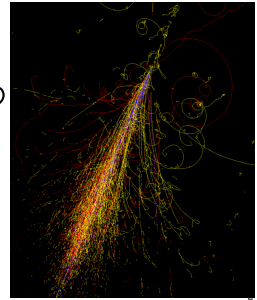
第三回 COSMOS講習会

背景説明
FirstKissサンプルの内容説明
実践練習

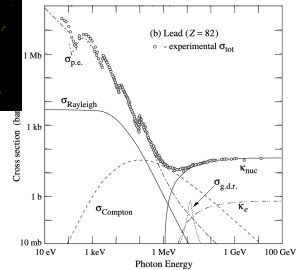
背景

- 「空気シャワーシミュレーション = CORSIKA」という現状
- 空気シャワー観測の精密化、応用の広がりから空気シャワーシミュレーションの重要性が増している。CORSIKAだけでいいか？=> CORSIKAも大幅アップデート(CORSIKA8)が進行中
  - 空気シャワー計算は直接較正ができないため、複数のコードが必要
  - 応用に向けた自由度の高いコードが必要（複数のコードが個性を出してもいい）
- 笠原が開発したCOSMOSが古くから存在
- COSMOSを受け継ぎ、CORSIKAの検証・CORSIKAと違う応用、を実現しよう。
- 空気シャワーシミュレーション開発を通して、空気シャワーの物理・シミュレーション技術を学び続けよう => COSMOS Xの開発
- 主催者も「学び」中です。COSMOS Xは過去のCOSMOSから大幅変更をしたので、未確認の機能がたくさんあります。教えながら学ぶので、一緒に開発に参加するつもりで協力お願いします。=> Slackを継続します

空気シャワー
シミュレーションとは何か？



- 粒子ひとつひとつの運動と反応を追い続ける
  - 「十分低いエネルギー（恣意的）」になったら吸収
- 連続的な変化：電離損失、電磁場中の運動
  - 「十分に短いステップ」で計算
- 不連続な反応：「平均自由行程∝（密度・断面積）$^{-1}$」や「寿命」で表現される確率過程
  - 粒子増殖とエネルギー減少、粒子種の変化
  - 反応後の粒子種分布、エネルギー分布、角度分布
  - 正確な素課程（微分断面積）の情報が必要



異なる素課程が競合する例

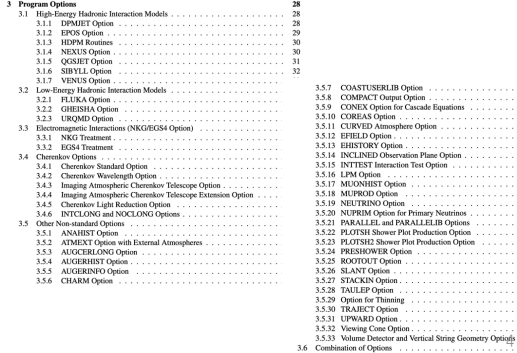
- 次に何が起きるか？の確率過程をサイコロ（乱数）を振って決めていく

  => モンカルロ計算・一人の母親から始まる数10世代の巨大人生ゲーム
- 空気シャワーの平均的描像ではなく、fluctuationも評価可能

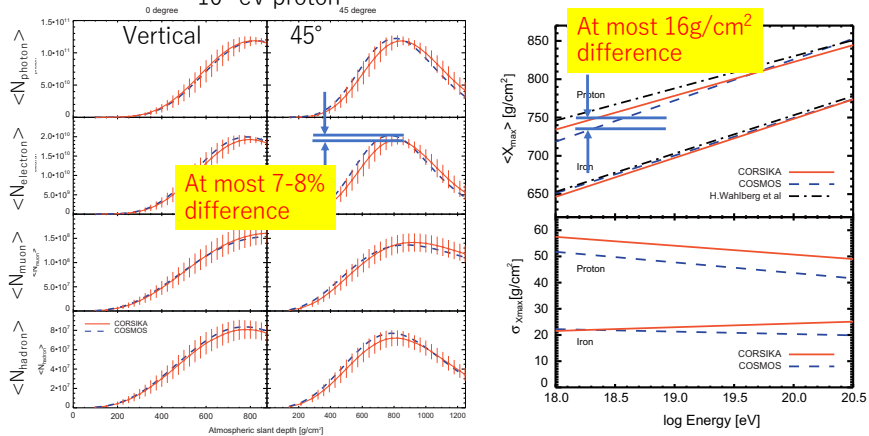
CORSIKAでいいじゃん？

- いいよ
- CORSIKAを自分で使える？やりたいことができる？
  （誰かが作ったパッケージを動かしているだけじゃない？）
- たくさんのオプションを「選ぶ」
- 自分のやりたいオプションがあるか？

- CORSIKA8が普及したら何が起きる？
  「え、CORSIKA7と違う！」



CORSIKAと COSMOSの比較



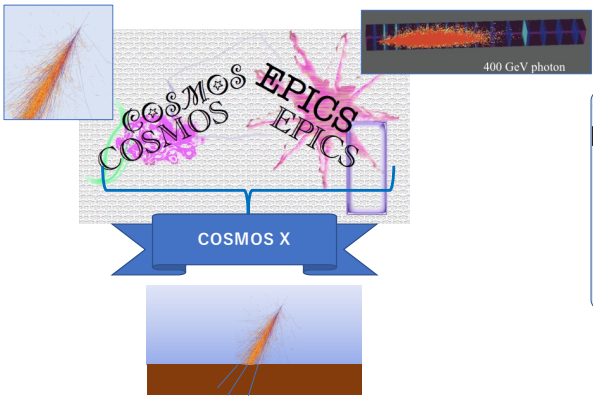
S. Roh et al., Astropart. Phys., 44 (2013) 1–8

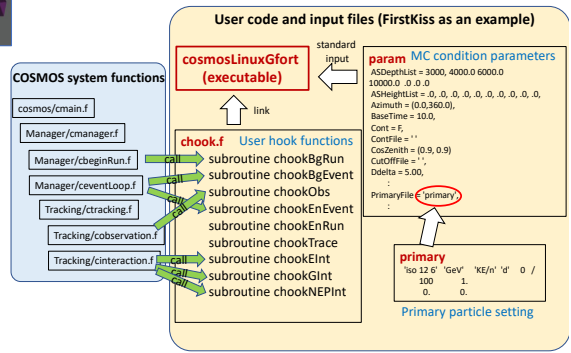
ICRC2021 (Berlin, online) ID725

COSMOS X as a general purpose air shower simulation tool

T. Sako,[a] T. Fujii,[b,c] K. Kasahara,[d] H. Menjo,[e] N. Sakaki, [f] N. Sakurai,[g] A. Taketa,[h] Y. Tameda[i] for the COSMOS X development team

[a] Institute for Cosmic Ray Research, the University of Tokyo, [b] Hakubi Center for Advanced Research, Kyoto University, [c] Graduate School of Science, Kyoto University, [d] Faculty of Systems Engineering and Science, Shibaura Institute of Technology, [e] Institute for Space-Earth Environmental Research, Nagoya University, [f] Computational Astrophysics Laboratory, RIKEN, [g] Graduate School of Science, Osaka City University, [h] Earthquake Research Institute, University of Tokyo, [i] Osaka Electro-Communication University, Department of Engineering Science

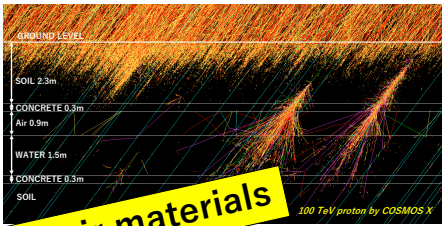




- Air shower MC simulation tool becomes more and more important in CR physics
  - *PID, muon puzzle, LHC, thunder cloud, solar gamma rays, etc…*
- COSMOS is an air shower MC simulation tool with flexible user control functions
- Combining with a detector simulation tool EPICS, ***extended COSMOS, COSMOS X***, is born

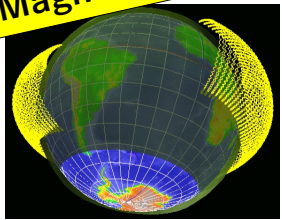
COSMOS X Applications

A 100TeV proton shower above and below the ground



Non-air materials

Magnetic field



Trajectory of a charged particle trapped in the geomagnetic field

Tracking of a 10TeV proton in the solar atmosphere

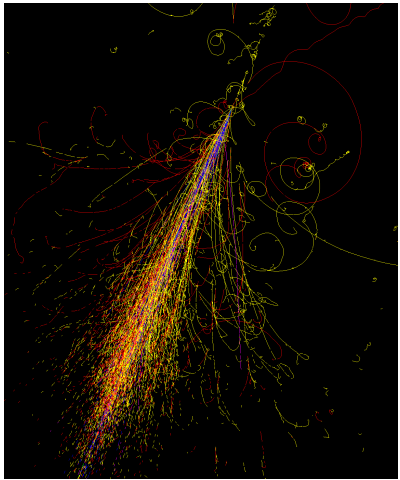
Non-earth atmosphere

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

- Variety of applications are possible using COSMOS X
  - Non-air materials : any gas, liquid solid
  - Arbitrary magnetic and electric fields
  - Non-earth conditions : spherical shell structures with a common center and arbitrary radi
- Try, enjoy and feedback us!!

FirstKissの説明

First Kiss sample



全ての粒子の大気中のトラックを出力し可視化

空気シャワーシミュレーションに必要な入力

- 入射粒子の
  - 種類
  - エネルギー

  } primary
  - 到来方向（天頂角、方位角）
  - 入射高度
- 観測地点の
  - 緯度、経度、標高
- 繰り返し回数
- 使用する相互作用モデル

} param

これらはユーザーの目的によらず必須

空気シャワーシミュレーションで必要な出力

- 高度指定？大気中？
- 全粒子？条件指定（エネルギー、粒子種…）？
- 粒子の物理量？トラック？ヒストグラム？

} chook.f

これらはユーザーの目的に大きく依存

```
sako@tadhcp231 FirstKiss % ls
Alt                 Readme              chook.mk            param.demo
CMakeLists.txt      Seed                chookHybAS.f        primary
Makefile            Vis                 cosmosLinuxGfort    trace1
Makefile.legacy     chook.f             param               trace2
```

この３つ（のうち重要部分）だけ理解しよう

primary file

```
sako@tadhcp231 FirstKiss % cat primary
#
#   'e-'  'MeV'     'KE/n'  'd'     0   /   24-12 Mg  22-11 Na
#
#------------------------------------------------------------
  'iso 14 7'  'GeV'     'KE/n'  'd'     0   /
          100          1.
           0.              0.
```

Nucleon = 核子(陽子と中性子)
Nucleus = 原子核
Nuclei　= 原子核の複数形

- 入射粒子の種類とエネルギーを指定するファイル
- #で始まる行はコメントなので無視
- 本体の 1 行目
  - ‘iso 14 7’ は「質量数14, 原子番号 7の原子核」つまり「窒素原子核」
  - エネルギーの単位は “GeV”, エネルギーの定義は “KE/n” (kinetic energy per nucleon)
    - 注：1GeV/nの運動エネルギー（KE）を持つ窒素原子核は、全体で 14GeVの運動エネルギーをもつ。質量エネルギー（約1GeV/核子）も含めると全体で 約28GeVのエネルギー。入射原子核のエネルギーの定義には要注意。
  - スペクトル（ここでは関係ない）のタイプは “d”つまり微分スペクトル
  - フラックスの重み（ここでは関係ない）はゼロ

- 本体の 2 行目
  - 100 GeV/n
  - 相対フラックス強度 1

- 結局、このファイルは「運動エネルギー 100GeV/nの窒素原子核」を指定するもの

さまざまな primary指定

- CosmosX_0.08/LibLoft/Data/Primary/ にサンプルあり
- 右の例は、protonとHeの mixed compositionでそれぞれの微分スペクトルを指定すると、flux比に応じて入射粒子種とエネルギーを決定
- 下の例は 10TeV ガンマ線入射の例
- 各スペクトルの最後は”0 0”で指定

```
#
#
#-------------------------------------------------------------
  'gamma'   'TeV'     'E'  'd'     0   /
       10.              1.
            0.               0.
```

```
#
# The next is an example of a complex composition at low energy
#  This is a plausble example.  The unit of flux is
#   /(m2 sec sr Gev)
#-------------------------------------------------------------
  'p'   'GeV'     'KE/n'  'd'     0   /
        1.0     650
        1.5     500
        2.0     350
        3.0     190
        4.0     120
        5.0     80
        10      20
        20      4.0
        50      .41
        100     .08
        200     .013
        500.    1.2e-3
        1000.   2.e-4
        10.e3   3.17e-7
        100.e3  5.e-10
        200.e3  6.9e-11
        500.e3  5.07e-12
       1000.e3  6.34e-13
       10000e3  6.3e-16
       1.e9     4.0e-22
       1.e10    3.5e-25
        0.      0.
  'He'  'GeV'   'KE/n'  'd'     0  /
        1.0     50
        1.5     40
        2.      26
        3.      12
        5.      4.8
        10.     1.
        20.     .2
        50.     .02
        100.    3.7e-3
        200.    5.e-4
        400.    7.7e-5
        2.e3    9.98e-7
        20.e3   1.99e-9
       200.e3  3.97e-12
       400.e3   6.1e-13
       800.e3  8.46e-14
      8000.e3   8.46e-17
      8.e8      5.3e-23
         0        0
```

param file

DepthList: 観測大気深さのリスト（kg/m^2）
HeightList: 観測高度のリスト（m）

DepthListが優先される。HeightListを使いたい時は、対応するDepthListの数値を負にしておく。

InitRN: 乱数seed。二番目が負の時は計算機の時刻を利用。

PrimaryFile: 入射粒子情報を指定した primary fileのファイル名を指定

CosZenith, Azimuth: 入射天頂角（の cosine）と方位角の範囲を指定

DestEventNo: 2番目の数が入射粒子数を指定

最低限これだけ知っていれば、いろいろなことができる。

```
bash-3.2$ cat param
 &PARAM
 LatitOfSite = 30.110000
 LongitOfSite = 90.53
 YearOfGeomag = 2019.500

 DepthList =  2000 4000 6000 8000 0
! DepthList = -3000 -4000 -6000 -10000 0
! HeightList =  6000.0 4000.0 2000.0 .0 .0 .0 .0 .0
 ASDepthList = 2000 4000.0 6000.0 8000.0  .0 .0 .0 .
! ASDepthList = -10000  -4000.0 -2000.0 -1000.0
! ASHeightList =  6000.0 4000.0 2000.0 .0 .0 .0 .0

 SeedFile = 'Seed'
 InitRN = 300798  -3319907
 PrimaryFile = 'primary'
! PrimaryFile = 'primary_e'
! CosZenith = (1.0, 1.0)
 CosZenith = (0.9, 0.9)
 Azimuth = (0.0,0.0)
 HeightOfInj = 100.0e3
 DestEventNo = 1000 2

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

観測地の緯度経度・地磁気計算のための時間

param file (2)

Generate: “as”を指定すると電磁シャワーはB近似で置き換え。縦発達だけを高速で計算したい時、ハドロン反応だけを追いたい時、は有効。”em”は電磁シャワーを完全追跡（default）。”em/as”は両方実施。

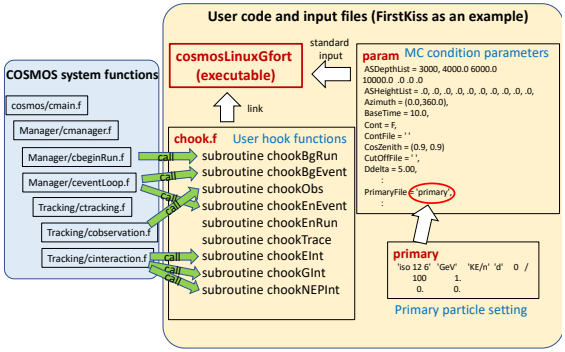
IntModel: ハドロン反応モデル。右の例は <2GeVで PHITSを、>2GeVで DPMJET3を利用。（切り替え方法、要マニュアル化）

Trace: Trace file出力の方法。全粒子のトラックを含む巨大な出力ファイルができるので、高エネルギー、複数入射の時はゼロにすること。トラック出力を自分で定義したい時は 100<=Trace<160とし、chookTraceを自分で記述。
FirstKissでは Trace=21で、自動的に定型の trace fileが生成される。

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

 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 = 41    ! for display with Earth
 Trace = 21    ! for display with detector
 TraceDir = './'
 WaitRatio = 0.01
 Within = 99999
 Za1ry = 'cos 1'
 &END
```

chook.f



- 計算の要所要所でユーザーが書く
  subroutine chookXXXが自動的に呼ばれる。
- chook.f内のsubroutineとendの間と自分で書けば、好きな処理をできる。

```
bash-3.2$ cat chook.f
!  #define  ONLYELEC

#include "cmain.f"
#include "chookHybAS.f"
!!! #include "ctemplCeren.f" not needed now
!      If you would supply your own cmyEfield.f and/or cmyBfield.f
!      put your cmyEfield.f in this folder; to do so,
!      probably it's better to copy cmyEfield.f in $COSMOSTOP/cosmos/
!      here and modify it.  The file here will override the one
!      in $COSMOSTOP/cosmos/
!!! #include "cmyEfield.f" not needed now
!!! #include "cmyBfield.f" not needed now

!  ************************************ hook for Beginning of a Run
!  * At this moment, all (system-level) initialization for this run
!  * has been ended.  After this routine is executed, the system goes into the
!  * event creation loop.
!      *
      subroutine chookBgRun
      implicit none
#include "Zmanagerp.h"
#include "Ztrack.h"
#include "Ztrackv.h"
#include "Zobs.h"
#include "Zobsp.h"
#include "Zobsv.h"

      real(8):: oldv
      integer:: icon

!            namelist output
      call cwriteParam(ErrorOut, 0)
!            primary information
      call cprintPrim(ErrorOut)
!            observation level information
      call cprintObs(ErrorOut)
      call epResetEcrit(0, "Air", 81.0d-3, oldv, icon)
      write(0,*) 'icon=',icon,'Default Ecrit oldv(MeV)=', oldv*1000,
     * ' has been reset to ', 81, ' MeV'

      end

!     ************************************ hook for Beginning of  1 event
!     *  All system-level initialization for 1 event generation has been
!     *  eneded at this moment.
!     *  After this is executed, event generation starts.
!     *
      subroutine chookBgEvent
      end



!     ************************************ hook for observation
!     *  One particle information is brought here by the system.
!     *  All information of the particle is in aTrack
!     *
      subroutine chookObs(aTrack, id)
      use modCodeConv
!
!     Note that every real variable is in double  precision so
!     that you may output it in sigle precision to save the memory.
!     In some cases it is essential to put it in sigle (say,
```

subroutine

chook.f

chookObs(aTrack, id) : 粒子が指定観測高度に達した時に呼ばれる。Track型の構造体 aTrackに該当粒子の全ての情報が、idには サブルーチンが呼ばれた理由があり。

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.

```
!     ********************************** hook for observation
!     *  One particle information is brought here by the system.
!     *  All information of the particle is in aTrack
!     *
      subroutine chookObs(aTrack, id)
      use modCodeConv
!
!     Note that every real variable is in double  precision so
!     that you may output it in sigle precision to save the memory.
!     In some cases it is essential to put it in sigle (say,
!     for gnuplot).
!
      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

      integer::i
      integer:: pdgcode  ! ptcl code by PDG M.C
!
!     For id =2, you need not output the z value, because it is always
!     0 (within the computational accuracy).
!

      if(id .eq. 2) then
            call ccos2pdg(aTrack%p, pdgcode)

            write(*, '(3i3, i12, 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
     *           pdgcode,
     *           aTrack%p%fm%p(4)-aTrack%p%mass, ! KE in GeV.
     *           aTrack%pos%xyz%r(1), aTrack%pos%xyz%r(2) !  x, y in m
      endif

      end
```

FirstKiss出力（1）

```
      write(*, '(3i3, i12, 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
*           pdgcode,
*           aTrack%p%fm%p(4)-aTrack%p%mass, ! KE in GeV.
*           aTrack%pos%xyz%r(1), aTrack%pos%xyz%r(2) !  x, y in m
```

```
1  1  0          22     1.1939E-03      600.4          4720.
1  1  0          22     4.5868E-03      183.4          5600.
1  1  0          22     1.2021E-04     -120.8          5303.
1  8  0          14     8.1848E-02      8011.          2361.
2  8  0          14     8.1848E-02     1.1833E+04      3543.
3  8  0          14     8.1848E-02     1.4229E+04      4286.
4  8  0          14     8.1848E-02     1.6039E+04      4847.
1  7  0         -12     3.3552E-02      9544.          1630.
2  7  0         -12     3.3552E-02     1.4125E+04      2451.
3  7  0         -12     3.3552E-02     1.6999E+04      2966.
4  7  0         -12     3.3552E-02     1.9168E+04      3356.
1  8  0         -14     5.5767E-02     -1357.          1.1828E+04
2  8  0         -14     5.5767E-02     -2126.          1.7601E+04
3  8  0         -14     5.5767E-02     -2612.          2.1226E+04
4  8  0         -14     5.5767E-02     -2980.          2.3964E+04
```

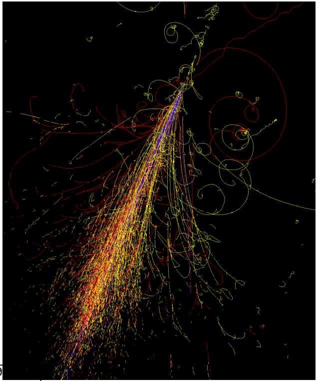
chookObs()による出力

- chookObs()で指定した標準出力と trace のふたつの出力
- 自分に必要な情報だけを出力すればいい

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
deuteron	kdeut	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	24
Λ	klambda	18	Λ_c	klambdac	21
Σ	ksigma	19	Ξ	kgzai	20
ω	komega	25	ϕ	kphi	26

FirstKiss出力（2）

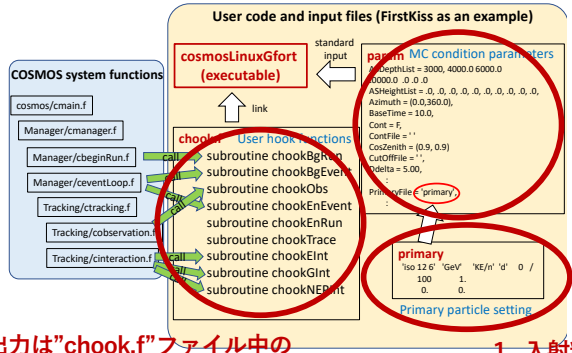
- chookObs()で指定した標準出力と traceのふたつの出力
- trace 出力(predefined)
- “x, y, z, ID, KE, Q”を、ひとつの粒子についてtrackが終わるまで羅列
- ひとつの粒子のtrackが終わったら空行で区切り
- これをひたすらつないで描いたので、ReadTraceMacro.Cの出力
- 100<=Trace<160として、subroutine chookTrace を書けば、自分の望みのtrace出力ができる。（全ての粒子の運動にアクセスできる。）



```
47435.23596485388     1.3009746114822280E-10   97941.50523180205     9   1400.000000000000       7   0.00000000000
41220.26220180405    -2.465814886252033        85097.43448453739     9   1399.999707199394       7   14269.31776835590


41220.26220180405    -2.465814886252033        85097.43448453739     2  3.8868093212764186E-04  -1   14269.31776835590
41219.59375602753    -0.3327319270908439       85096.32305271465     2  3.8867828324107160E-04  -1   14272.35099510635
41218.96074246645     1.818719704320210        85095.22640315845     2  3.8867563382288318E-04  -1   14275.38421292654
41218.35870980538     3.987413539006735        85094.14642198778     2  3.8867298388068228E-04  -1   14278.41742181541
41217.79767105777     6.183277481181099        85093.09981215891     2  3.8867033342665578E-04  -1   14281.45062177161
41217.29744589954     8.404239949093991        85092.07559459360     2  3.8866768247447579E-04  -1   14284.48381279322
41216.84737922833    10.64540761941022         85091.07236360318     2  3.8866503103470449E-04  -1   14287.51699487784
41216.44635539558    12.89460199171139         85090.06632055267     2  3.8866237911181411E-04  -1   14290.55016802505
41216.09650123827    15.15549861232135         85089.06746378010     2  3.8865972670694307E-04  -1   14293.58333223321
41215.76900706250    17.42386677835082         85088.07803790236     2  3.8865707382403536E-04  -1   14296.61648749893
41215.48735192407    19.70006346528575         85087.09251406517     2  3.8865442046644725E-04  -1   14299.64963382068
41215.27110632349    21.98350413237249         85086.10725008935     2  3.8865176663535909E-04  -1   14302.68277119762
41215.12595759560    24.27373365142514         85085.12473425407     2  3.8864911233165225E-04  -1   14305.71589962959
41215.02362945374    26.56446143632382         85084.13802200419     2  3.8864645755495180E-04  -1   14308.74901911471
41214.99253158825    28.85953115786967         85083.15662579949     2  3.8864380230574120E-04  -1   14311.78212965138
41215.01428372149    31.16134164846920         85082.19092156859     2  3.8864114658915825E-04  -1   14314.81523123866
```

COSMOS Xのまとめ



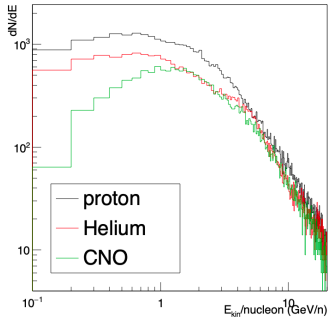
2. シミュレーションの条件は”param”ファイルで指定

3. 経過・結果の出力は”chook.f”ファイル中のchookXXXサブルーチンを書いて指定

1. 入射粒子は”primary”ファイルで指定

PrimaryHowToサンプル

- 入射粒子の情報を出力
- スタックを初期化する => 全ての粒子情報を消去 => シャワー計算が終了し次の入射粒子に
- primary fileで指定した mixed composition, spectrumがどう実現しているかを確認可能




```
      subroutine chookBgRun
      end

!     ********************************** hook for Beginning of  1 event
!     *  All system-level initialization for 1 event generation has been
!     *  eneded at this moment.
!     *  After this is executed, event generation starts.
!     *
      subroutine chookBgEvent

      implicit none
#include "Ztrack.h"

      type(coord):: angle
      type(track):: incident

      call cqIncident(incident, angle)

!      write(0,*) '1ry c,subc,chg, TE   cos-zenith '
      write(*,'(A, I3, I3, I3, F12.1, F8.3)') "PRIM ",
     *   incident%p%code, incident%p%subcode, incident%p%charge,
     *     incident%p%fm%p(4), incident%vec%coszenith

!     Kill all particles in stack
      call cinitStack

      end
```

FORTRAN初心者むけサンプル

- https://www.dropbox.com/s/i4uai25smqbsuwl/Lateral.tar.gz?dl=0
- Lateralというサンプル。シャワー中の粒子種別の lateral分布を chook.f内で計算して出力

param file

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

! DepthList =  2000 4000 6000 8000 0
 DepthList = -1 0
 HeightList = 4000.0 .0
! ASDepthList = 2000 4000.0 6000.0 8000.0  .0 .0 .0 .0 .0 .0 .0 .0
! ASDepthList = -10000  -4000.0 -2000.0 -1000.0
! ASHeightList =  6000.0 4000.0 2000.0 .0 .0 .0 .0

 SeedFile = 'Seed'
 InitRN = 300798  -3319907
 PrimaryFile = 'primary'
! PrimaryFile = 'primary_e'
! CosZenith = (1.0, 1.0)
 CosZenith = (0.9, 0.9)
 Azimuth = (0.0,0.0)
 HeightOfInj = 100.0e3
 DestEventNo = 1000 2
```

DepthListを使わないのでマイナス

```
      subroutine chookEnEvent
      use myhist

      do i=1,nbin
         write(*,'(i4, 5i10)') i, (hpart(j,i),j=1,nid)
      end do

      write(*,'(A)') ' '

      end
```

サブルーチン間で共有する変数の定義

粒子種別粒子数のヒストグラム配列変数

primary file, 5TeV proton

```
#
#   'e-'  'MeV'     'KE/n'  'd'     0   /   24-12 Mg  22-11 Na
#
#-------------------------------------------------------------
  'proton'  'TeV'     'KE/n'  'd'     0   /
          5         1.
           0.              0.
```

```
PRIM    6 -1  1       5000.9    0.903
BEGIN_EVENT
    1       162        38         0         0         0
    2       505       123         0         0         0
    3       455        97         0         0         0
    4       311        53         0         0         0
    5       367        49         0         0         0
    6       387        76         2         0         0
    7       323        54         3         0         0
    8       348        42         1         0         0
    9       336        47         0         0         0
   10       307        25         0         0         0
   11       281        37         0         0         0
   12       297        40         0         0         0
   13       269        28         0         0         0
   14       279        21         0         0         0
   15       242        21         2         0         0
   16       265        22         1         0         0
   17       245        19         1         0         0
   18       231        17         1         0         0
```

基本設計

- シミュレーション(RUN)開始時に粒子別、距離別の計数用2次元整数配列変数を定義。異なるサブルーチン間で配列変数を共有 **module**
- 粒子入射時(EVENT)に、配列の中身をゼロにする。入射粒子情報を出力。 **chookBgEvent**
- 粒子が観測高度に達したら、粒子種、距離別に配列のカウントを増加 **chookObs**
- ひとつのシャワー（EVENT）が終了したら、結果を出力 **chookEnEvent**

共通変数の定義 module、EVENTの開始

```
module myhist
implicit none
integer, parameter::nid = 5
integer, parameter::nbin = 100
real, parameter::rstep = 10.0  ! bin step in meter
integer, dimension(nid,nbin)::npart
end module myhist
```

- Moduleで定義しmodule名（ここではmyhist）をつけた変数は、myhistを指定することでどのサブルーチンでも利用可能
- ヒストグラムを定義するには、ビン数(nid, nbin)とビン幅（rstep）の定義が必要（粒子種は整数なのでビン幅は定義せず）
- FORTRANでは配列は1から始まる（Cではゼロから）

```
      subroutine chookBgEvent
      use myhist
#include "Ztrack.h"

      type(coord):: angle
      type(track):: incident

      call cqIncident(incident, angle)

!      write(0,*) '1ry c,subc,chg, TE   cos-zenith '
      write(*,'(A, I3, I3, I3, F12.1, F8.3)') "PRIM ",
     *   incident%p%code, incident%p%subcode, incident%p%charge,
     *     incident%p%fm%p(4), incident%vec%coszenith

      npart = 0  ! npart(i,j) = 0 (i = 1,..., j = 1,...)

      write(*,'(A)') 'BEGIN_EVENT'

      end
```

chookBgEvent: 入射粒子情報と配列の初期化

構造体 (type)

CosmosX_0.08/Cosmos/cosmos/Ztrack.h

```
!   #ifndef Ztrack_
!   #define Ztrack_
!    structure used when tracking a particle
!    ************************
#include  "Zcondc.h"
#include  "Zptcl.h"
#include  "Zcoord.h"
#include  "Zpos.h"
#include  "Zdirec.h"
#include  "Zmagfield.h"
!     ----------------------
      type track      ! full particle attributes in Cosmos
        sequence

          type(ptcl):: p    ! basic ptcl attributes.

!               position and time
          type(position):: pos
          real*8 t              ! time in length/beta (m)
          type(direc):: vec
          real*4  wgt           ! weight for thin sampling
           integer*2 where    ! current obsSite no. (0 is initial value)
          integer*2 asflag   ! non 0, if As has been generated from this
!                             ptcl (only for electrons)
          real*8   user      ! user use

! particle information at production
          type(ptcl):: inip    ! basic ptcl attributes.
          type(position):: inipos
          real*8 init           ! time in length/beta (m)
          type(direc):: inivec

! parent particle information
          type(ptcl):: parp    ! basic ptcl attributes.
          type(direc):: parvec

#if LABELING > 0
          integer  label     ! 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.
           integer  info       ! 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.
#endif
      end type track
!   #endif  /* Ztrack.h */
```

CosmosX_0.08/LibLoft/Header/Zptcl.h

```
!     ******************************************************
      type ptcl       ! particle at production
        sequence
!                 4 momentum.

      type(fmom):: fm
!
          real*8 mass
          integer*2 code, subcode
          integer*2 charge
!       code: ptcl code
!     subcode: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
      end type ptcl
!     ******************************************************
```

配列の加算

- 追跡粒子が指定高度に達したらchookObs()が呼ばれる
- 粒子の種類が photonなら1、電子陽電子なら2、ミューオンなら3とする
- Moduleで定義した距離のビン幅 rstepを利用して、粒子のコアからの距離が何ビン目に当たるか計算
- 該当ビンの粒子数を +1

```
!     ********************************** hook for observation
!     *  One particle information is brought here by the system.
!     *  All information of the particle is in aTrack
!     *
      subroutine chookObs(aTrack, id)
      use modCodeConv
      use myhist
!
!     Note that every real variable is in double  precision so
!     that you may output it in sigle precision to save the memory.
!     In some cases it is essential to put it in sigle (say,
!     for gnuplot).
!
      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

      integer::i
      integer:: pdgcode         ! ptcl code by PDG M.C

      real x, y, radius
      integer ibin, pid
!
!     For id =2, you need not output the z value, because it is always
!     0 (within the computational accuracy).
!


      pid = 0

      if(id .eq. 2) then
            call ccos2pdg(aTrack%p, pdgcode)
            if ( pdgcode.eq.22 ) pid = 1   ! photon
            if ( (pdgcode.eq.11).or. (pdgcode.eq.-11) ) pid = 2  !e+/-
            if ( (pdgcode.eq.13).or. (pdgcode.eq.-13) ) pid = 3  !mu+/-

            x = aTrack%pos%xyz%r(1)
            y = aTrack%pos%xyz%r(2)
            radius = sqrt(x*x + y*y)
            ibin = int(radius/rstep) + 1

            if ((ibin.ge.1).and.(ibin.le.nbin).and.(pid.ne.0)) then
               npart(pid,ibin) = npart(pid,ibin) + 1
            endif
         endif

      end
```

粒子番号の対応

5.1. 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

実践練習

- LateralをDLして動かしてみる
  - Application/Example/Lateral/… とする
  - ./Scrpt/CompileExampleByCMake.sh Application/Example/Lateral
  - cd Application/Example/Lateral
  - ./cosmosLinuxGfort < param > out
  - outの出力結果確認
- 条件を変えてみる
  - Neutrinoや hadronも数えてみよう (PDGって？)
  - 入射粒子 protonを gammaにしてみよう
  - 入射エネルギーを変えてみよう
  - R方向分布ではなく、x,y分布(2次元)にしてみよう
  - 1 eventごとではなく、全 eventの合計を出してみよう
  - 複数の高度で lateral分布を作ってみよう
    ：

明日の発表