CosmosX manual

Release 0.9.0

COSMOS X development team

Feb 13, 2025

CONTENTS:

1 What is COSMOS X? 1
- 1.1 What can we do with COSMOS X? . . . 1
- 1.2 Structure . . . 1

2 How to use COSMOS X for the first time? 7
- 2.1 Environment . . . 7
- 2.2 Download . . . 7
- 2.3 Installation . . . 7
- 2.4 Test program (First Kiss) . . . 8

3 How to edit the user control files? 11
- 3.1 primary file . . . 11
- 3.2 param file . . . 12
- 3.3 userhoook subroutines . . . 13

4 How to optimize my simulation? 15
- 4.1 General . . . 15
- 4.2 Hadronic interaction model . . . 17
- 4.3 Thinning . . . 18
- 4.4 AS, hybrid method . . . 18
- 4.5 Magnetic field . . . 18
- 4.6 Electric field . . . 19
- 4.7 Non-air material, non-earth sphere . . . 20
- 4.8 User defined Trace output . . . 20

5 How to use COSMOS X EPICS? 21

6 Examples 23
- 6.1 FirstKiss . . . 23
- 6.2 PrimaryHowTo . . . 23
- 6.3 MyField . . . 24
- 6.4 Upgoing . . . 25
- 6.5 GencolLike . . . 25
- 6.6 Sun . . . 27
- 6.7 HybASForUHEG-N-S-Effect . . . 29

7 References 43

A Unit of physical quantities 45

B Coordinate system **47**
B.1 The E-xyz system . . . 47
B.2 The detector system and the primary system . . . 47

C Parameter list **49**
C.1 Parameter list of *&PARAM* in *param* file . . . 49
C.2 Parameter list of *&HPARAM* in *param* file . . . 51

D Subroutines and variables **53**
D.1 User accessible types . . . 53
D.2 User accessible subroutines and functions . . . 65
D.3 User accessible variables . . . 71

E Physics **73**
E.1 Particle Identification code . . . 73
E.2 Physics list and references . . . 74
E.3 Hadronic interaction models . . . 74

F Platforms **77**
F.1 Platforms tested . . . 77

Bibliography **79**

Index **81**

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.
- **Scrpt/** 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
```

(continues on next page)

(continued from previous page)

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

(continues on next page)

(continued from previous page)

```
      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.
            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
```

(continues on next page)

(continued from previous page)

```
      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.

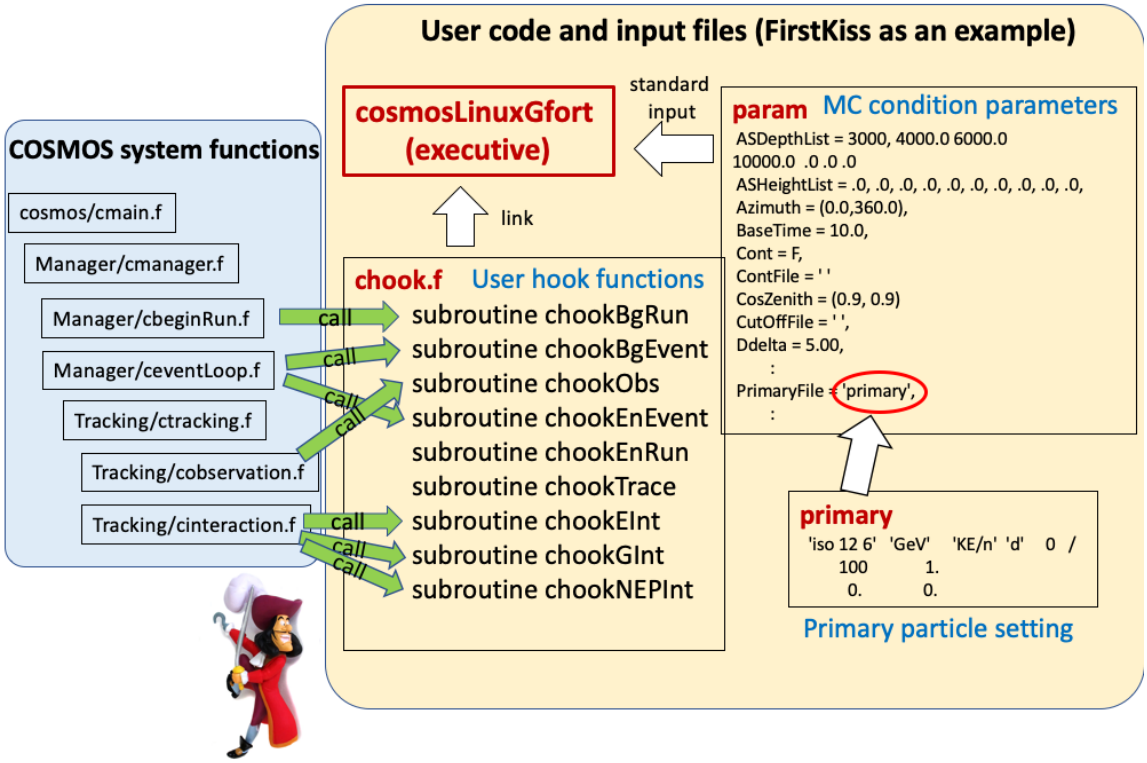

Fig. 1.1: Relation between the COSMOS system functions, userhook functions and parameter files.

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,

https://www.icrr.u-tokyo.ac.jp/cosmos/

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
        - `source Scrpt/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*.
        - `./Scrpt/CompileLibraryByCMake.sh` to make library

[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
```

  - 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 Scrpt/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*.
  - `./Scrpt/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

```
./Scrpt/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 4.8.

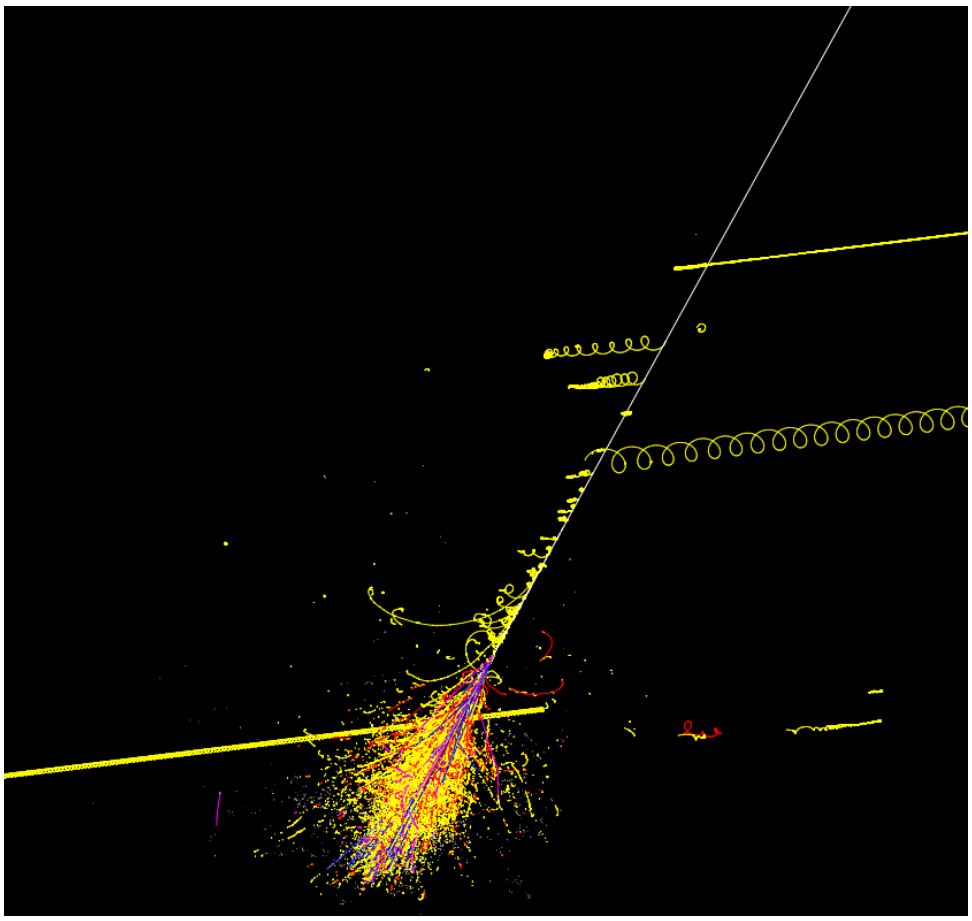

Fig. 2.1: ROOT visualization of a 100 GeV/n nitrogen shower in the *FirstKiss* sample.

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`.

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 $\mathrm{flux} \times E^X$ 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

- `./Scrpt/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.

subroutine chookBgRun()
This subroutine is called when the program starts.

subroutine chookBgEvent()
This subroutine is called every time a new injection happens. To record the information of individual primary particle, edit this routine.

subroutine chookObs(*aTrack*, *id***)**
This subroutine is called every time individual particle passes the observation levels defined by `DepthList` or `HeightList`.

subroutine 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.

subroutine chookEnRun()
This subroutine is called when all shower trackings are completed.

subroutine 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.

subroutine chookEInt(*never***)**
This subroutine is called when an electron (or positron) interaction occurs.

Parameters
never *[integer,out]* :: if `never` is set non zero, this routine will never be called.

subroutine chookGInt(*never***)**
This subroutine is called when a gamma-ray interaction occurs.

subroutine chookNEPInt(*never***)**
This subroutine is called when a non-electromagnetic interaction occurs.

CHAPTER

FOUR

HOW TO OPTIMIZE MY SIMULATION?

4.1 General

- Normally, the user may handle *chook.f* in the user's application area to get a desired output from Cosmos.
- However, in some case, the user may want to introduce non default effects (say, an electric field dependent on the particle position), which cannot be handled by using the input "namelist" data.
- For such a purpose, several files are prepared in `$LIBLOFT/Header`. They are summarized in Table 4.1

Table 4.1: cmy-type program files

file	used to specify or add	called when
cmyBfield.f	non-default magnetic field	ObjFile is non blank
cmyEfield.f	complex electric field	`HowEfield` ==2
cmyCerenkov.f	treatment of Cerenkov light	`Trace` > 160
cmyTracePlus.f	to show the shower spread scale	the user puts `call cmyTracePlus``in ``chookBgEvent`
cthinning.f	non default thinning	`ThinSampling = T`
chookHybAs.f	non default analytical AS generation	Generate contains *as*
chookEabsorb.f	non default dead particles treatment	`Eabsorb` ! = 0

- Their handling scheme is summarized here. We assume environmental variables *$LIBLOFT* and *$COSMOSTOP* have been set. In what follows, when "Cosmos" is used to show a file path, it means *$COSMOSTOP* (so, *Cosmos/cosmos* means *$COSMOSTOP/cosmos*; Similarly, *LibLoft* is used as *$LIBLOFT*.)
- As mentioned abvoe, *chook.f* is used to handle the program run. It contains `#include "cmain.f"`, which is located in *Cosmos/cosmos*. The *cmain.f* contains lines:

```
#include "cmyBfield.f"
#include "cmyBfield.f"
#include "chookHybAS.f"
#include "chookEabsorb.f"
#include "cthinning.f"
#include "cmyCerenkov.f"
#include "cmyTracePlus.f"
```

- "cpp" (C pre-processor) tries to find af file (xxx) specified by a `#include "xxx"`, as follows. Suppose the file which contains this include statement is in a folder A. + Then, "xxx" is first sought for in the folder A (the definition of the "current folder" for "cpp"). + If not found, the folders specified by `$INCHEADER` are examined. + `$INCHEADER` is defined in "site.config" as `-I./ -I$COSMOSINC -I$LIBLOFTINC`, where `$COSMOSINC` is defined as `$COSMOSTOP/cosmos` and `$LIBLOFTINC` as `$LIBLOFT/Header`[1] . + So the current application area is first examined. If not found, *Cosmos/cosmos*, and if not, lastly *LibLoft/Header*. + Of course, if the last search fails, the compilation stops with an "error message".
- Sppose the following cases + In the user's application area, there is no "special files" such as *cmyBfield.f* etc (default status). In this case, the special files will be obtained from *LibLoft/Header*. + One or more of the special files are copied to the user's application area but the user neither modify them nor touch *chook.f* and/or namelist input so that the special files are affected by that. In this case, the ones in the application area will be used but there will be no change from the previous case. + If some of copied "special file" are modified and *chook.f* and namelist input are also adjusted to be consistent with the modifications, the non-default behavior of the program run will be realized. + The user should not insert `#include "cmyBfield.f"` etc in *chook.f* nor add *.o* files to "objs" (say `objs = chook.o` in *chook.mk*.

4.1.1 Details of each special file case

cmyBfield.f

- The default magnetic field is the Geomagnetic field for the Earth. It is the IGRF version. The year is fixed by "YearOfGeomag" in the namelist.
- To give a different magnetic field for the Earth or non-Earth environment, *cmyBfield.f* must be copied to the user's application area.One example is in *cosmosx/Application/Example/Sun*. We employ a model constructed by Hakamada. The code is put in *LibLoft/Env/Sun/Import/SunMag_PFSS_by_Hakamada/*

In the program, the folllowing is mentioned.

References for PFSS model:

[1] K. Hakamada, M. Kojima, T. Ohmi, M. Tokumaru, and K.Fujiki, Sol. Phys. 227, 387 (2005).

[2] K. Hakamada, Sol. Phys. 159, 89 (1995).

Not sure the above is relevant or not.

To do a Cosmos job with the magnetic field of the Sun[2] .

cmyBfield.f:

Copy *LibLoft/Header/cmyBfield.f* to the *Applications/Example/Sun/* (done). Modify the program to include call `calcmagFE(...)` which is an interface routine (in C-language) to the Hakamada's routines (done).The interface routines are placed in *LibLoft/Env/Sun/Import/SunMag_PFSS_by_Hakamada/Interface/*

SunDef.d:

Make this data file in which `&ObjParam` namelist data to describe the Sun is given (done). The name, *SunDef.d*, may be arbitrary but the namelist name, `&ObjParam` should be kept.

[1] Instead of `-I./`, `-./` was wrongly used in some old site.config

[2] In what follows, "done" means, the described action has been done already

ObjFile:
The file name, *SunDef.d* in this case, must be spcified by a variable, `ObjFile` in `&Hparam` (done).

SunAtmos.d, AtmosFile, AtmosModel:
To define the sun's atmosphere, the user must create a file (in this example, *SunAtmos.d*) in which a 3-column table of "height(m) temperature(K) and density(kg/m3) exist (done)). The name must be given in the `AtmosFile` variable in `&Hparam` where `AtmosModel = 3` (see the next item) must also be given (done).

atmosModel.sh:
The description is for building with legacy make.

The user can verify the current atmosphere model by calling a script, "atmosModel.sh". The script is stored in *Cosmos/Scrpt/*. (So it's better to put *Cosmos/Scrpt* in the `PATH` envrionmental variable). Invoking the command shows the current atmosphere model (1-standard for the Earth, 2-time and location dependent NRL atmosphere for the Earth, 3-special one like for the sun.) If needed, the user can change the model, and the library will be renewed. Note that this action only changes the library so the user must recompile own application.

Also see Section 4.5 .

cmyEfield.f

See Section 4.6 .

4.2 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.

Table 4.2: Available versions of hadronic interaction models

Model	Name in param	Environmental var.	Env. Value	Comments
QGSJET II-03	qgsjet2	QGS	qgsjetII-03	
QGSJET II-04	qgsjet2	QGS	qgsjetII-04	default of QGSJET
QGSJET III	qgsjet2	QGS	qgsjetIII	
SIBYLL 2.1	sibyll	SIBYLL	sibyll2.1	
SIBYLL 2.3c	sibyll	SIBYLL	sibyll2.3c	
SIBYLL 2.3d	sibyll	SIBYLL	sibyll2.3d	
EPOS LHC v3700	epos	EPOS	epos-lhc-v3700	
EPOS LHC v3400	epos	EPOS	epos-lhc-v3400	
EPOS v1.99	epos	EPOS	epos199	

The list of available models and the versions of each model are summarized in Table 5.2 . For the models,

QGSJET, SIBYLL, and EPOS, you can select a version. When you build the library, set the corresponding environmental variable in the 3rd column in Table 4.2 to the value in the 4th column. When you want to use the QGSJET-III model for example, build the library with the following command:

```
env QGS=qgsjetIII ./Script/CompileLibraryByCMake.sh
```

Build an executable and run with parameter like

```
IntModel = '"phits" 2.0 "dpmjet3" 80 "qgsjet2"'
```

QGSJET-III will be used for interactions above 80 GeV.

If you use legacy make, move to the directory *$COSMOSTOP/LibLoft/Scrpt*, and type:

```
intModel.sh
```

You can find choices 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 list shown.

4.3 Thinning

4.4 AS, hybrid method

4.5 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 4.3: 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.

4.6 Electric field

4.6.1 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 4.6.3.

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.

4.6.2 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.

4.6.3 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 `$LIBLOFT/Header/cmyEfield.f` to your project directory. The local file, `./cmyEfield.f` will be automatically included in the build process.
2. Edit *cmyEfield.f* to define customized electric field, `cmyEfield(aTrack, Efout)`.
3. 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. (see Section 6.3)

4.7 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.

4.8 User defined Trace output

CHAPTER

FIVE

HOW TO USE COSMOS X EPICS?

CHAPTER

SIX

EXAMPLES

6.1 FirstKiss

6.2 PrimaryHowTo

This sample example explains how primary particle composition and flux are defined in the *'primary'* file. In this sample COSMOS simulation ONLY outputs primary information at the beginning of each event (primary injection) and exits without any air shower production. Output file can be converted into the spectra by a ROOT script so that we can confirm if the spectra are as expected.

Users edit following three files:

primary param chook.f

Detail of these files are explained below.

6.2.1 Usage

```
make clean; make
./cosmosMacGfort < param > out 2> err # (here 'MacGfort' depends on your␣
↪system.)
root primary_analysis.cpp # (assuming you can use ROOT.)
```

6.2.2 primary file

primary file in this directory defines the proton, Helium and CNO fluxes. Mixed composition is simulated at the same time.

primary file is specified in the *param* file as

PrimaryFile = 'primary',

You can use different file name if necessary.

6.2.3 param file

Simulation condition is specified in the *param* file. In this specific sample, the *param* file describes as below.

DepthList = -1.0, HeightList = 99e3, HeightOfInj = 100e3,

Particles are injected from 100km (100e3 meters) above the sea level and the obsevation site is defined at 99km. So there is essentially no interaction. Usually DepthList is used to define the obsevation height

in kg/m^2 and HeightList in meter is calculated inside COSMOS. However, if it is defined in negative value like this sampe, HeightList is referred and DepthList is calculated.

DestEventNo = 100000

100k primary particles are injected. Because we do not simulate air shower, it is done in O(10sec).

6.2.4 chook.f

chook.f contained user subroutines where the users can define actions at each process of simulation. In this sample, except `chookBgEvent` subroutine, nothing is done. You can confirm the subroutines are empty.

The subroutine `chookBgEvent` is called when a new injection particle is defined. In this sample in `chookBgEvent`, the information (particle code etc) are printed to the standard output. Immediately after that, all information in the particle stack is cleaered using

```
call cinitStack
```

so that simulation of this event terminates. (It does not track down to 99km as explained above.)

6.3 MyField

This example shows how to set electric/magnetic field. Electric field is controlled by the parameter, HowEfield in `&HPARAM`, which is 0 (disabled) by default. Note that dielectric breakdown is not taken into consideration even if very strong electric field is defined. Magnetic field is controlled by the parameter, HowGeomag in `&HPARAM`, which is 11 (constant geomagnetic field calculated at `BaseL`) by default.

Users might edit the following files;

primary param chook.f cmyEfield.f cmyBfield.f CMakeLists.txt/chook.mk

6.3.1 Usage

At the top of the CosmosX source tree,

```
Scrpt/CompileExampleByCMake.sh Application/Example/MyField
cd Application/Example/MyField
./cosmosLinuxGfort < param out 2> err
 # here 'LinuxGfort' might be different depending on your system
 # For 'param' file, see below.
 # trace1 will be generated.
```

ReadTraceMacro.C can be used in the directory to show the trace file for *param-e1* and *param-e2*. For *param-b1*, the macro in the *FirstKiss* example will be used.

6.3.2 param file

3 different param files are given. For *param-e1* and *param-e2*, *primary_e* (50GeV electron) will be read as primary and for *param-b1*, *param_n* (100GeV/n Nitrogen) will be read.

- *param-e1* `HowEfield=1` is set. User can define upto 5 layers with constant electric fields by parameters `myEf(i)%H1`, `myEf(i)%H2`, and `myEf(i)%Ef` in `&HPARAM`.
- *param-e2* `HowEfield=2` is set. User defined subroutine, `cmyEfield`, will be called to calculate the electric field at any place.

- *param-b1* `HowGeomag=22` is set. In this example, the magnetic field exists at any place and the field is the constant defined by the parameters (`MagN`, `MagE`, `MagD`) in `&HPARAM`. (`MagN`, `MagE`, ``MagD`) are North, East and Down components of the magnetic field in the unit of Tesla in the detector system at BaseL.

6.3.3 chook.f

This file is not specific to this example (electric/magnetic field). To show the area where the electric field is not zero, subroutine cmyTracePlus is called to show several layers of square in the trace file.

6.3.4 cmyEfield.f

In this example, artifitial electric field along z-axis in the detector system is defined at x,y>0. The z component of the field strength is calculated by:

Efz= 360d3*sin(3.1415/2.*h/2d3)*exp(-h/5.d3) [V/m]

where *h* is height [m]. In the *param-e* file, the primary will be injected with zenith angle >0. The particle enters into the region where the electric field is finite in the way of shower development and generate additional cascade by the field.

6.4 Upgoing

Upgoing primary (e.g, tau lepton generated UHE nue inside earth)

We assume tau- or tau+ lepton which goes up from the surface area of the Earth. For tau > 10^15 eV is interesting but we here consider ~10TeV tau for quick calculation.

If `Trace` > 0 is given, the `cmyTracePlus` routine is called inside `chookBgEvent` in *chook.f*; it wrties a fake trace data at every observation depth. The trace data is by a virtual heavy ion of negative charge and drwan with a white square line; its center is the 1ry axis position at the observation depth. The default size is 6km x 6km. To omit the square, comment out the `call cmyTracePlus` in *chook.f*. To modify the number of squares or to chage the shape, etc, modified *cmyTracePlus.f*. (Need not to add `#include` or similar action.) It is included by `#include` in the *cmain.f* in Cosmos/cosmos; cmyTracePlus.f in the prsent directory has priority. (See last few slides in *FigA.pdf* and the comment in *cmhyTracePlus.f*).

6.5 GencolLike

The stuff here is to get information of multi-particle production by high energy hadron inelastic collisions; (the information is the one contained in "ptcl"; particle kind, momentum etc)

Similar information can be obtained by using *Cosmos/Util/Gencol*, but there is some difference:

1) we can use a compound target such as PWO, Air (however note for dpmjet3) etc.

2) the program gives only produced particle information (/ptcl/ info.) so we cannot know diffraction information. (Probably, possible to know it, if *getDiffCode.f* and *Zintmodel.h* are imported from Gencol).

One strange and interesting case is that Gencol cannot be used for K0l/K0s incident with the dpmjet3 interaction model; in that case, we get a message,"pdf is not ready for this projectile" (pdf: parton distribution function).

But using this, we can get result for that case (results seems to be OK; comparison with results by other models indicating so).

1) In param, you must set

a) For $PARAM

```
Freec  = f
IntModel = '"epos"'     ! as you like
LatitOfSite = -90.,
LongitOfSite = 0.,
CosZenith = (1.0, 1.0).
HeightOfInj = 5100.  or   4500.
DestEventNo = 500000000,  500000000     ! large number
```

b) For $HPARAM

```
UserHooki= 1,  100   ! output type &  actual # of events to be generated
           ready made  output type is one of 0,1,2: see later
DpmFile ="dpmjet.inp"  ! if  target is air, comment out this
                       ! if  target is non air and model is dpmjet3
                       ! use this; see later for dpmjet.inp.
AtmosFile="Air+solid.d" ! this is "atmosphere" with Air + Solid medium
```

```
--------
The target can be Air or another simple  nucleus (Fe, Pb...) or compound
nuclei (PWO etc).
For that  we
use  Air+Solid.d   as the "Atmosphere":
#-------------------------------------------------------------------
      4000    260.            0.80     Pb
      5000    250.            0.50     Air
```

To specify that the target is air, we must give injection height h >= 5000. To Specify Pb as the taget , 4000<=h <5000. If you replace Pb by PWO the target will be PWO. The actual nucleus is randomly seleccted with the weight of cross-section and number density of the nucleus.

NOTE: In the case of `IntModel='"dpmjet3"'`, to use , e.g, Pb target, the user must activate dpmjet.inp by `DpmFile ="dpmjet.inp"` and adjust the content of *dpmjet.inp* by giving the following 3:

PROJPAR TARPAR ENERGY

Two energy terms should be larger than the one in primary file. (1st < 2nd one is better).

However, PWO like compound case cannot be managed by this way. We must make *dpmjet.inp* (diff. from the we have been treating) and *dpmjet.GLB* files. How to make such *dpmjet.inp* and *dpmjet.GLB* will be given somewhere in *LibLoft/Util/*

2) Incident particle may be set by primary file.

3) Output type by UserHooki. The first value of UserHooki fix the output type. See subroutine chookNEPInt in *chook.f*

6.6 Sun

6.6.1 Purpose of this Example

The aim is to perform Monte Carlo simulations in a non-Earth environment, using the Sun as an example.

- First, basic data representing the Sun should be prepared. The variables should be named the same as for Earth, but with values appropriate for the Sun. The data should be stored in a file named *SunDef.d*, although the name can be chosen freely. The file with the red text name should be located in the Sun folder.
- The data should be recorded in a namelist named `&ObjParam` in the file mentioned above (the namelist name `ObjParam` is fixed). The values for Earth can be used as a reference and should be mentioned in the comments. Most of the data does not need to be extremely precise.
- The structure of the solar atmosphere is included in *SunAtmos.d*, which can also be given a freely chosen name. As the composition differs from the Earth's atmosphere, the sunAir defined as `$LIBLOFT/Data/Media/sunAir` should be used[1] .
- specify the data as follows:

  vnet Copy code

```
#h(m)          temp(K)       density(kg/m3)
#-----------------------------------------------------
-100e3       9396          .000415 sunAir
   0         6967          .000304
   150e3       5560          .000147
   ...
```

  For the first row, use the above format. If the composition changes at a certain altitude, use a different file name for that altitude.

- AtmosModel and AtmosFile

  In the namelist data for the `&Hparam` parameter file, include the following:

```
AtmosModel=3
AtmosFile=SunAtmos.d
```

- Rebuild the libary by the following command (cmake)

```
$ env ATMOSMODEL="Non-Standard" ./Scrpt/CompileLibraryByCMake.sh
```

  (Or run the *atmosModel.sh* command (legacy make))

  Currently, the library uses the standard Earth atmosphere as the default option, but you can use the NRL Earth atmosphere, which depends on time and location, or a special atmosphere like the solar atmosphere. You can check which option is being used by running the *atmosModel.sh* command. If you want to change the atmosphere model, also use this command.

- Coordinate systems

  or the Sun, use the same coordinate system as for the Earth. Consider the Sun as a sphere and take the center as the origin of the E-xyz system. The x, y, and z axes can be determined appropriately

[1] If a suitable substance is not available in the Media folder, refer to Section *NewMedia*. The figure for the data used can be found on pages 1 and 2 of *sunAtmos.pdf*.

depending on the problem. Longitude and latitude are also determined accordingly. The detector and primary systems can be determined in the same way.

- When you create and run the program, a trace file is generated. You may expect to see an image like a cascade in the Earth's atmosphere, but the solar atmosphere is so rarefied that there is little cascade development. You will likely see only a small number of particles spiraling in the magnetic field. Only about 10 to 20 out of hundreds of events will produce visually appealing images.
- If you output trace data in the E-xyz system, you must output it in double precision; otherwise, the accuracy will be insufficient to produce correct drawings. If the drawing system is performing calculations in single precision, even with double precision output, the same will hold true (for example, in the Geomview case, the two .png files starting with Exyz in the Fig folder). In other cases, verb|Trace=1| is used with the primary system.

If BaseM doesn't exist, proceed to create it and the following steps.

- **BaseM:** It would be good to create new data with a new name by referring to Air in LibLoft/Data/BaseM. You don't need to create it in BaseM at this stage, but eventually, it should be put in BaseM.

  The value 29.0 in the middle is the molar mass – molecular weight (in /Mol or g/Mol units) for gases, so make sure to enter it.

  Decompose the molecule into atoms, enter A and Z, and enter the relative abundance N. The ratio only needs to be relative.

  The value at the bottom can be found by referring to the PDG's web page in the following order:

```
PDG Web Page
Reviews, Tables & Plots
Atomic and Nuclear Properties of Materials (rev.) Interactive version
Select Interactive Version
Click on Inorganic compounds (Al through Fe) under the Periodic Table
...
Click on Mixtures
...
Find the relevant substance.
If there is one, select TEXT from the Table of muon dE/dx and Range: PDF␣
↪TEXT
The row with values like this should appear at the top of the resulting␣
↪table:

Sternheimer coef: a k=m_s x_0 x_1 I[eV] Cbar delta0
                  0.1091 3.3994 1.7418 4.2759 85.7 10.5961 0.00
```

  Enter those values. If the substance cannot be found, enter -100.

- **Create Media Table** Using the BaseM created above, create Sampling Tables, etc. Go to *LibLoft/Util/Elemag/BremPair*. Read the Readme file. Run the following command:

```
make clean
./CreateTab basic_media_table output_table_dir
```

  Tables can also be created by calling other commands from this command. For example, for Air:

```
Air
Air.xcom
```

Furthermore, create:

```
Air.GLB
Air.inp
```

These are necessary for the hadron interaction of dpmjet3. To create them, execute the iniGlauber command without variables, find out how to use it, and execute it. The five resulting files should be placed in *LibLoft/Data/Media*.

- **Sampling Test** To test the Sampling Table created above, refer to item 2) of Readme in BremPair.

6.7 HybASForUHEG-N-S-Effect

6.7.1 Purpose of this example

To learn about Hybrid AS calculation, handling magnetic pair creation, and various coordinate systems. Also, to understand how to obtain information when interactions occur. chook.f, param, primary to be used: Each has a suffix of -XXX (XXX represents BfieldLine, etc.). Use the same XXX for compiling and execution. Let's summarize the types and meanings of XXX.

XXX	Purpose/Description
BfieldLine	Drawing magnetic field lines
Mag	Interaction between UHE-gamma and magnetic field
sync	Studying synchrotron radiation effects
Bounce	Drift and reflection of charged particles due to magnetic field

Once you have decided the XXX you want to try, remove chook.f using rm chook.f, then create a symbolic link by executing the command:

```
ln -s chook.f-XXX chook.f
```

After that, use

```
make clean; make
```

to create an executable binary (cosmos...). To execute it, use the command:

```
./cosmos... < param-XXX > outputfile 2> errorfile
```

The primary-XXX file is used for sampling the 1ry.the interactions.

- Gamma rays undergo pair production in strong magnetic fields. This is essential to investigate the generation of cosmic rays from environments such as supernova remnants, not just Magnetars.
- However, even in weak magnetic fields like the Earth's magnetic field, UHE gamma rays undergo pair production. The threshold is around 3 ~ 4 x 10^19 eV.
- The origin of UHE gamma rays could be related to the GZK cutoff, but currently, there is no evidence supporting such a hypothesis.

- Nevertheless, it is important to know how UHE gamma rays would be observed if they arrived.
- However, it is not advisable to use extensive computation time for this purpose. By performing a Hybrid AS calculation with some roughness and using a 1-dimensional calculation to estimate the development of the electron component, we can capture the characteristics. Even in the case of UHE that would take 100 years for a single event calculation using a laptop, it can be done in less than 10 seconds with Hybrid AS calculation. * Hybrid AS is a technique where, at high energies, the usual particle tracking is performed, and if the energy of the electron falls below a certain threshold during its tracking, the subsequent development is replaced with an analytical solution (not a general term). * Since handling spread using analytical solutions is not easy, 1-dimensional calculations are usually performed. * Even after connecting with analytical solutions, particle tracking is possible up to low energies. By comparing the number of particles tracked with Hybrid AS and low energies, it is possible to examine the accuracy of Hybrid AS and more.
- The probability of gamma-ray pair production is proportional to $\sin\theta$, where θ is the angle between the direction of the gamma rays and the direction of the magnetic field lines.
- Therefore, it is also important to know the direction of the magnetic field lines. Everyone has seen the illustration of the magnetic field lines from the south pole to the north pole. Thinking that the magnetic field lines are concentrated near both poles, near the south and north pole points, they tend to think that the lines are being sucked into the Earth's surface. Although they do indeed concentrate in density, they also penetrate the Earth's surface at midlatitudes.
- It is also necessary to visually display such magnetic field lines of the Earth's magnetic field.
- By doing so, in the mid-latitude region where TA's observation equipment is located, it can be observed that gamma rays arriving from the north side have significantly larger $\sin\theta$ compared to tose arriving from the south side (Fig. 6.1).

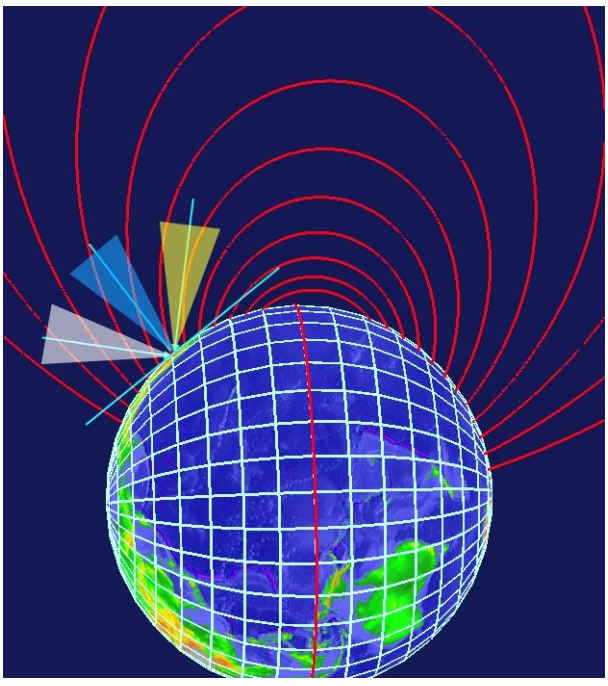

Fig. 6.1: Distribution of magnetic field lines in the vicinity of Lat ~ 39^o , Long ~ -112^o

- Those arriving from the north side undergo pair production at high altitudes, and the resulting electrons sequentially undergo synchrotron radiation, emitting gamma rays (proportional to E_e^2), and undergo considerable development before entering the atmosphere and causing the usual cascade process with their energy divided into smaller parts.

- As a result, the fluctuations in the development of air showers become smaller, and the maximum development point is in the upper atmosphere.
- Those coming from the south side have $\sin\theta$ 0, so the magnetic field effect is weak, and it becomes a region where the LPM effect works until the first pair production. The electrons produced by pair production are also affected by the LPM effect, causing a more significant disturbance and larger fluctuations compared to the ones coming from the north side. The maximum development point is also deeper.
- When dealing with the Earth's magneticfield, it is necessary to understand various coordinate systems, their mutual transformations, and the representation of the three components of the magnetic field vector B.
- In understanding the Earth's magnetic field, it is also worth taking a slight detour to examine the motion of low-energy charged particles influenced by the Earth's magneticfield.

6.7.2 Definition of Coordinate Systems

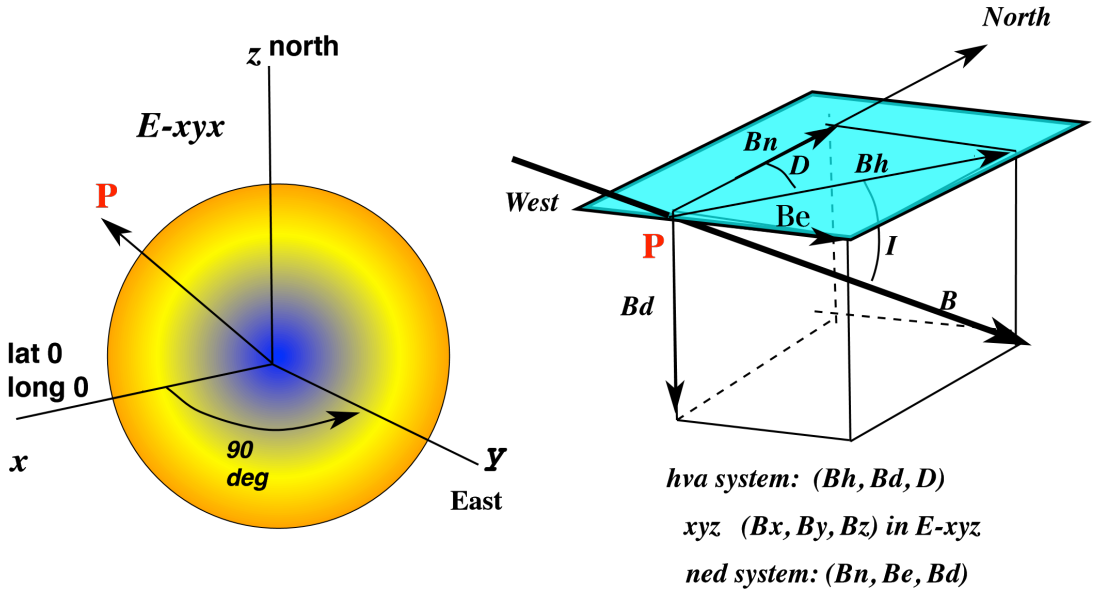

Fig. 6.2: Left: E-xyz system. Right: Components of the geomagnetic vector B

Earth-centered Cartesian Coordinate System (E-xyz):

Within Cosmos, the position coordinates are expressed using the E-xyz system. It is a coordinate system with the Earth's center as the origin, the direction of (0,0) latitude and longitude as the x-axis, the north pole direction as the z-axis, and the east direction as the y-axis.

LLH System:

When given the components of the position vector r in the E-xyz system, it may not be intuitive. Therefore, the LLH system is used, where the components are expressed as latitude, longitude, and height (Lat, Long, Height). It is a representation that is more familiar in everyday life. The height is defined as the distance from the mean sealevel defined for each location (in meters), and the latitude and longitude are in degrees.

SPH System:

Although less intuitive than LLH, the spherical coordinate system (SPH) is sometimes used. It consists of the polar angle θ, azimuthal angle φ (angle measured counterclockwise from the x-axis, with the y-axis at $\varphi = 90^o$), and thedistance r from the origin. It can be represented as :math:` (r sin theta cos varphi , r sin theta sin varphi , r cos theta).

6.7.3 Definition of the eomagnetic Vector

- Geomagnetic data is obtained from the International Geomagnetic Reference Field (IGRF), which is revised every five years (expected to be revised in 2025). The data can be found in *$LIBLOFT/Data/Geomag/igrf* (*igrf* is a symbolic link to the actual data). If an updated version is installed, the link needs to be updated as well.
- There are three ways to represent the magnetic field vector B at a certain point P near the Earth: * NED: ruby Copy code The standard representation in the IGRF is NED (North, East, Down), where B = (Bn, Be, Bd). * HVA: Another representation is HVA, where B = (Bh , Bd , D). Here, D is the declination angle between Bh and Bn. * XYZ: Within Cosmos, the E-xyz system's orthogonal components (Bx , By , Bz) are used.

6.7.4 Coordinate Transformations

- Transformation of Position Coordinates: The program takes the following form:

```
#include "Zcoord.h"
type(coord)::from ! Coordinate to be transformed.
type(coord)::to ! Transformed coordinate.
...
call ctransCoord2(sys, from, to)
...
```

Here, `sys`: represents the target coordinate system which is one of "xyz", "llh", or "sph". `from%sys`: contains one of these strings representing the current system. `from%r(1:3)`: represents the components of the position vector. `to:` receives the transformed components. `to%sys` will contain the same value as 'sys'. `to` may be the same varialbe as `from`.

- Conversion of Geomagnetic Vector B The program takes the following form:

```
#include "Zcoord.h"
#include "Zmagfield.h"
character(3)::sys ! Desired format of the transformed components: "ned",
↪"hva", or "xyz".
type(coord)::pos ! Position vector at the location where the magnetic␣
↪field is given.
type(magfield)::fromB ! Magnetic field vector to be transformed.
! fromB%r(i) for i=1,3 are the components,
! fromB%sys is the current format of the components.
type(magfield) :: toB ! Transformed magnetic field vector.
...
call ctransMagTo(sys, pos, fromB, toB)
```

Note: In the actual program, instead of type(coord)::pos, type(position)::pos is often used, which includes:

```
#include "Zcoord.h"
#include "Zpos.h"
type(position) :: pos
pos%xyz%r(i) ! i=1,3 are the components of the position vector.
pos%xyz%sys ! "xyz," "llh," etc.
pos%radiallen ! Distance from the origin.
```

(continues on next page)

(continued from previous page)

```
pos%depth ! Thickness in kg/m2 .
pos%height ! Altitude in meters.
pos%colheight ! Altitude of the last hadronic interaction in meters.
```

6.7.5 Exercise Details

Exercise to draw magnetic field lines as shown in Fig. 6.1. - Check if *chook.f* is linked to *chook.f-BfieldLine* using `ls -l`. - If it is not linked, execute the following commands:

```
rm chook.f
ln -s chook.f-BfieldLine chook.f
```

- In this example, the structure of the program follows the usual application format. At the stage when the overall initialization is complete (inside chookBgRun in *chook.f*), it calls the subroutine testBfieldline to perform the desired task and stops and exits upon return. The essential part is the content of testBfieldline.f, so if it is made the main program, chook.f is not necessary, but then it becomes necessary to manually read the IGRF data files. In this case,we take a shortcut to avoid such hassle.
- The program testBfieldline.f is included in the last part of chook.f.
- The structure of the program to draw magnetic field lines is as follows: 1. Read the starting point's latitude, longitude, and height from stdin. 2. If the latitude is above 90, exit. 3. Create the position information of the starting point. at that location. Note: The IGRF data is valid near the Earth. At distant points, the accuracy 4. Calculate the B decreases due to factors such as solar wind, and the variable "icon" becomes 1 instead of 0 in the `call cgeomag(...,icon)` for points over 105 km. This situation occurs when starting from points close to the poles. to the E-xyz system to simultaneously display the magnetic field lines and the Earth. 5. Convert B to the E-xyz system to simultaneously display the magnetic field lines and the Earth 6. Move the current position a short distance "leng" in the direction of B. the northern hemisphere, the direction vector of the magnetic field lines needs to be reversed to draw the lines coming from the southern hemisphere (controlled by the variable "reverse" in the program). 7. If the altitude at that point is below the starting point, exit the loop and proceed to c). 8. The position information consists of three components (x, y, z), but to draw the magnetic field lines with the Earth, it is necessary to output them in the same format as the trace information in FirstKiss, which allows drawing the tracks of particles. The output format of each line is: x y z code K.E charge S. "K.E" (kinetic energy) can be any value, and setting "code" and "charge" to 2 and 1, respectively, will draw positrons, which re represented by default as red lines. "S" represents the accumulated traveled distance divided by the speed of light (β), but since it is not used, any value can be used. When drawing multiple magnetic field lines, separate the data of each line with two lines of blank space. 9. back to b). 10. c: back to a).
- Compilation and Execution: 1. Confirm that the link of chook.f is correct. 2. Execute the following commands: make clean; make. This will create the executable file *cosmosxxx* (*xxx* depends on *$(ARCH)*, which can be *MacIFC*, etc.). 3. The required input data for execution is param-Bfieldline. Specify the year of the geomagnetic field data with YearOfGeomag, but in practice, it is rarely used. Any valid value can be provided. 4. Where to provide the starting point data (LLH)? Since it is set to read from stdin, in the usual case, when the program starts running, you would input the data from the terminal. However, in this case, the execution command (`./cosmosMacIFC < param-Bfieldline`) uses param-Bfieldline as stdin. 5. Therefore, after reading *&PARAM* and *&HPARAM* in param-Bfieldline, the program continues to read LLH data. In the example, the longitude is fixed at -112 degrees, and the latitude is given from 70 to 10 degrees in 5-degree increments. The altitude is constant at 30 km. 6. Execute the command: `./cosmos$ARCH <`

`param-Bfieldline` > `TraceBfield`. This will create trace data in the same format as the regular particle tracking. 7. To visualize the data, use the command: dispcosTraceGeomv TraceBfield earth. Note: Currently, geomview is required for displaying the trace. Omitting "earth" will result in no Earth being displayed. If you want to display the shower trace with the Earth, you need to identify the position of the shower and zoom in to see it. Exercise to draw cosmic rays that follow magnetic field lines, as shown in Fig. 6.3.

- Although we are dealing with UHE-gamma (Ultra-High-Energy Gamma, let's see what happens to low-energy cosmic rays as they follow the Earth's magnetic field lines. This movement called "drift and bounce" will be clearly seen in thefollowing animation:

http://cosmos.icrr.u-tokyo.ac.jp/cosmosHome/Movie/e+e-BounceDrift.mov

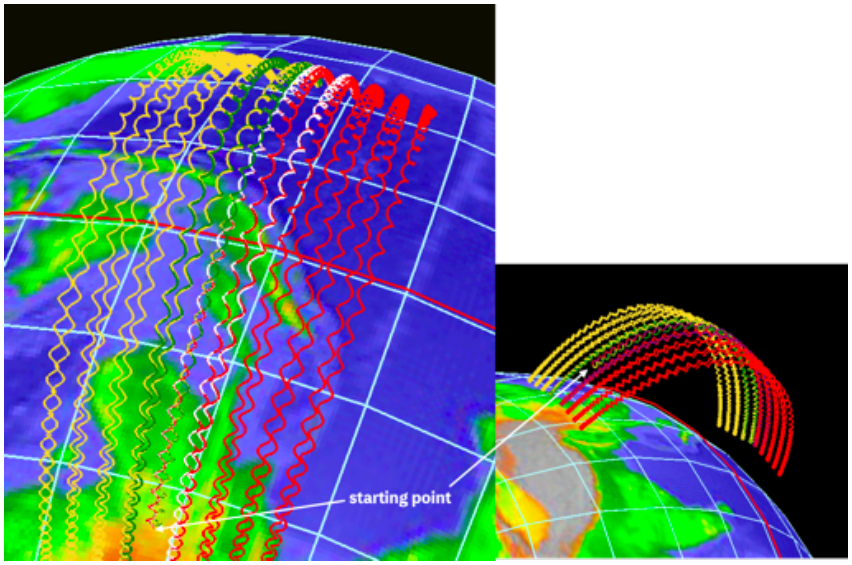

Fig. 6.3: Motion of positive and negative charged particles in the Earth's magnetic field. Electrons (yellow), positrons (red), protons (white), and antiprotons (green) with the same momentum. They move by alternating between Bounce and Drift. Electrons and antiprotons shift to the east, while positrons and protons shift to the west. Near the starting point, the trajectories of electrons and positrons are obscured by antiprotons and protons, respectively, as their orbits coincide. Protons and antiprotons decay faster. The behavior of Drift is easier to underst and from the left figure.

- Compilation and Execution:
  1. Since we will be using chook.f-Bounce, chook.f needs to be linked to it. Note that chook.f itself is not actually used, so attention should be paid to its content.
  2. Use the commands: make clean; make to create the executable program.
  3. The parameter file is "param-Bounce" + To fix the injection point, angle etc, we should know the definition of the following input parameters.
     - Reference observation point: The altitude is the altitude of the lower observation surface plus the OffsetHeight. The default value of OffsetHeight is 0. Latitude and longitude are defined by LatitOfSite and LongitOfSite.
     - Zenith angle and azimuth angle are determined by sampling using input parameters. Their values are determined by the direction cosine of the vector drawn from the reference observation point to the source (starting point) of the cosmic ray if it traveled in a straight line from the source. This is not the angle at the starting point (it's the opposite of the vector drawn from the starting point to the observation point).

  - HeightOfInj: The altitude of the cosmic ray's starting point. The latitude and longitude of the starting point are determined by the above conditions. If the zenith angle at the starting point is higher than the reference observation point (if the zenith angle is not zero), the value will be smaller at the observation point. The cosine value will be larger.
  - XaxisFromSouth: South refers to the south along the longitude. When defining the coordinates of a horizontally placed detector, it specifies the angle (in degrees, ranging from -360^o to 360^o) that sets the

x-axis from the south in an eastward direction. If the absolute value is greater than 360^o (default), the x-axis is aligned with the magnetic east. The z-axis is oriented upward in the vertical direction.

- Parameter values in param-Bounce:
  - Azimuth=(90,90): Cosmic rays incoming from the north.
  - CosZenith=(0.2,0.2): The cosine of the zenith angle at the observation point is 0.2. The value at the starting point is written to stderr at the end of program execution, which is 0.43 in this case.
  - BorderHeightH = 5000e3, BorderHeightL = 40e3: Particle tracking is performed within this altitude range. The default value is 0, in which case the tracking range is below the starting point of the 1-ry particle and the lowest observation surface.
  - The incident cosmic ray is specified in "primary-Bounce". For example:

```
#---------------------------------
'e-' 'GeV' 'p' 'd' 0 /
0.3 1
0. 0.
```

    This specifies a 0.3 GeV/c electron ('e-').

  - Trace=41: Record the particle trace. The coordinate system is E-xyz. Since TraceDir='./', the trace will be written as *trace1* in the current directory.
  - With the current settings, the particles are oriented downward, and the magnetic field lines are oriented upward. However, due to the low energy, the particles end up spiraling along the magnetic field lines and moving upward.

4. Execution: Run the command `./cosmosXXXX < param-Bounce` to execute the program. To simulate positrons, protons, or antiprotons, modify the 'e-' in primary-Bounce to 'e+', 'p', or 'pbar' (or p~), respectively.

Getting more sidetracked: Let's explore what happens to cosmic rays as they perform Bounce and Drift motions along the Earth's outer circumference. We will examine their relationship with the magnetic field and how far they travel.

1. The first systematic observation to answer such questions was probably conducted by AMS01 (Physics Letters B472, (2000). 215-226; Physics Letters B 484, (2000) 10-22, AMS collaboration).
2. To reproduce such observations in simulations, it is not enough to manually input the position and momentum of incident particles. It is necessary to track the secondary particles generated by the interaction with the atmosphere by randomly injecting low-energy cosmic rays into the Earth. In that case, considerations must be made for the energy spectra of each type of incident cosmic ray, the location-dependent cutoff due to the Earth's magnetic field, and other factors.
3. An example is shown in Fig. 6.4.

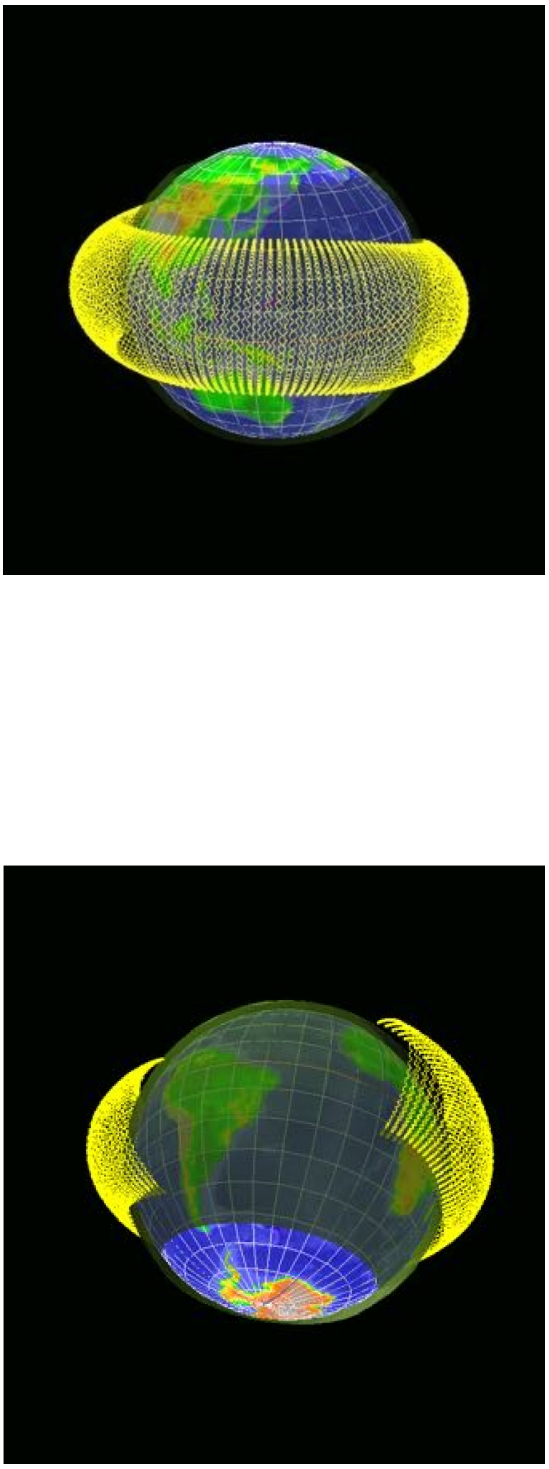

Fig. 6.4: Example of long-life e- ; AM defines Slong life a flight time > 0.2 s. This example is likely in the 20-second range. It can be seen from the video at that the electron is a descendant of the proton collision.

4. From observations, AMS01 determined the momentum, origin, and locations where electrons, positrons, protons, and antiprotons penetrate into the atmosphere. They calculated the trajectories and flight times of these particles and obtained distributions as shown below.
5. The specific simulation method for the corresponding their analysis will be discussed separately. For now, let's focus on the results.
6. In the figures, λ_m represents the geomagnetic latitude in radians. AMS uses the symbol Θ_M and radians as units.

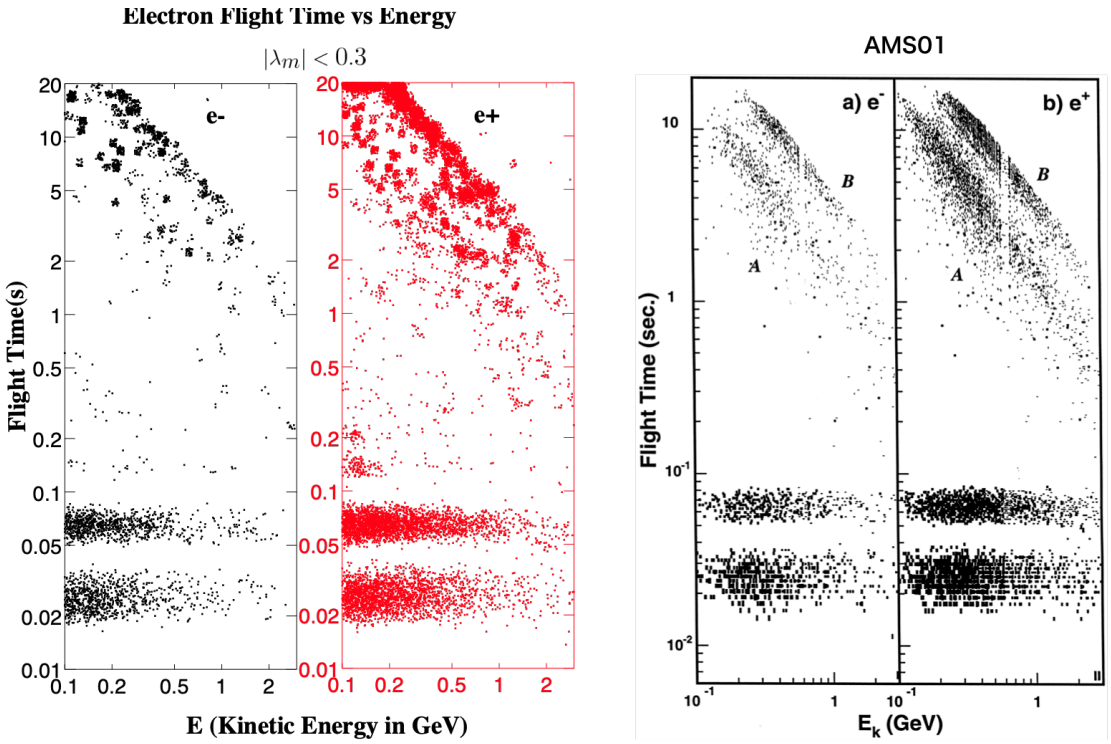

Fig. 6.5: Correlation between energy and flight time of e- and e+ . The right side shows the observation results, and the left side shows the Monte Carlo simulation results.

Now let's move on to the simulation of pair production by UHE-gamma in magnetic fields.

1. Since we will be using chook.f-Mag, we need to link it with chook.f.
   - Monitor the name of the first interaction of gamma rays using chookGInt and define a common variable to hold it. Treat it as a module at the beginning of the program.

```
module modFirstIntInfo
character(5) :: Intname ! one of mpair, pair, ...
end module modFirstIntInfo
```

   - In chookBgRun, call epResetEcrit to reset the critical energy to 81 MeV. The default value in the PDG is approximately 87 MeV, but it seems too large for the calculation of AS size (Ne). Since the hybrid calculation is an approximate calculation, it doesn't need to be too precise, but 81 MeV is chosen.

- In chookBgEvent, initialize the variable that holds the interaction name for each particle to blank. Intname = ‘ ‘.
- chookObs does not record the passage of individual particles through the observation surface.
- In chookEnEvent, obtain information about the incident particle from cqIncident. Retrieve the direction vector of the source and calculate the zenith angle and azimuthal angle. Use cqFirstIP to determine the position of the first collision point. Use cgeomag to obtain the magnetic field strength (B) at that point, and use cgetBsin to calculate $B\sin(\theta)$ (where θ s the angle between the particle trajectory and the magnetic field line). Write the event header, including “Bsin”, “Zenith angle at the lowest layer of 1ry”, “Zenith ale at the source of 1ry”, “Azimuthal angle”, “Altitude of the first collision point”, and “Interaction name”.
- Next, calculate the AS size at each observation surface. Use “out#” as a marker. Write the “Layer number”, “Vertical depth (g/cm2)”, “Slant depth”, “Zenith angle”, “Azimuthal angle”, “Bsin”, and “Interaction name”. (Note: Slant depth is calculated by dividing the vertical depth by cos(zenith angle). It introduces significant errors for zenith angles greater than 60 degrees.)
- chookGInt: Retrieve the name of the first interaction of gamma rays as IntInfArray(ProcessNo)%process and set never=1 to avoid further calls in this event (In older interface, it was never called for all subsequent events).

2. Use param-Mag for input parameters. The essential ones are:
   - Azimuth = (0.0, 360.0): UHE-gamma arriving from the north should experience the magnetic field effect. Therefore, we only need to investigate the 1ry arriving from approximately 90 ± 30 degrees (where the magnetic field effect is not signfificant) and the south side at approximately 270 ± 30 degrees. However, for the incident 1ry, we consider all azimuth angles and examine the results for each azimuth angle separately.
   - Zenith angle is set to cos(0.6 ~ 0.8).
   - Observation altitude is 100 ~ 1000 g/cm2 . The size of Ne is determined using ASDepthList.
   - Number of events is 1000.
   - Set Generate to ‘as’ and focus on the size of Hybrid AS. Since low-energy particles are not tracked, ‘em’ is not included.
   - HeightOfInj = 5000e3: Inject particles from a sufficiently high altitude, which can be considered as vacuum, higher than the usual 100 km.
   - LatitOfSite = 39.3, LongitOfSite = -112.9: Use the TA site as an example of a mid-latitude site.
   - LpmEffect = T: LPM effect is included by default.
   - KEminObs = 8*7.0e5: Consider 1ry with energies above 10^19 eV for Hybrid calculations, so the energy of particles to be tracked (observed) is set to be 700 TeV or higher.
   - PrimaryFile = ‘primary-Mag’: Specifies the gamma rays and their energies in this file.
   - HowGeomag = 2: B varies with position.
   - MagBrem = 2, MagBremEmin = 3e7: Include synchrotron radiation for energies above 3 x 10^16 eV. If MagBrem = 1, gamma rays are not generated, only the energy loss of electrons is considered.

- MagPair = 1, MagPairEmin = 2e10: Consider magnetic pair production. The minimum energy is 2 x 10^19 eV.
- RatioToE0: Follow hadronic interactions down to per-nucleon energy of RatioToE0 * E0.
- WaitRatio = 0.005: Do not connect to the analytic solution of AS until the energy of electrons becomes less than or equal to E0 * WaitRatio.
- UpsilonMin = 3e-3: High-energy magnetic effects become significant at $\Upsilon = \gamma B/B_c \approx 0.1$ or higher. Consider synchrotron radiation for values above this threshold (Bc = 4.4 x 10^13 Gauss).

3. Compile and execute:

```
make clean; make;
./cosmosXXX < param-Mag > YYYY
```

From the output, you should be able to create graphs similar to the following (replace YYYY with any desired name).

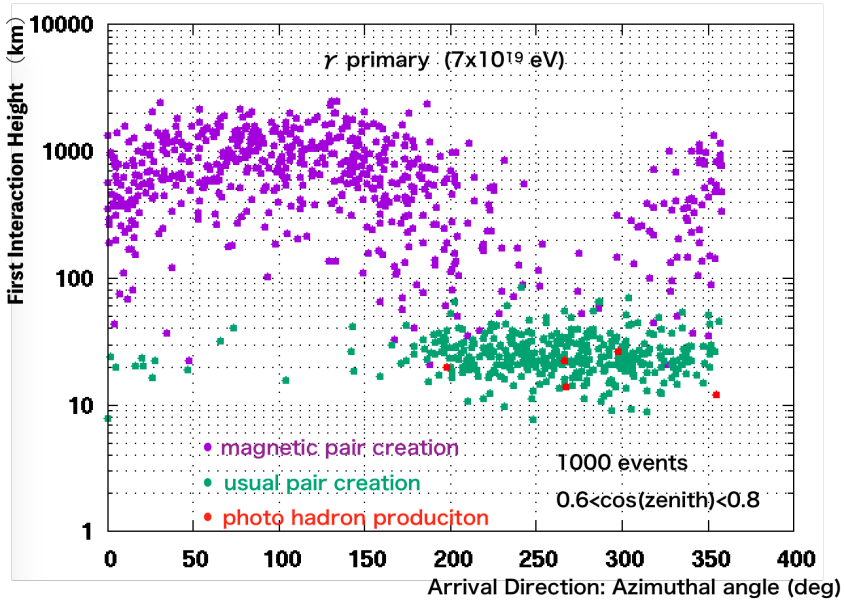

Fig. 6.6: Correlation between arrival direction and the first interaction altitude. Interactions include magnetic pair creation, regular pair creation, and photo-hadron production. The energy of 1ry is 7 x 10^19 eV.

6.7.6 Magnetic Brems (Synchrotron Radiation)

- Electrons and positrons generated through pair production due to the magnetic field are UHE (Ultra High Energy). Even if they penetrate into the atmosphere, the LPM effect causes significant fluctuations in shower development. However, the fluctuations become smaller because synchrotron radiation occurs in the geomagnetic field. Since the electrons are UHE, classical equations cannot be used to describe synchrotron radiation.
- The program uses chook.f-sync, which is a modified version of chook.f called chook.f-Mag that retrieves information about electron synchrotron radiation through subroutine chookEInt in chook.f-sync. Therefore, chook.f is linked to chook.f-sync.
  - In chookEInt: if (IntInfArray(ProcessNo).process == “sync”) then This detects synchrotron

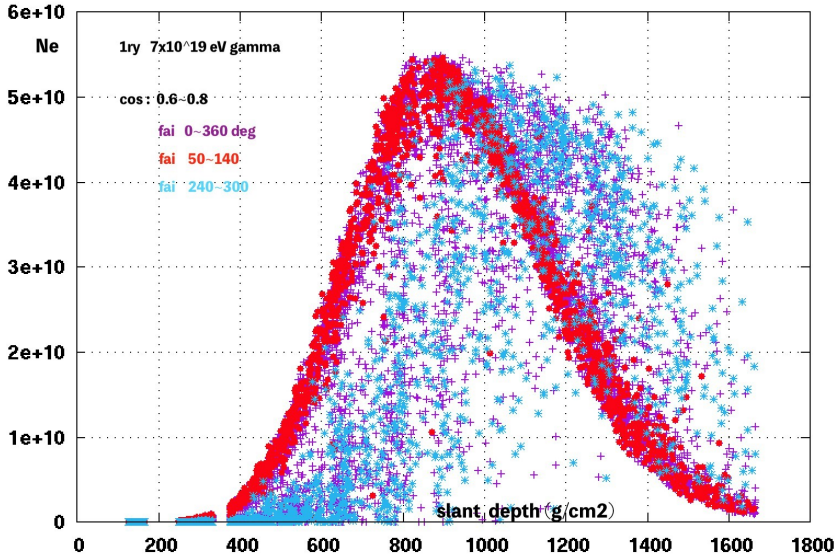

Fig. 6.7: Ne transition curve. The fluctuations are significantly different depending on the azimuth angle.

radiation and outputs the energy of gamma rays and the energy of the parent electron with the marker sync#.

- Since synchrotron radiation can occur multiple times and we want to output all of them, set never = 0.
- Use param-sync. The basics are the same as param-Mag. The primary is primary-sync, and we use a slightly lower energy of 5 x 10^19 eV (5e10 GeV).
- Azimuth uses the high probability range of 60 ~ 120 degrees where magnetic pair creation occurs. The number of events is reduced to 100.

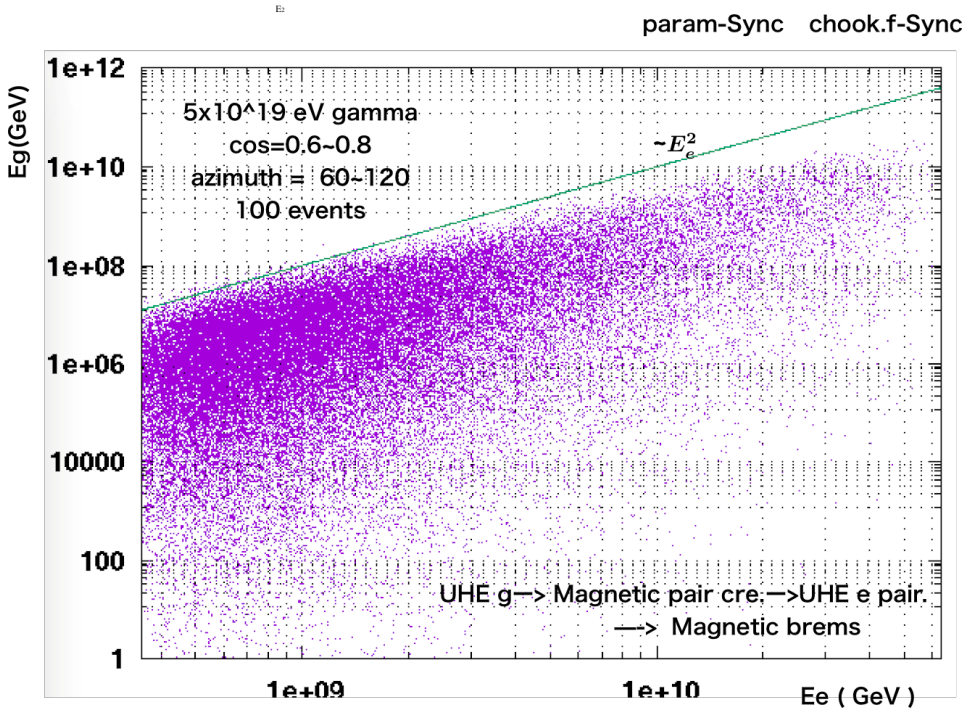

Fig. 6.8: Correlation between the energy of gamma rays and the energy of the parent electron from synchrotron radiation of electrons generated through magnetic pair creation of UHE-gamma with an energy of 5 x 10^19 eV. As Ee approaches Egamma1ry , Egamma max ~ Ee2 deviates. Examples where more than 10% of the energy is radiated in asingle step can be observed for UHE electrons.

CHAPTER

SEVEN

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 `chook0bs` 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.

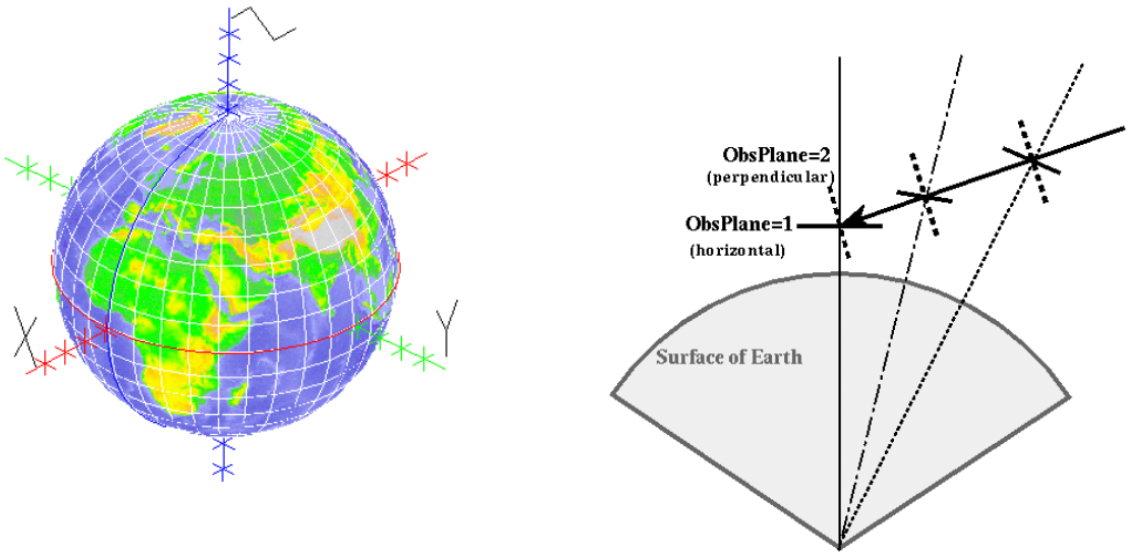

Fig. 2.1: The basic (interal) coordinate system (left Fig.) and observation plains (right Fig.)

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	$\mathrm{kg/m^2}$	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	$\mathrm{kg/m^2}$	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

APPENDIX

D

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 from the center of Earth
- **% 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)]* :: Moliere unit
- **% 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

`type primaries`

Type fields

- **`% each`** (maxNoOfComps) *[component]*
- **`% cummInteFlux`** (maxNoOfComps) *[real(8)]*
- **`% no_of_comps`** *[integer]* :: how many diff. compositions
- **`% NoOfSamplings`** *[integer]* :: total number of samplings including ones discarded by cutoff
- **`% NoOfSampComp`** (maxNoOfComps, 2) *[integer]* :: 1 is for number of sampling including discarded ones due to cutoff. 2 is only for employed ones. after a sampling of a 1ry, the following is fixed.
- **`% label`** *[integer]* :: sampled primary label
- **`% sampled_e`** *[real(8)]* :: sampled energy(or momentum) as defined in etype. If this is in total energy in GeV, it is the same as `particle%fm%e`
- **`% particle`** *[ptcl]*

Header
Zprimary.h

type component

1 component of 1ry

Type fields

- **% label** *[integer]* :: composition label number
- **% symb** *[character(16)]* :: 'P', 'gamma' etc.
- **% eunit** *[character(3)]* :: 'GeV' etc
- **% etype** *[character(4)]* :: 'KE/n' etc
- **% diff_or_inte** *[character(1)]* :: 'd' or 'i'
- **% flatterer** *[real(8)]* :: dI/dE*E**flatterer
- **% cut** *[real(8)]* :: lower cut off.
- **% cut2** *[real(8)]* :: upper cut off.
- **% energy** (maxSegments+1) *[real(8)]* :: segment left energy
- **% flux** (maxSegments+1) *[real(8)]* :: input flux dI/dE * E**flatterer above; from input table directly below; made by subroutines
- **% code** *[integer]* :: particle code
- **% subcode** *[integer]*
- **% charge** *[integer]*
- **% togev** *[real(8)]*
- **% norm_inte** (maxSegments+1) *[real(8)]* :: normalized integral flux > E at segment left value
- **% beta** (maxSegments+1) *[real(8)]* :: dI/dE=(true flux)= const*E**(-beta)
- **% no_of_seg** *[integer]* :: no of segments given
- **% inte_value** *[real(8)]* :: integral flux from min. E
- **% emin** *[real(8)]* :: min and max energy defined

Header

Zprimary.h

D.2 User accessible subroutines and functions

D.2.1 Useful

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. You may call this at any moment after the event initialization has been ended.

Parameters

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

`subroutine cqNoOfPrim`(*no*)

inquire the number of primaries sampled specified in the primary spectrum file (MeV/n etc). You may call this after the event initialization has been ended. Note this is not necessary the total energy of the incident.

Parameters

no *[integer,out]* :: no. of sampled primaries so far.

`subroutine cqPrimE`(*pOrE*)

inquire sampled primary energy or p or rigidity as it is

Parameters

pOrE *[real(8),out]* :: sampled primary energy or p or rigidity

`subroutine cqPrimary`(*prm*)

inquire all about current primaries (input primary spectrum information)

Parameters

prm *[*primaries*,out]* :: primary information

`subroutine cqFirstID`(*depth*)

inquire the first interaction depth of the incident particle (vertical depth). *After uv6.30, the knock-on process by p, He, etc is not regarded as the first interaction point;* only their nuclear interaction is picked up. For non-nucleus, all interaction types are considered. If you want to have more detailed control, you may use chookNEPInt, chookGInt and/or chookEInt.

Parameters

depth *[real(8),out]* :: the first interaction depth

`subroutine cqFirstIPI`(*aTrack*)

inquire the complete first interaction point info. as a track of the primary. The definition of the first interaction point is the same as for cqFirstID.

Parameters

aTrack *[*track*,out]* :: if ptrack%pos%depth=0. no interaction has been occurred

`subroutine cqIncident`(*aTrack*, *angleAtObs*)

inquire incident particle. You may call this at any moment after the event initialization has been ended. The program unit containing this call must have `#include "Ztrack.h"`.

Parameters

- **aTrack** *[*track*,out]* :: incident particle track information. You will see the ingredient by looking at the chookObs routine.
- **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.

D.2.2 Manager

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 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

D.2.3 EM

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

D.2.4 Tracking

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

Average dedx (2.2×10^{-2} is set 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]* :: level # 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 cdet2prim(*ly*, *a*, *b*)

xyz to primary system coord. conversion

Parameters

- **ly** *[integer(2),in]* :: base level # (xyz2prim case)->BaseL)
- **a** *[coord,in]* :: input. coord. in 'xyz'
- **b** *[coord,out]* :: output. transformed coord. in 'prim'

subroutine cxyz2primD(*a*, *b*)

cxyz2primD: xyz to primary system for Direction cos. See the parameter description of `cxyz2prim()`

subroutine csetPos(*location*)

set position information for a given xyz

Parameters

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

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

D.2.5 Atmosphere

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

`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

D.2.6 Geomag

`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 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/

D.2.7 General

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*) `NoOfSites` can be set upto 50 for the default configuration, which is defined by MAX_NO_OF_SITES in *Zmaxdef.h*.

NoOfASSites *[integer]*

maximum index for observation depths. (*Zobsv.h*) `NoOfASSites` can be set upto 50 for the default configuration, which is defined by MAX_NO_OF_AS_SITES in *Zmaxdef.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*)

AngleAtObsCopy *[coord]*

direction cos at the deepest Obssite in 'det' system (*Zincidentv.h*)

DcAtObsXyz *[coord]*

transfroamtion of `AngleAtObsCopy` to "xyz" system (*Zincidentv.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
D_s	kds	30	Ξ_c	kXic	31
Ω_c^0	komeC0	32	τ	ktau	33
ν_τ	kneutau	34	η'	ketap	35
Δ	kDelta	36	Ξ_c^0	kXic0	37

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 4.2
QGSJET II-04	qgsjet2		See Section 4.2
QGSJET III	qgsjet2		See Section 4.2
EPOS 1.99	epos		See Section 4.2
EPOS LHC v3700	epos		See Section 4.2
Sibyll 2.1	sibyll		See Section 4.2
Sibyll 2.3c	sibyll		See Section 4.2

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

A

AngleAtObsCopy (*fortran variable*), 72
ASObsSites (*fortran variable*), 72
assite (*fortran type*), 64

B

bpTbl (*fortran type*), 62

C

cavedEdx() (*fortran subroutine*), 68
ccos2kf() (*fortran subroutine*), 67
ccos2pdg() (*fortran subroutine in module modCodeConv*), 67
cdet2prim() (*fortran subroutine*), 69
cdet2xyz() (*fortran subroutine*), 69
cdet2xyzD() (*fortran subroutine*), 69
cgeomag() (*fortran subroutine*), 70
cgetNmu() (*fortran subroutine*), 68
chookBgEvent() (*fortran subroutine*), 13
chookBgRun() (*fortran subroutine*), 13
chookEInt() (*fortran subroutine*), 13
chookEnEvent() (*fortran subroutine*), 13
chookEnRun() (*fortran subroutine*), 13
chookGInt() (*fortran subroutine*), 13
chookNEPInt() (*fortran subroutine*), 13
chookObs() (*fortran subroutine*), 13
chookTrace() (*fortran subroutine*), 13
ciniTracking() (*fortran subroutine*), 69
ckf2cos() (*fortran subroutine*), 67
cllh2eCent() (*fortran subroutine*), 69
CompASAge (*fortran variable*), 71
CompASNe (*fortran variable*), 71
component (*fortran type*), 64
coord (*fortran type*), 53
cprintPrim() (*fortran subroutine*), 67
cqEventNo() (*fortran subroutine*), 65
cqFirstColMedia() (*fortran subroutine*), 69
cqFirstID() (*fortran subroutine*), 66
cqFirstIPI() (*fortran subroutine*), 66
cqIncident() (*fortran subroutine*), 66
cqIniRn() (*fortran subroutine*), 65
cqNoOfPrim() (*fortran subroutine*), 66
cqPrimary() (*fortran subroutine*), 66
cqPrimE() (*fortran subroutine*), 66
csetMagField() (*fortran subroutine*), 71
csetPos() (*fortran subroutine*), 69
ctransMagTo() (*fortran subroutine*), 70
cvh2den() (*fortran function*), 70
cvh2temp() (*fortran function*), 70
cwriteParam() (*fortran subroutine*), 67
cxyz2det() (*fortran subroutine*), 68
cxyz2detD() (*fortran subroutine*), 69
cxyz2primD() (*fortran subroutine*), 69

D

DcAtObsXyz (*fortran variable*), 72
direc (*fortran type*), 53

E

element (*fortran type*), 55
epmedia (*fortran type*), 55
epResetEcrit() (*fortran subroutine*), 68

F

fmom (*fortran type*), 53

K

klena() (*fortran function*), 71

M

magfield (*fortran type*), 64
Media (*fortran variable*), 72
MediaNo (*fortran variable*), 72
MovedTrack (*fortran variable*), 72

N

NoOfASSites (*fortran variable*), 71
NoOfSites (*fortran variable*), 71

O

ObsSites (*fortran variable*), 72

P

position (*fortran type*), 53
primaries (*fortran type*), 64
ptcl (*fortran type*), 54

S

site (*fortran type*), 63
SmpCnst (*fortran type*), 58

T

track (*fortran type*), 54
TrackBefMove (*fortran variable*), 72