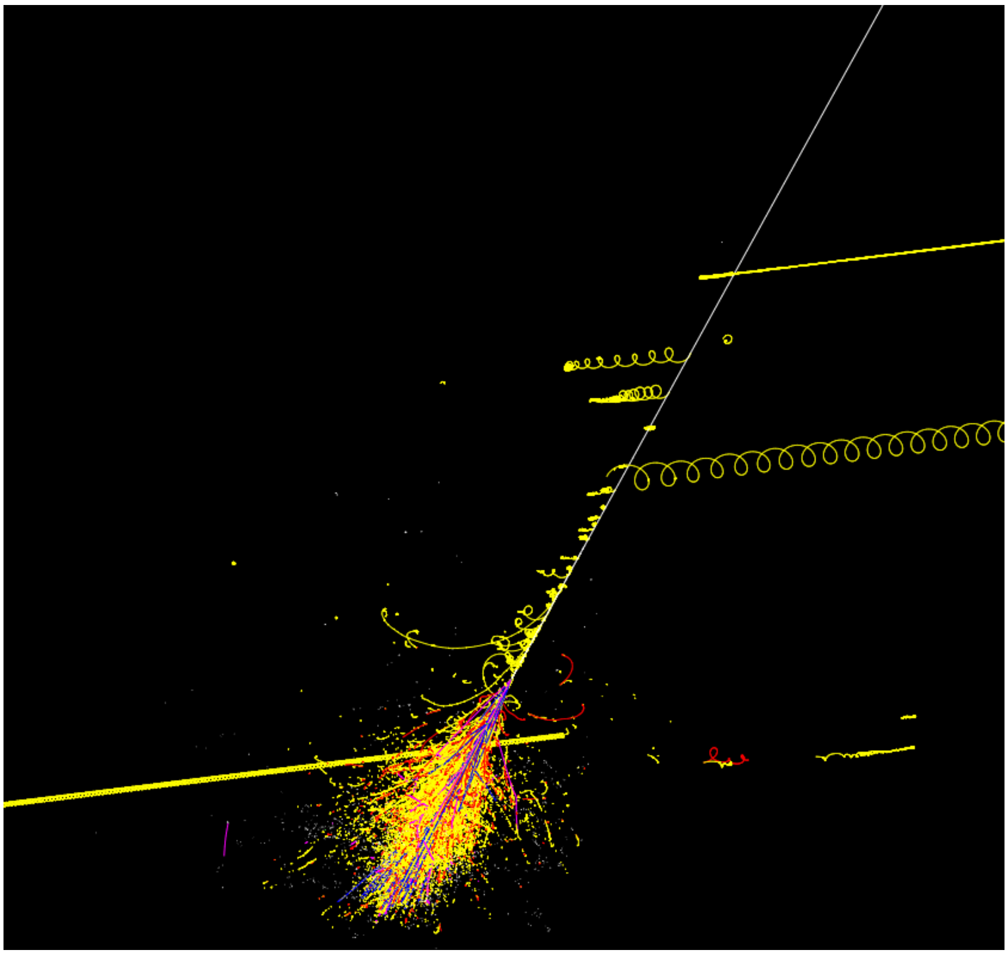

Fig. 2.1: ROOT visualization of a 100 GeV/n nitrogen shower in the *FirstKiss* sample.

Table 2.1: Definition of the variables found in track structure

variable	variable type	description
p	struct ptcl	particle attributes defined in *Zptcl.h*
pos	struct position	position, structure defined in *Zpos.h*
t	real*8	time in length/beta (m)
vec	struct direc	
wgt	real*4	weight for thin sampling
where	integer*2	current obsSite no. (0 is initial value)
asflag	integer*2	non 0, if As has been generated from this ptcl (only for electrons)
user	real*8	user use
If LABELING>0		
label	integer	put a label (1,2,…) on each particle. There is a global label counter which is cleared at the start of 1 event generation. It is counted up when a particle is poped up from the stack. The label counter is given to the label of the poped up particle. This may be needed to judge if the same particle crosses a given observation place more than once.
info	integer	for each particle, when a particle is born this is initialized to 0. If the ptcl goes higher than 380km, 1 is added. This is for AMS observation.

CHAPTER THREE

HOW TO EDIT THE USER CONTROL FILES?

3.1 primary file

As found in Listing 1.1, the primary file is composed of multiple text lines of two categories.

The first category such as:

```
'iso 14 7' 'GeV' 'KE/n' 'd' 0/
```

contains particle type, unit of energy, definition of energy, spectrum type and power of the flux normalization. Following list describes detail of each item, but more practical examples are found in the *CosmosX/LibLoft/Data/Primary/* directory, especially the sample.d file contains the explanations and lists of available symbols.

- **particle type** : The sample above indicates an isotope of mass number 14 and atomic number 7, *i.e.*, Nitrogen. This is a general definition of nucleus. For other elementary and composite particles such as electrons and protons specific names, *e, e-, electron* and *p, proton* are defined, respectively. A list of definitions is found in the *sample.d* file.
- **unit of energy** : Unit of energy is defined either in *eV, MeV, GeV, TeV, PeV* and *EeV*.
- **definition of energy** : Definition of energy is available in total energy per particle (*E/P* or *E*), kinetic energy per particle (*KE/P* or *KE*), energy per nucleon (*E/n*), kinetic energy per nucleon *(KE/n)*, momentum per particle (*p/P* or *p*) and momentum per nucleon (*p/n*). Energy and momentum are considered in the same unit with the speed of light $c = 1$.
- **spectrum type** : Definition of the following flux values in differential (*d*) or integral (*i*).
- **flux power** : Definition of the following flux values in flux $\times$ E^X Power X is indicated.
- **option (min and max)** : As options, you can add two more values to specify the minimum and maximum energies to simulate.

Then the following lines as the second category such as:

```
100 1.
0.  0.
```

contain the energy and flux. The line with two zeros means the end of the flux data. Single non-zero line as shown here is used to make a mono-energetic incident.

To learn how primary file works, try a sample *PrimaryHowTo*. Here only cmake compile procedure is explained. For make compilation, please refer Section 2.4.

- move to the *CosmosX* directory

- `./Script/CompileExampleByCMake.sh ./Application/Example/PrimaryHowTo`
- `cd Application/Example/PrimaryHowTo`
- `./cosmosLinuxGfort < param > out`
- If you can use ROOT, `root primary_analysis.cpp`

This sample shows mixed composition primary generation without any shower simulation. After output primary information, particle track is killed in `chookBgEvent` subroutine in *chook.f*. Any *primary* file you define can be tested based on this sample.

3.2 param file

Users will edit this file frequently. As seen in Listing 1.2, various parameters can be specified. The *param* file is separated in two blocks. The first block enclosed by `&PARAM` and `&END` contains general parameters while the second block with `&HPARAM` contains special parameters which users do not need to edit in the usual use. Explanation of all parameters are given in Section C. Here some important parameters are introduced.

- **DepthList** : List of the observation depths in $\mathrm{kg/m^2}$ in increasing order. Userhook subroutine `chookObs` is called when a particle passes this depth. Zero means the end of the list. Negative depth values are replaced with the values in the *HeightList*.
- **HeightList** : List of the observation heights in m in decreasing order. Userhook subroutine `chookObs` is called when a particle passes this height. End of the list is specified by zero in the `DepthList`. So the `HeightList` must be accompanied by an appropriately edited `DepthList` like *"-1 -1 -1 -1 "* when the user wants to define height values.
- **ASDepthList** and **ASHeightList** : Same definition as `DepthList` and `HeightList` but used when Generate specifies *'as'*.
- **DestEventNo** : Number of injection to be simulated. If two numbers are given like `DestEventNo = 10000 1000`, the program run will stop at the completion of the second numver of events (1000). You may continue the job with `cont=t` function (see below) for another set of (1000) events, and repeat such a process until all the first number (10,000) of events are finished.
- **CosZenith** : Ragne of the cosine zenith angle of the primaries. Set `CosZenith = (0.5, 1.0)` if you want to inject at the zenith angle between 60 degrees and vertical. Note that this zenith angle is the direction of primary particles measured at the deepest observation level. At other depths, the value is not strictly the same.
- **Azimuth** : Range of the azimuthal angle of the primaries in degrees.
- **PrimaryFile** : Name of the file where the primaries are defined.
- **Generate**: When `Generate = 'as'`, electro-magnetic showers are replaced with an analytical calculation of the B-approximation. Number of particles at the depths (or heights) specified in the `ASDepthList` (or `ASHeightList`) are stored. As the result is given only in 1-dimensionally, this option is useful for a fast calculation of longitudinal development or hadronic components. If you want to simulate full em showers (3D Monte Carlo) in parallel to the B-approximation, `Generate = 'em/as'` can do it. Without specification of this parameter, full em showers are always active.
- **Trace** : Output of particle track information is controlled as seen in the *FirstKiss* example in Section 2.4. Because the size of the trace output is large, when you simulate many showers, do not forget to set `Trace = 0`. To define custom output, set the Trace value more than or equal to 100

and less than 160. Then edit the userhook subroutine `chookTrace` in *chook.f*. See also Section 3.3.

- **InitRN** : Initial random number seeds of the first event. If you put a negative number for the second value, like `InitRN = 12345 -77777333`, Cosmos will use a timer value and the host name to make the initial seed.

3.3 userhoook subroutines

Various userhook subroutines are prepared in *chook.f*. Each subroutine is called when predefined conditions are satisfied in the main routines of COSMOS X. Users can edit the contents of the userhook subroutines to access the particle information, to control the output format and so on. Explanation of each userhook subroutine is given below.

- **chookBgRun** : This subroutine is called when the program starts.
- **chookBgEvent** : This subroutine is called every time a new injection happens. To record the information of individual primary particle, edit this routine.
- **chookObs(aTrack, id)** : This subroutine is called every time individual particle passes the observation levels defined by `DepthList` or `HeightList`.
- **chookEnEvent** : This subroutine is called when all particle tracking of each shower is completed. To output the statistics of each shower, for example, edit this routine.
- **chookEnRun** : This subroutine is called when all shower trackings are completed.
- **chookTrace** : This subroutine is called every time a particle moves when $100 \leq Trace < 160$. Track information before and after movement can be accessed through the variables `TrackBefMove` and `MovedTrack` defined as track structure found in Tab.1.
- **chookEInt(never)** : This subroutine is called when an electron (or positron) interaction occurs.
- **chookGInt(never)** : This subroutine is called when a gamma-ray interaction occurs.
- **chookNEPInt(never)** : This subroutine is called when a non-electromagnetic interaction occurs.

CHAPTER

FOUR

HOW TO OPTIMIZE MY SIMULATION?

4.1 Hadronic interaction model

ON GOING

Hadronic interaction models are selected through *IntModel* in the *param* file. An example found in the *FirstKiss* sample is:

```
IntModel = '"phits" 2.0 "dpmjet3" ',
```

meaning PHITS and DPMJET3 are used below and above 2 GeV, respectively.

How can we know the list of available models and the version of each model? Move to the directory *$COSMOSTOP/LibLoft/Script*, and type:

```
./intModel.sh
```

You can find the choice of high energy hadronic interaction models, *qgsjet2, epos* and *sibyll*, and current version. If you want to switch between the versions, you can select from the

4.2 Thinning

4.3 AS, hybrid method

4.4 Magnetic field

The magnetic field is controlled by the parameter `HowGeomag`. By default, `HowGeomag` is 11 where constant magnetic field will be applied, calculated by the IGRF model at the date of `YearOfGeomag` at the place of the detector system at (`LatitOfSite`, `LongitOfSite`).

Table 4.1: HowGeomag parameter

HowGeomag	description
1	no magnetic field is taken into account until the first collision. The field is position dependent and calculated with the IGRF model at each position.
2	magnetic field exists everywhere. The field is position dependent and calculated with the IGRF model at each position.
11	same as 1 but constant. The field is calculated with the IGRF model at `BaseL`.
12	same as 2 but constant. The field is calculated with the IGRF model at `BaseL`.
21	same as 1 but constant. The field is defined by `MagN`, `MagE` and `MagD` parameters in `&HPARAM`.
22	same as 2 but constant. The field is defined by `MagN`, `MagE` and `MagD` parameters in `&HPARAM`.
31	same as 1
32	same as 12
others	same as one of the above

In most cases, `HowGeomag=11` is recommended but when `Reverse` is not 0 (back-tracking is enabled), please set `HowGeomag=2`.

When you want to define arbitrary magnetic field, in environment such as another planet, you have to define subroutine `cmyBfield( yearin, pos, MagF, icon )` and `ObjFile` to enable the subroutine.

4.5 Electric field

4.5.1 Introduction

Simple but unrealistic electric fields can be used without any coding by the user. It may be used to see the basic effect of electric field on charged particle motion. If the user wants to use more realistic field effect, it is better to make the cosmos library following the procedure described in Section 4.5.3.

In every case, the field strength must be given in unit of V/m. The final field vector, $\vec{\mathcal{E}}$, must be given in the E-xyz system.

4.5.2 Simple electric field

An electric field can be specified by referring to the height(H), distance to the shower axis (R) and time information (T) of each charged particle, where H is the height in m a.s.l, R (in m) the horizontal distance if `DefofR='h'` (default) or perpendicular distance if `DefofR='p'`, T the time (in ns) spent from the starting point of the primary particle.

- If T is used, H is neglected.
- So the field is determined by H and R or T and R.
- If R is not given, only H or T is used.
- If neither H nor T is used, only R is used.
- If non of H,T,R is used, the filed will be 0.

To specify *H,T,R*, a variable, `myEf` and its components are used. For example, if the user want to give an electric field at $0 < H < 1000$ and $2000 < H < 3000$, respectively

```
myEf(1)%H1=0
myEf(1)%H2=1000
myEf(2)%H1=2000
myEf(2)%H2=3000
```

may be given in the *param* file. Corresponding field vectors may be given as

```
myEf(1)%Ef=Ex,Ey,Ez
myEf(2)%Ef=Ex',Ey',Ez'
```

where `Ex` etc are numerical values in V/m. The vectors must be given in the detector system (vertically upward direction is the +*Z* direction. Internally, the values are converted into the ones in the E-xyz system.

The height list by `myEf` must be given from lower ones (note: the observation height list in the *param* file is given from higher to lower height order). For *T, R*, the same format is used. The max number of fields is 5.

To activate the specifications by `myEf`, `HowEfield=1` must be given in the `&HPARAM` section of *param* file. Its default is 0 which means non-existence of the electric field.

4.5.3 Arbitrary electric field

To use more realistic fields, the user must define a subroutine whose name is `cmyEfield(aTrack, Efout)`, where `aTrack` is the track information in the E-xyz coordinate system and `Efout` is the electric field the user should define in the E-xyz system. And finally `HowEfield=2` should be specified in the `&HPARAM` section of *param* file at run.

Even if `cmyEfield()` is defined as above, the user can set `HowEfield=0` or 1.

Let's assume the filename *cmyEfield.f* where `cmyEfield(aTrack, Efout)` is defined.

1. First, copy `$COSMOSTOP/cosmos/cmain.f` to your project directory. And comment out `#include "cmyEfield.f"`.
2. Edit *cmyEfield.f* to define customized electric field, `cmyEfield(aTrack, Efout)`.
3. Edit makefile.
   - If you use *cmake*, edit *CMakeLists.txt* to add *cmyEfield.f*.

     ```
     #add_executable(cosmos${ARCH} chook.f)
     add_executable(cosmos${ARCH} chook.f cmyEfield.f)
     ```

   - If you use *make*, edit *chook.mk*, Notice that *cmyEfield.o* have to be added, not *cmyEfield.f*.

     ```
     #objs = chook.o
     objs = chook.o cmyEfield.o
     ```

4. Build executable as described in Section 2.4.

An example project to enable arbitrary electric field can be found in *Application/Example/MyField/*, which will be useful as a template.

4.6 Non-air material, non-earth sphere

Usually air shower simulation codes handle only air (mixture of N_2 , O_2 and other rarer gas) as a medium. Also usual simulation codes assume a flat atmosphere or a spherically symmetric atmosphere centered at the center of the earth. However, COSMOS X allows to arrange non-air material such as water and soil. Their shapes are limited in shells with a common center, but the radius is not limited to the radius of the earth. This means COSMOS X is able to simulate air showers in the other planets, or stars including their ground.

To define non-standard environment, users can use a variable **ObjFile** in the param file such as:

```
ObjFile = ”obsfile”
```

Here *obsfile* is a file name that contains actual definition of the environment. When the *ObjFile* variable is not specified in the *param* file, usual atmosphere is setup.

4.7 User defined Trace output

CHAPTER

FIVE

HOW TO USE COSMOS X EPICS?

CHAPTER

SIX

REFERENCES

APPENDIX

A

UNIT OF PHYSICAL QUANTITIES

The unit is based on SI. The following units are used internally and users will obtain outputs from COSMOS in those units in principle. It is also very important to note that all real variables are given in the double precision; users are recommended to output them in the single precision to save the disk space if necessary.

- **Length** : m.
- **Energy** : GeV. Note, however, users can specify the energy of primary particles in a variety of units, such as MeV, MeV/n, TeV etc., or in momentum.
- **Magnetic field strength** : Tesla. Note that 1 Gauss is 10^{-4} T.
- **Thickness of air** : $\mathrm{kg/m^2}$. $\mathrm{1g/cm^2}$ is $\mathrm{10kg/m^2}$ ($\mathrm{1000g/cm^2} = 10,000\mathrm{kg/m^2}$). The air density is in $\mathrm{kg/m^3}$.
- **Time** : sec. However, time in the `chookObs` routine is already converted to nsec. The default Cerenkov output contains time factor in (length in cm)/beta for saving output space.
- **Angle** : Angles in degree are used to specify the latitude and longitude of the observation place. The declination angle is also in degree.

APPENDIX

B

COORDINATE SYSTEM

B.1 The E-xyz system

Fig. 2.1 illustrates the basic coordinate system which is internally used in Cosmos. The x-axis is directed to the longitude 0 and latitude 0. The y-axis is to the 90 degrees east and the z-axis to the north. The Earth is expressed by a complete sphere. The origin is at the center of the Earth. This coordinate system is abbreviated as the E-xyz system.

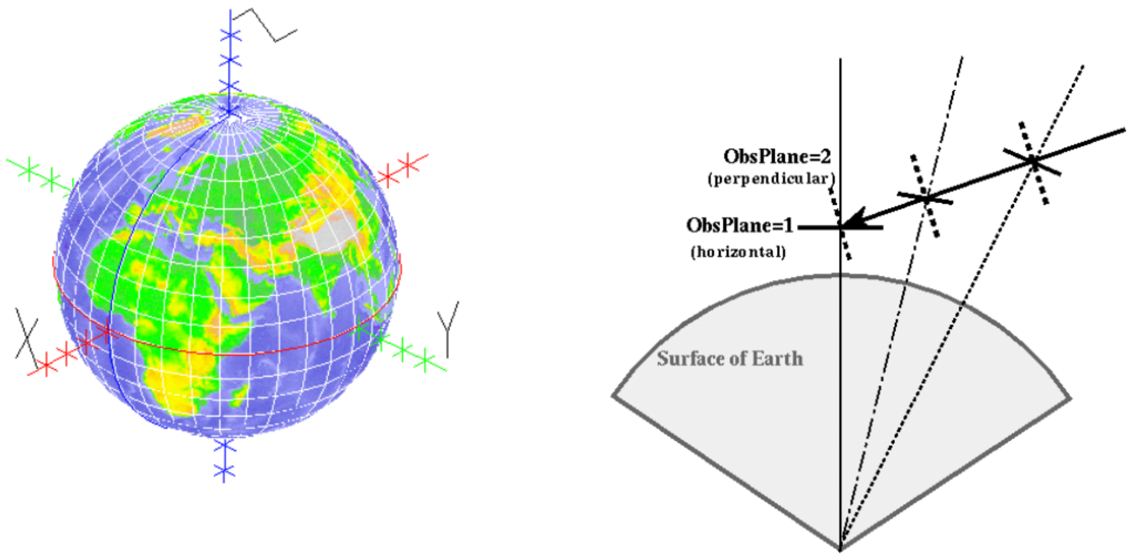

Fig. 2.1: The basic (interal) coordinate system (left Fig.) and observation plains (right Fig.)

B.2 The detector system and the primary system

You may setup several observation levels, at each of which you can record particle information passing through it. The plane of the levels may be horizontal or perpendicular to the primary particle direction. Since detectors are normally placed horizontally, we call the horizontal recording system the “detector system”. (See Fig. 2.1 right). From the figure the user might misunderstand that partiles are recorded when they cross a plane which is tangential to the surface of a sphere (this was acutually true once in older versions), but it is the surface of a sphere, *i.e.*, particles are recorded when they cross this spherical surface. However, the x-y plane of the coordinate system is tangetial as it is the rectangular system. The x-axis of the system is directed to the magnetic east (default), the y to the magnetic north and the z to the vertical.

The coordinate system whose x-y plane is perpendicular to the primary is called the “primary system”. Hence the z-axis is the primary direction (upward is positive), the x-axis is directed to the vector product of the z and the vertical direction, the y to the vector product of the z and the x-axis direction.

In both cases, the origin is assumed to be the location of a specified observation place. If the primary direction is the vertical, both systems are almost identical, as far as near z-axis particles are concerned.

APPENDIX

C

PARAMETER LIST

[Parameter lists are read in *LibLoft/Manager/creadParam.f*.]

In this appendix, explanation of all parameters available in the *param* file is listed. The parameters are classified in two categories named `&PARAM` and `&HPARAM` as found in the *param* file. The former is for a standard use while the latter is for a special use. The parameters are read as `namelist` of FORTRAN. Header files defining these variables are found under *CosmosX/LibLoft/Header/ZincForNameL.h*. Some general rules of the description are

- Separator between values can be either a space or a comma.
- Comment out is defined by a '!' at the first column of a line.
- If two values are given when a single value is requested, the first value is ignored.

C.1 Parameter list of *&PARAM* in *param* file

Table 3.1: List of *&PARAM* parameters

Parameter name	Unit	Description example	Simple explanation and reference in this manual
Site and Time			
LatitOfSite	degrees	30.11	Latitude of the lowest observation height (>0 for north)
LongitOfSite	degrees	90.53	Longitude of the lowest observation height (>0 for east)
YearOfGeomag	year	2019.500	Year to determine the geomagnetic field
Sampling height			
ASDepthList	kg/m^2	2000.0 4000.0 6000.0 8000.0	List of AS sampling depth from higher altitude to lower. Negative is ignored. ASHeightList has a priority. See Section 4.3 for AS.
ASHeightList	m	6000.0 4000.0 2000.0 .0	List of AS sampling height from higher altitude to lower. Negative is ignored. More priority than ASDepthList.
DepthList	kg/m^2	2000.0 4000.0 6000.0 8000.0 0.0	List of particle sampling depth. chookObs subroutine is called at these depths.

continues on next page

Table 3.1 – continued from previous page

Parameter name	Unit	Description example	Simple explanation and reference in this manual
HeightList	m	6000.0 4000.0 2000.0 0.0	List of particle sampling height. chookObs subroutine is called at these heights.
Initialization and Primary injection			
SeedFile			
InitRN		300798 13319907	Two integers as initial random number seed.
		0 -1	With negative second value, random seed is generated according to the system clock, process ID and host IP address.
PrimaryFile		'primary'	File name of primary spectrum data
CosZenith		(0.5, 1.0)	Range of cosine zenith angle of primary injection
Azimuth	degree	(0.0, 360.0)	Range of azimuth angle of primary injection
HeightOfInj	m	100.0e3	Primary injection height above the deepest observation height
DestEventNo		1000 2	Two integers as number of events to be generated
Simulation options			
Generate		'em/as'	Method to generate electromagnetic shower. see Section 4.3
ThinSampling			
IntModel			
IncMuonPolari			
KEminObs			
LpmEffect			
MinPhotoProdE			
PhotoProd			**Obsolete**. Remove this if existing in old samples
BaseTime			
Cont			
ContFile			
CutOffFile			
Ddelta			
DeadLine			
DtGMT			
Freec			
Hidden			
Job			
ObsPlane			
OneDim			
SkeletonFile			
SourceDec			
TimeStructure			

continues on next page

Table 3.1 – continued from previous page

Parameter name	Unit	Description example	Simple explanation and reference in this manual
Trace			Format of trace output. Section B for coordinate system.
		0	No trace output (recommended in mass production)
		1	(x,y,z) in the primary system in meter.
		11	(x,y) in meter and z in kg/m2 in the primary system.
		21	(x,y,z) in the detector system in meter.
		31	(x,y) in meter and z in kg/m2 in the detector system.
		100	When more than or equal to 100 and less than 160, `chookTrace` is called.
			More in *Cosmos/Doc/ParamUsage1*.
TraceDir			
WaitRatio			
Within			
Za1ry			

C.2 Parameter list of *&HPARAM* in *param* file

SUBROUTINES AND VARIABLES

D.1 User accessible types

`type coord`

During the paticle tracking, this system is used.

Type fields

- **`% r`** (3) *[real*8]*
- **`% sys`** *[character(4)]* :: which system. one of 'xyz', 'llh' or, 'sph'.

Header Zcoord.h

`type position`

location of a particle

Type fields

- **`% xyz`** *[coord]* :: in xyz
- **`% radiallen`** *[real(8)]* :: in m . radial length
- **`% depth`** *[real(8)]* :: in kg/m2 depth
- **`% height`** *[real(8)]* :: in m. vertical height(from sea level
- **`% colheight`** *[real(8)]* :: in m. where the latest nuclear collision took place. (iniitial value is very large value).

Header Zpos.h

`type direc`

Type fields

- **`% w`** *[coord]*
- **`% coszenith`** *[real(8)]* :: cos of the zenith angle. it is defined as follows: Let's assume w and position are given in xyz sytem.

Header Zdirec.h

`type fmom`

Type fields

- **`% p`** (4) *[real(8)]* :: four momentum in GeV. p(1) is x component. Note. Momentum is given in the Earth xyz system.

`type ptcl`

particle at production

Type fields

- **% fm** *[fmom]* :: 4 momentum
- **% mass** *[real(8)]* :: mass
- **% code** *[integer(2)]* :: particle code
- **% subcode** *[integer(2)]* :: used mainly to identify paticle/antiparticle if the difference is important. To set particle, "ptcl" is used. anti-partilce, 'antip" is used for particles For particles of which partilce/antiparticle nature can be judded by its code and charge, the user need not specify it when using cmkptc subroutine. give 0. subcode for gamma ray may be used to identify brems gamma and direct gamma by kdiretg, kcasg
- **% charge** *[integer(2)]* :: charge

Header Zptcl.h

`type track`

full particle attributes in Cosmos

Type fields

- **% p** *[ptcl]* :: basic ptcl attributes.
- **% pos** *[position]* :: position
- **% t** *[real(8)]* :: time in length/beta (m)
- **% vec** *[direc]* :: direction
- **% wgt** *[real(4)]* :: weight for thin sampling
- **% where** *[integer(2)]* :: current obsSite no. (0 is initial value)
- **% asflag** *[integer(2)]* :: non 0, if As has been generated from this ptcl (only for electrons)
- **% user** *[real(8)]* :: user use
- **% inip** *[ptcl]* :: Particle at production
- **% inipos** *[position]*
- **% init** *[real(8)]* :: time in length/beta (m)
- **% inivec** *[direc]*
- **% parp** *[ptcl]* :: parent particle
- **% parvec** *[direc]*
- **% label** *[integer]* :: (if LABELING > 0) put a label (1,2,...) on each particle. There is a global label_counter which is cleared at the start of 1 event generation. it is counted up when a particle is poped up from the stack. The label_counter is given to the label of the poped up particle. This may be needed to judge if the same particle crosses a given observation place more than once.

- **% info** *[integer]* :: (if LABELING > 0) for each particle, when a particle is born this is initialized to 0. If the ptcl goes higher than 380km, 1 is added. This is for AMS observation.

Header Ztrack.h

`type element`

Type fields

- **% A** *[real(8)]* :: mass number
- **% Z** *[real(8)]* :: atomic number
- **% N** *[integer]* :: nucleon number. N-Z is number of neutrons.

Header Zelement.h

`type epmedia`

Type fields

- **% noOfElem** *[integer]* :: actual number of elements
- **% elem** (maxElements) *[element]*
- **% No** (maxElements) *[real(8)]* :: number of each element; copied to OrigNo and then normalized. so that sum be 1.0
- **% OrigNo** (maxElements) *[real(8)]* :: number of each element
- **% MolMass** *[real(8)]* :: olar mass normally no unit but in that case the value should be the same as the one in the unit of g/mol. default init value 0 will be given just before reading the data. Normally not used except for atmosphere. For the gas media, better to be given.
- **% w** (maxElements) *[real(8)]* :: No(i)A(i)/sum(No(i)A(i)) same note as No
- **% npercm3** (maxElements) *[real(8)]* :: # of i-th element /cm3=No/Ai*rho*wi same note as No
- **% nsigma** (maxElements) *[real(8)]* :: No(i)s_i
- **% sumns** *[real(8)]* :: sum of above
- **% ndpsigma** (maxElements) *[real(8)]* :: No(i)dps_i for DP
- **% sumndps** *[real(8)]* :: sum of above
- **% sumNo** *[real(8)]* :: sum of OrigNo(:)
- **% colElem** *[integer]* :: element # at which interaction took place
- **% colA** *[integer]* :: A of such one:. int(elem(colElm)%A+0.5)
- **% colZ** *[integer]* :: Z of such one: int(elem(colElm)%Z)
- **% colXs** *[real(8)]* :: x-section for that target(mb)
- **% xs** *[real(8)]* :: this is xs for the media.
- **% ndensity** *[real(8)]* :: effective number density /cm^3
- **% wp** *[real(8)]* :: plasma frequency x hbar (GeV)

- **% n** *[real(8)]* :: refractive index
- **% nd** *[real(8)]* :: number of ingredients / cm^3
- **% A** *[real(8)]* :: sum No x Ai
- **% Z** *[real(8)]* :: sum No x Zi
- **% Z2** *[real(8)]* :: sum No x Zi**2
- **% ZZ1** *[real(8)]* :: sum No x Zi(Zi+1) for electron; this * t /(gamma beta2)^2 = Xc^2 (t g/cm2)
- **% MoliereForXc2** *[real(8)]*
- **% MoliereExpb** *[real(8)]* :: exp(b) = t x this (t in g/cm2)a for z=1 and beta =1. this = 6702 sum/A sum = Sum No x Zi^(1/3)(Zi+1)/(1+3.327(Ziz/137)^2)
- **% Z1_3rd** *[real(8)]* :: <Z^1/3> not <Z>^(1/3)
- **% Z2_3rd** *[real(8)]* :: <Z^2/3> not <Z>^(2/3)
- **% mbtoPgrm** *[real(8)]* :: 10^-27 x N0/A. If multiplied to sigma in mb, we obtain probability / (g/cm2).
- **% mbtoPkgrm** *[real(8)]* :: mbtoPgrm/10d0
- **% mbtoPcm** *[real(8)]* :: rho x mbtoPgrm. If multiplied to sigma in mb, we obtaind probability / cm
- **% mbtoPX0** *[real(8)]* :: mbtoPgrm x X0g. If multiplied to sigma in mb, we obtain probability /radation length. next ones are used when we approximate a compound /molecule as an atom
- **% mbtoPgrm2** *[real(8)]*
- **% mbtoPcm2** *[real(8)]*
- **% mbtoPX02** *[real(8)]*
- **% Z2byAeff** *[real(8)]* :: sum wi x Zi**2/Ai
- **% Z5byAeff** *[real(8)]* :: sum wi x Zi**5/Ai
- **% Aeff** *[real(8)]* :: sum wi x Ai
- **% Z2eff** *[real(8)]* :: Z2byAeff x Aeff
- **% Zeff** *[real(8)]* :: sqrt(Z2eff)
- **% Zeff3** *[real(8)]* :: Zeff**(1/3)
- **% LogZ** *[real(8)]* :: log(Zeff)
- **% A2eff** *[real(8)]* :: sum wi x Ai^2
- **% ZbyAeff** *[real(8)]* :: sum wi x Zi/Ai
- **% I** *[real(8)]* :: average ionization potential energy in GeV.
- **% rho** *[real(8)]* :: density in g/cm^3
- **% X0** *[real(8)]* :: radiation length. in cm
- **% X0m** *[real(8)]* :: radiation lenght in m.

- **% X0g** *[real(8)]* :: radiation length. in g/cm^2
- **% X0kg** *[real(8)]* :: reaiation length in kg/m^2 =X0g*10
- **% gtocm** *[real(8)]* :: g/cm^2 to cm.
- **% kgtom** *[real(8)]* :: gtocm*1.0d-3: kg/m2 to m.
- **% dEdxatp3m** *[real(8)]* :: dE/dx at p=3me for electron. ~ Ecrit
- **% Ecrit** *[real(8)]* :: GeV for electron
- **% Ecritmu** *[real(8)]* :: GeV for muon
- **% rhoc** *[real(8)]* :: comp.rhoc is copied whenever new comp. comes note;this is real*8 while comp.rhoc is real*4
- **% gasF** *[integer]* :: flag for gas. If 1, media is gas, 0 –>solid
- **% name** *[character(8)]* :: name of media
- **% format** *[integer]* :: format of the basic table. (1 or 2)
- **% s1** *[real(8)]* :: Migdal's s1
- **% logs1** *[real(8)]* :: log(s1)
- **% basearea** *[real(8)]* :: pi x Re**2 * N* Z /A *X0g = 0.15 Z/A*X0g
- **% cScrC1** *[real(8)]* :: const which appears in the complete screening crossection
- **% cScrC2** *[real(8)]* :: the other such one
- **% cScrMain** *[real(8)]* :: (4/3C1 + C2)
- **% BirksC1** *[real(8)]* :: quenching correction coef.
- **% BirksC2** *[real(8)]*
- **% BirksCC** *[real(8)]*
- **% Birks** *[character(1)]* :: flag to identify what quenching correction should be applied using BirksC1, etc.
- **% srim** *[integer]* :: index for srim data in module srimdata
- **% tbl** *[bpTbl]*
- **% sh** *[sternh]*
- **% cnst** *[SmpCnst]*
- **% pe** *[photoE]*
- **% urb** *[urban]*
- **% mu** *[mubpn]*
- **% xcom** *[epxcom]*

Header Zmedia.h

`type SmpCnst`

Type fields

- `% CompScrE` *[real(8)]* :: Energy above which we can use complete screening cross-sections. evaluate at Eg/Ee=x= 0.99.
- `% BrScrE` *[real(8)]* :: below this, partial screened cross-section is needed (= ComScrE)
- `% BremEgmin` *[real(8)]* :: min. ratio Eg/Ee for brems at high energy region
- `% BremEemin` *[real(8)]* :: Below this, partial screening brems x-section is not made. (Seltzer table 1 is used)
- `% BremLEemin` *[real(8)]* :: log10 of BremEemin
- `% BremEeminLPM` *[real(8)]* :: Min. energy of e+/e- above which LPM can be applied, if wanted.(Acutal application will be done if, Ee > Flpm*this and LPMeffect=T. Flpm and LPMeffect can be controled by epicsfile
- `% BremTXTL` *[integer]* :: Size of the Brems total x-section table. in the energy region BremEemin ~ BrScrE: ~log10(BrScrE/BremEemin)*10
- `% BremEsize` *[integer]* :: Size of log10 energy for 2D table for brems in the region A. BremTXTL/2
- `% BremUminLA` *[real(8)]* :: min of uniform random number in the region A at energies BremEemin ~ BrScrE: 0.1
- `% BremUmaxLA` *[real(8)]* :: max of uniform random number in the region A at energies BremEemin ~ BrScrE: 1.0
- `% BremUszLA` *[integer]* :: Size of uniform random nubmbers for 2D table for brems in the region A.: 20
- `% BremdULA` *[real(8)]* :: step of u in region A at Low energies
- `% BremdETXL` *[real(8)]* :: log10E step for brem total cross secton log10(BrScrE/BremEemin)/(BremTXTL-1)
- `% BremdEL` *[real(8)]* :: log10 step for 2D brem table at low energies
- `% BremUminLB` *[real(8)]* :: min of uniform random number in the region B0. sqrt(u)
- `% BremUmaxLB` *[real(8)]* :: max. sqrt(BremUminLA)
- `% BremUszLB` *[integer]* :: u talbe size in region B. 20
- `% BremdULB` *[real(8)]* :: step of u in B
- `% PairEgmin` *[real(8)]* :: min. Eg above which pair cross section is computed 1.1 MeV. However, at energies from PairEgmin to PairNonSc, B.H original xsection is used as xsec= Norm * B.H where Norm * B.H(10MeV) = Pair(10MeV)
- `% PairNonSc` *[real(8)]* :: see above.
- `% PairLEgmin` *[real(8)]* :: log10 of PairEgmin
- `% PairEgmaxL` *[real(8)]* :: Eg where LPM effect starts to appear
- `% PrScrE` *[real(8)]* :: below this, screened cross-section is used.
- `% PairTXTL` *[integer]* :: Size of the Pair total x-section table. in the energy region PairEgmin ~ PairEgmaxL; log10(PairEgmaxL/PairEgemin)*10

- **% PairEsize** *[integer]* :: Size of log10 energy for 2D table for pair in the region A,B. PairTXTL/2
- **% PairUminLA** *[real(8)]* :: min of uniform random number in the region A at energies PairEgmin ~ PairEgmaxL: 0.05
- **% PairUmaxLA** *[real(8)]* :: max of uniform random number in the region A at energies PairEgmin ~ PairEgmaxL: 1.0
- **% PairUszLA** *[integer]* :: Size of uniform random nubmbers for 2D table for pair in the region A.: 20
- **% PairdULA** *[real(8)]* :: step of u in region A at Low energies
- **% PairdETXL** *[real(8)]* :: log10 step of total pair cross-sec at low E
- **% PairUminLB** *[real(8)]* :: min of uniform random number in the region B 0. sqrt(u)
- **% PairUmaxLB** *[real(8)]* :: max. sqrt(PairUminLA)
- **% PairUszLB** *[integer]* :: u talbe size in region B. 20
- **% PairdULB** *[real(8)]* :: PairdULB=(PairUmaxLB-PairUminLB)/(PairUszLB-1))
- **% PairdELA** *[real(8)]* :: log10(PairEgmaxL/ PairEgmin) /(PairEsize-1)
- **% PairdELB** *[real(8)]* :: sqrt(log10(PairEgmaxL/ PairEgmin))/(PairEsize-1)
- **% BrEeminS** *[real(8)]* :: for Seltzer cross-section; lower energy region
- **% BrEgminS** *[real(8)]* :: Eg min for Seltzer Brems. (not ratio 1keV)
- **% BrLEeminS** *[real(8)]*
- **% BrEemaxS** *[real(8)]*
- **% BrTXTS** *[integer]*
- **% BrES** *[integer]*
- **% BrUminSA** *[real(8)]*
- **% BrUmaxSA** *[real(8)]*
- **% BrUszSA** *[integer]*
- **% BrdUSA** *[real(8)]*
- **% BrdETXS** *[real(8)]*
- **% BrdES** *[real(8)]*
- **% BrUszSB** *[integer]*
- **% BrUminSB** *[real(8)]*
- **% BrUmaxSB** *[real(8)]*
- **% BrdUSB** *[real(8)]*
- **% BrEeminS2** *[real(8)]* :: for Seltzer cross-section; higher energy region upto 10 GeV

- **% BrEgminS2** *[real(8)]* :: Eg/Ee min for Seltzer Brems.
- **% BrLEeminS2** *[real(8)]*
- **% BrEemaxS2** *[real(8)]*
- **% BrTXTS2** *[integer]*
- **% BrES2** *[integer]*
- **% BrUminSA2** *[real(8)]*
- **% BrUmaxSA2** *[real(8)]*
- **% BrUszSA2** *[integer]*
- **% BrdUSA2** *[real(8)]*
- **% BrdETXS2** *[real(8)]*
- **% BrdES2** *[real(8)]*
- **% BrUszSB2** *[integer]*
- **% BrUminSB2** *[real(8)]*
- **% BrUmaxSB2** *[real(8)]*
- **% BrdUSB2** *[real(8)]*
- **% BrEgminH** *[real(8)]* :: for LPM
- **% BrEe1H** *[real(8)]*
- **% BrLEe1H** *[real(8)]* :: log10(BrEe1H)
- **% BrneH** *[integer]*
- **% BrdU1H** *[real(8)]*
- **% BrdEH** *[real(8)]* :: log E step cnst.BrdEH= log10(cnst.BrEe2H/cnst.BrEe1H)/(cnst.BrneH-1) inverse of the above
- **% BrEe2H** *[real(8)]* :: max Ee where table is available
- **% BrU1H** *[real(8)]*
- **% BrU2H** *[real(8)]*
- **% Brnu1H** *[integer]* :: =(cnst.BrU2H-cnst.BrU1H+0.00001d0)/cnst.BrdU1H+1
- **% BrneH2** *[integer]* :: for 2D table E size
- **% BrdEH2** *[real(8)]* :: // E bin
- **% BrEe2H2** *[real(8)]* :: max E for 2D table
- **% BrU3H** *[real(8)]*
- **% BrU4H** *[real(8)]*
- **% Brnu2H** *[integer]*
- **% BrdVU2H** *[integer]* :: = cnst.Brnu2H-1
- **% BrdU2H** *[real(8)]* :: = (cnst.BrU4H - cnst.BrU3H)/cnst.BrdVU2H
- **% BrPow** *[real(8)]*

- **% PrEg1H** *[real(8)]* :: minimum Eg above which LPM works
- **% PrLEg1H** *[real(8)]* :: log10 of PrEg1H
- **% PrneH** *[integer]* :: number of Eg bins
- **% PrdU1H** *[real(8)]* :: du
- **% PrdEH** *[real(8)]* :: dE in log10(Eg)
- **% PrU1H** *[real(8)]* :: minimum u= 0
- **% PrU2H** *[real(8)]* :: maximum u= 1
- **% Prnu1H** *[integer]* :: numboer of u bins
- **% PrEg2H** *[real(8)]* :: max Eg where table is available. ———–muons nuclear interaction
- **% muNVmin** *[real(8)]* :: min of Eg(virtual)/Emu by muon nuc. int.
- **% muNdU** *[real(8)]* :: du for sampling table
- **% muNTXT** *[integer]* :: total xs, dEdx(v<vmin), dEdx(vall), tab size.
- **% muNEmin** *[real(8)]* :: above this, muon nuc. int. is treatable
- **% muNLEmin** *[real(8)]* :: log10 of muNEmin
- **% muNEmax** *[real(8)]* :: above this, use some scaling(sampling)
- **% muNEmax1** *[real(8)]* :: max E of 1D table
- **% muNdETX** *[real(8)]* :: log10 Energy step for total muon nuc. int prob.
- **% muNdE** *[real(8)]* :: log10 Energy step for sampling table
- **% muNUsize** *[integer]* :: sampling table size for u.
- **% muNEsize** *[integer]* :: sampling table size for log10 E
- **% muNpwtx** *[real(8)]* :: prob/X0 energy dependence; power. set after table for total prob. is read
- **% muNpwdEdx0** *[real(8)]* :: dEdx(v<vmin)/Emu enery dependence; power set after table is read
- **% muNpwdEdxt** *[real(8)]* :: dEdXt(v<vmax)/Emu energy dependence, power set after table is read
- **% muBrVmin** *[real(8)]* :: brems min of Eg/Emu. for muon Brems
- **% muBrdU** *[real(8)]* :: du for sampling table
- **% muBrTXT** *[integer]* :: total xs, dEdx(v<vmin), dEdx(vall), tab size.
- **% muBrEmin** *[real(8)]* :: above this, muon brems is treatable
- **% muBrLEmin** *[real(8)]* :: log10 of muBrEmin
- **% muBrEmax** *[real(8)]* :: above this, use some scaling
- **% muBrEmax1** *[real(8)]* :: max E of 1D table
- **% muBrdETX** *[real(8)]* :: log10 Energy step for total muon brems prob.
- **% muBrdE** *[real(8)]* :: log10 Energy step for sampling table

- **% muBrUsize** *[integer]* :: sampling table size for u.
- **% muBrEsize** *[integer]* :: sampling table size for log10 E dependence can be neglected
- **% muPrVmin** *[real(8)]* :: pair creation min of Eg(virtual)/Emu by muon pair cre.
- **% muPrdU** *[real(8)]* :: du for sampling table
- **% muPrTXT** *[integer]* :: total xs, dEdx(v<vmin), dEdx(vall), tab size.
- **% muPrEmin** *[real(8)]* :: above this, muon pair creation is treatable
- **% muPrLEmin** *[real(8)]* :: log10 of muPrEmin
- **% muPrEmax** *[real(8)]* :: above this, use some scaling
- **% muPrEmax1** *[real(8)]* :: max E of 1D table
- **% muPrdETX** *[real(8)]* :: log10 Energy step for total muon pair prob.
- **% muPrdE** *[real(8)]* :: log10 Energy step for sampling table
- **% muPrUsize** *[integer]* :: sampling table size for u.
- **% muPrEsize** *[integer]* :: sampling table size for log10 E dependence can be neglected
- **% how** *[integer]*
- **% NormS** *[real(8)]* :: normalization const
- **% NormPS** *[real(8)]* :: normalization const
- **% NormCS** *[real(8)]* :: normalization const
- **% NormSH** *[real(8)]* :: normalization const

Header ZbpSample.h

`type bpTbl`

Type fields

- **% BrTXL** (mxBrTXL,2) *[real(8)]*
- **% BrSTLA** (mxBrTblLA, 1) *[real(8)]*
- **% BrSTLB** (mxBrTblLB, 1) *[real(8)]*
- **% BrTXH** (mxBrTXH,2) *[real(8)]*
- **% BrSTHA** (mxBrTblHA, 1) *[real(8)]*
- **% BrSTHB** (mxBrTblHB, 1) *[real(8)]*
- **% PrTXL** (mxPrTXL) *[real(8)]*
- **% PrSTLA** (mxPrTblLA, 1) *[real(8)]*
- **% PrSTLB** (mxPrTblLB, 1) *[real(8)]*
- **% PrTXH** (mxPrTXH) *[real(8)]*
- **% PrSTH** (mxPrTblH, 1) *[real(8)]*

- **% BrTXS** (mxBrTXS,2) *[real(8)]*
- **% BrSTSA** (mxBrTblSA, 1) *[real(8)]*
- **% BrSTSB** (mxBrTblSB, 1) *[real(8)]*
- **% BrTXS2** (mxBrTXS2,2) *[real(8)]*
- **% BrSTSA2** (mxBrTblSA2, 1) *[real(8)]*
- **% BrSTSB2** (mxBrTblSB2, 1) *[real(8)]*
- **% MuNTX** (mxMuNTX) *[real(8)]*
- **% MuNdEdx0** (mxMuNTX) *[real(8)]*
- **% MuNdEdxt** (mxMuNTX) *[real(8)]*
- **% MuBrTX** (mxMuBrTX) *[real(8)]*
- **% MuBrdEdx0** (mxMuBrTX) *[real(8)]*
- **% MuBrdEdxt** (mxMuBrTX) *[real(8)]*
- **% MuPrTX** (mxMuPrTX) *[real(8)]*
- **% MuPrdEdx0** (mxMuPrTX) *[real(8)]*
- **% MuPrdEdxt** (mxMuPrTX) *[real(8)]*
- **% MuNTbl** (mxMuNTbl, 1) *[real(8)]*
- **% MuBrTbl** (mxMuBrTbl, 1) *[real(8)]*
- **% MuPrTbl** (mxMuPrTbl, 1) *[real(8)]*

Header ZbpTbl.h

`type site`

Type fields

- **% pos** *[position]*
- **% Txyz2det** (3,3) *[real(8)]* :: xyz to detector system transform mat
- **% Tdet2xyz** (3,3) *[real(8)]* :: inverse of above
- **% zpl** *[real(8)]* :: z value in 1ry system
- **% mu** *[real(8)]*
- **% minitime** *[real(8)]*

Header Zobsv.h

`type assite`

Type fields

- **% pos** *[position]*
- **% zpl** *[real(8)]*
- **% mu** *[real(8)]* :: Moliere Unit
- **% esize** *[real(8)]* :: electron size

- **`% age`** *[real(8)]* :: size weighted age

Header Zobsv.h

`type magfield`

Type fields

- **`% x`** *[real(8)]* :: in earth_center coordinate
- **`% y`** *[real(8)]*
- **`% z`** *[real(8)]*
- **`% sys`** *[character(4)]* :: which system. ‘xyz’, ‘ned’, ‘hva’

Header Zmagfield.h

D.2 User accessible subroutines

`subroutine cwriteParam`(*io*, *force*)

write parameters on the error output

Parameters

- **io** *[integer,in]* :: utput logical dev. #. ErrorOut –> stderr
- **force** *[integer,in]* :: if non zero, Hidden parameters are written. hidden ones are also written when Hidden=T

`subroutine cprintPrim`(*out*)

print primary information

Parameters **out** *[integer,in]* :: output logical device #

`subroutine cprintPrim`(*out*)

print observation information

Parameters **out** *[integer,in]* :: output logical device #

`subroutine ckf2cos`(*kf*, *code*, *subcode*, *chg*)

kf code to cosmos code.

Parameters

- **kf** *[integer,in]*
- **code** *[integer,out]*
- **subcode** *[integer,out]*
- **chg** *[integer,out]*

`subroutine ccos2kf`(*code*, *subcode*, *chg*, *kf*)

cosmos code to kf code;

Parameters

- **code** *[integer,in]*
- **subcode** *[integer,in]*

- **chg** *[integer,in]*
- **kf** *[integer,out]*

`subroutine epResetEcrit`(*io*, *name*, *newV*, *oldV*, *icon*)

reset Crittical energy of a given media with “name”

Parameters

- **io** *[integer,in]* :: output message device #

0	some message is put as standard Fortran error message
6	some message is put as sysout.
>0	assume logical device is open with that number
<0	no message is put, but see next

- **name** *[character(*),in]* :: media name such as “Air” if media with “name” is not found eroor message is
- **newV** *[real(8),in]* :: new critical energy (GeV) new value is set to media%Ecrit
- **oldV** *[real(8),out]* :: E crit so far defined. (GeV)
- **icon** *[integer,out]* :: 0 if ok. -1 if some error

`subroutine modCodeConv/ccos2pdg`(*aPtcl*, *pdgcode*)

convert from Cosmos particle structure to PDG code

Parameters

- **aPtcl** *[ptcl,in]* :: Cosmos particle structure
- **pdgcode** *[integer,out]* :: PDG particle code

`subroutine cavedEdx`(*eno*, *age*, *dedx*)

Average dedx (2.2×10^{-2} for the test)

Parameters

- **eno** *[real(8),in]* :: electon size
- **age** *[real(8),in]* :: age of the shower
- **dedx** *[real(8),out]* :: average dedx as defined in GeV/(kg/m2)

`subroutine cgetNmu`(*eth*, *nmu*)

compute muon numbers this is not yet made. tentatively nmu = 0 is given.

Parameters

- **eth** *[real(8),in]* :: Threshold energy of muons. (GeV)
- **nmu** (*NoOfASSites*) *[real(8),out]* :: output. number of muons E>eth

`subroutine cxyz2det`(*ly*, *a*, *b*)

convert coord value in the “xyz” system into “det” system.

Parameters

- **ly** *[integer(2),in]* :: layer # of the observation depth. Its origin is used to convert particle coordinate ('a' in E-xyz) into the detector coordinate ('b'). Detector origin is the crossing point of 1ry direction and spherical surface at a given depth (height). Z-axis is vertical, X-axis is XaxisFromSouth (~90 deg normally magnetic East at the BaseL. The x-y plane is tangential to the spherical surface at the origin. ly is normally `MovedTrack%where` (integer(2))
- **a** *[coord,in]* :: coord in 'xyz'
- **b** *[coord,out]* :: coord in 'det'

subroutine cdet2xyz(*ly*, *a*, *b*)

convert coord value in the "det" system into "det" system. See the parameter description of `cxyz2det()`

subroutine cxyz2detD(*ly*, *a*, *b*)

convert coord value in the "xyz" system into "det" system for Direction cos. See the parameter description of `cxyz2det()`

subroutine cdet2xyzD(*ly*, *a*, *b*)

convert coord value in the "det" system into "det" system for Direction cos. See the parameter description of `cxyz2det()`

subroutine cllh2eCent(*llh*, *xyz*)

convert llh coordinates to E-xyz

Parameters

- **llh** *[coord,in]* :: containing data in latitude, longitude, height.
- **xyz** *[coord,out]* :: The coordinate system is such that the origin is at the center of the earth

x-axis	directed to (0, 0) latitude and longitude.
y-axis	directed to (0, 90) latitude and longitude.
z-axis	directed to the north pole.

xyz.r(1)	x coordinate value in m
xyz.r(2)	y
xyz.r(3)	z

note: xyz can be the same as llh. time component is unchanged

subroutine csetPos(*location*)

set position information for a given xyz

Parameters **location** *[position,inout]* :: coord part of location is input.

subroutine cgeomag(*yearin*, *llh*, *h*, *icon*)

Parameters

- **yearin** *[real(8),in]* :: such as 1990.5
- **llh** *[coord,in]* :: position around the earth. in 'llh' form is better. if not 'llh' conversion is done here.

- **h** *[magfield,out]* :: magnetic field is set in the form of 'ned' (north, east-down). The unit is T.
- **icon** *[integer,out]* ::

0	o.k
1	too heigh location. result could be suspicious.
2	input parameter wrong....

subroutine ctransMagTo(*sys*, *pos*, *a*, *b*)

transform magnetic field components in one coordinate sytem to another.

a.sys \ sys	'xyz'	'hva'	'ned'
'xyz'	o	o	o
'hva'	o	o	o
'ned'	o	o	o

Parameters

- **sys** *[character*(*),in]* :: 'xyz', 'hva' or 'ned'. the target coordinate system where magnetic filed is represented.
- **pos** *[coord,in]* :: position where mag is given
- **a** *[magfield,in]*
- **b** *[magfield,out]* :: transformed component, b.sys=sys

subroutine cqEventNo(*num*, *cumnum*)

inquire the event number

Parameters

- **num** *[integer,out]* :: number of events in the current run
- **cumnum** *[integer,out]* :: cummulative number of events so far. may be used after the initialization of an event, then this gives the number for that event. It will differ from `num` if `Cont=t` is used.

subroutine cqIniRn(*ir*)

inquire the initial random seed

Parameters **ir** (2) *[integer,out]* :: the initial seed of the random number generator for the event.

subroutine cqIncident(*aTrack*, *angleAtObs*)

inquire incident particle

Parameters

- **aTrack** *[track,out]* :: incident particle track information
- **angleAtObs** *[coord,out]* :: to get direction cosined of the incident particle in the "detector system" of the deepest observation level. `angleAtObs%r(1)`, `angleAtObs%r(2)` and `angleAtObs%r(3)` are the three components.

subroutine ciniTracking(*incident*)

inquire first int point info.

Parameters **incident** *[track,in]*

subroutine cqFirstColMedia(*A*, *Z*, *xs*)

retrns first col. element and Xsecion on it

Parameters

- **A** *[integer,out]*
- **Z** *[integer,out]*
- **xs** *[real(8),out]* :: Xsection

subroutine cqIncident(*incident*, *AngleAtObs*)

inquire incident particle

Parameters

- **incident** *[track,out]*
- **AngleAtObs** *[coord,out]*

subroutine csetMagField(*sys*, *b1*, *b2*, *b3*, *b*)

set Calculated magnetic field to /magfield/ b

Parameters

- **sys** *[character(3),in]* :: which system. ‘xyz’, ‘hva’, ‘ned’ etc
- **b1** *[real(8)]* :: 3 components of mag. in the system ‘sys’
- **b** *[magfield,out]* :: /magfield/

function cvh2temp(*vh*)

temperature in Kelvin of ‘site’

Parameters **vh** *[real(8),in]* :: height in meter

Return **cv2temp** *[real(8)]* :: temperature in Kelvin

function cvh2den(*vh*)

density of air

Parameters **vh** *[real(8),in]* :: height in meter

Return **cvh2den** *[real(8)]* :: density in kg/m3

function klena(*cha*)

actual length of character string. note: if no character dec. is given for string, klena=0 will result. e.g. a=’abc’, klena(a). but klena(‘abc’) is ok.

Parameters **cha** *[character*(*),in]* :: character string

Return **klena** *[integer]* :: length of the string

D.3 User accessible variables

`NoOfSites` *[integer]*

No of particle observation sites (*Zobsv.h*)

`NoOfASSites` *[integer]*

maximum index for observation depths. (*Zobsv.h*)

`CompASNe` (i) *[real(8)]*

component A.S size produced by the input electron. For depths where this value is 0, avoid doing something here. (i=1, NoOfASSites; index for observation depths) (*Zobsv.h*)

`CompASAge` (i) *[real(8)]*

age of component A.S produced by the input electron. If this value is 2.0, the A.S is assumed to be very old and the CompASNe(i) is 0. You should skip treating deeper depths. (i=1, *`NoOfASSites`* ; index for observation depths) (*Zobsv.h*)

`ObsSites` (i) *[site]*

i=1, NoOfSites (*Zobsv.h*)

`ASObsSites` (i) *[assite]*

i=1, *`NoOfASSites`* (*Zobsv.h*)

`Media` (i) *[epmedia]*

Media information of *Media_no=i* (*Zmedia.h*)

`MediaNo` *[integer]*

current Media number. (*ZmediaLoft.h*)

`TrackBefMove` *[track]*

track before moved (*Ztrackv.h*)

`MovedTrack` *[track]*

contain track moved (*Ztrackv.h*)

APPENDIX E

PHYSICS

E.1 Particle Identification code

Cosmos uses a conventional particle code that differs completely from extensive one recommended in the Particle Data book. Subroutines to convert the Cosmos code to the PDG code are available and described in Sec.??. A particle is identified by the particle code, subcode and charge. When you need to identify a particle in the user hook routines, you may use the `#include "Zcode.h"` directive and refer the code names in that file rather than code numbers.

The following list is the names that represent the particles in Cosmos. They are roughly in the order of mass. The code for a heavy nucleus such as deuteron, alpha, … is not available when you judge particle type in air shower. It can be used to specify the primary particle type only. To judge a particle type of a nucleus in air shower, you may use `kgnuc` for particle code, and if it matches, you can identify the nucleus by testing the subcode and charge; the subcode expresses the mass number (A). To specify a primary you can also avoid using the naming below but use `'iso 3 2'`, for example, to express ^{3}He.

Table 5.1: particle code

particle	code name	code number	particle	code name	code number
photon	kphoton	1	electron	kelec	2
muon	kmuon	3	pion	kpion	4
kaon	kkaon	5	nucleon	knuc	6
ν_e	kneue	7	ν_μ	kneumu	8
nucleus	kgnuc	9	triton	ktriton	17
He	kalfa	10	LiBeB(A~8)	klibe	11
CNO(A~14)	kcno	12	H(A~25)	khvy	13
VH(A~35)	kvhvy	14	Fe(A~56)	kiron	15
D meson	kdmes	16	ρ	krho	25
Λ	klambda	18	Λ_c	klambdac	21
Σ	ksigma	19	Ξ	kgzai	20
ω	komega	26	ϕ	kphi	27
η	keta	28	deuteron	kdeuteron	29

The subcode is used to discriminate the particle from anti-particle, if the difference is essential such as the neutron and neutrino, but not used for, say, anti-protons because the charge can tell it. For K^0 mesons, the subcode is used to distinguish between ${K^0}_S$ and ${K^0}_L$. Cosmos does not assign the particle and anti-particle code to them. They are assumed to be produced in equal weight and the actual assignment is performed randomly when they interact. To identify the particle and anti-particle, you can use the subcode name, `regptcl` and `antip`. For ${K^0}_S$ and ${K^0}_L$ the subcode name, `k0l` and `k0l` may be used.

The “KF” code used in Particle Data Book can be converted to the Cosmos code by calling *ckf2cos()* as:

```
call ckf2cos(kf, code, subcode, chg)
```

where kf is an input integer kf code, and others are the output for Cosmos code.

The inverse conversion is possible by:

```
call ccos2kf(code, subcode, chg, kf)
```

The following fragment of a program code will tell you how to use these code and sub-code system.

```
!...
#include "Zcode.h"
!...
      record /track/ aTrack
!...
      if(aTrack.p.code .eq. knuc .and. aTrack.p.charge .eq. 0) then
!            neutron; judge if anti neutron or not.
          if(aTrack.p.subcode .eq. antip) then
!               this is anti neutron
          else if(aTrack.p.subcode .eq. regptcl) then
!               this is neutron
          else
!               error assignment
          endif
!...
```

E.2 Physics list and references

E.3 Hadronic interaction models

[Hadronic interaction models are read in *LibLoft/Had/Interface/cintModels.f*.]

[Available models are defined in *LibLoft/Header/BlockData/cblkEvhnp.h*.]

Available models are listed in Table 5.2.

Table 5.2: Hadronic interaction models

Model	Name in param	Energy range	Comment
PHITS	phits		
JAM	jam		
DPMJET3	dpmjet3		
Fritiof 7.02			
Fritiof 1.6	fritiof1.6		
GHEISHA	gheisha		
Nucrin	nucrin		
Ad hoc	ad-hoc		
IncDPM3	incdpm3		
Special	special		
QGSJET II-03	qgsjet2		See Section 4.1
QGSJET II-04	qgsjet2		See Section 4.1
EPOS 1.99	epos		See Section 4.1
EPOS LHC v3700	epos		See Section 4.1
Sibyll 2.1	sibyll		See Section 4.1
Sibyll 2.3c	sibyll		See Section 4.1

APPENDIX

F

PLATFORMS

F.1 Platforms tested

BIBLIOGRAPHY

[cosmosLegacy] The Users Manual of Cosmos, Cosmos group, http://cosmos.icrr.utokyo.ac.jp/cosmosHome/index.html

INDEX