FLUKA

蘭州大學
LANZHOU UNIVERSITY

Highlights on Advanced Topics

23rd FLUKA Beginner's Course
Lanzhou University
Lanzhou, China
June 2-7, 2024

This is not the end! Only the beginning …

- As said on the first day, most applications require data-cards only, exploiting the FLUKA built-in capacities
- Sometimes, more is needed…
- Many (more advanced) features not presented in this course
- Template User Routines are provided in $FLUPRO/usermvax
- Routines can be modified by the user to fit his input/output needs
- All shared parameters and variables (in COMMON blocks) are located in $FLUPRO/flukapro
- We'll give here some hints of what can be done, more can be found in the manual, in the fluka-discuss archive, or…

at the next FLUKA advanced course and workshop

Dates and venue not yet defined

Advanced geometry

Geometry directives

Special commands enclosing body definition:

$Start_xxx

.....

$End_xxx

where "xxx" stands for
“expansion“, "translat" or "transform"

They provide respectively a coordinate expansion/reduction, a coordinate translation or a coordinate roto-translation of the bodies embedded between the starting and the ending directive lines.

$start_transform	Trans:		
$end_transform			
$start_expansion	f:		
$end_expansion			
$start_translat	dx:	dy:	dz:
$end_translat			

Parentheses

Parentheses are grouping together combinations of bodies. Parentheses can be used in name based format only.

Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
* Subtract from body2 regions regR03, regR04, regR05


regV02 5  +body2 - (+body4  -body3)
          - (+body6  -body5 |+body8 -body7) - body9 -body10
regR03 5 +body4  -body3
regR04 5 +body6  -body5 |+body8 -body7
regR05 5 +body9 | +body10
```

Nested parentheses are supported, however:

- parentheses should be used with care since their expansion can generate a quickly diverging amount of terms.

- A partial optimization is performed on planes (aligned with the axes) and bounding boxes only

Lattice

FLUKA geometry has *replication* (lattice) capabilities

Only one *level is implemented* (no nested lattices are allowed)

- The user defines lattice positions in the geometry and provides transformation rules from the lattice to the prototype region:
  1. in the input with the ROT-DEFI card (see later)
  2. in a subroutine (lattic.f)

The lattice identification is available for scoring

Transformations include:
Translation, Rotation and Mirroring (the last only through routine).

WARNING:

Do not use scaling or any deformation of the coordinate system

Lattice

- The regions which constitute the elementary cell *(prototype)* to be replicated, have to be defined in detail
- The Lattices *(replicas/containers)* have to be defined as "empty" regions in their correct location.
  WARNING: The lattice region should map exactly the outer surface definition of the elementary cell.
- The lattice regions are declared as such with a LATTICE card at the end of the geometry input
- In the LATTICE card, the user also assigns lattice names/numbers to the lattices. These names/numbers will identify the replicas in all FLUKA routines and scoring
- Several basic cells and associated lattices can be defined within the same geometry, one LATTICE card will be needed for each set
- Non-replicas carry the lattice number **0**
- Lattices and plain regions can coexist in the same problem

LATTICE card

After the Regions definition and before the GEOEND card the user can insert the LATTICE cards

- WHAT(1), WHAT(2), WHAT(3)
  Container region range (from, to, step)
- WHAT(4), WHAT(5), WHAT(6)
  Name/number(s) of the lattice(s)
- SDUM
  blank — to use the transformation from the lattic routine
  ROT#nn — to use a ROT-DEFI rotation/translation from input
  name — the same as above but identifying the roto-translation by the name assigned in the ROT-DEFI SDUM (any alphanumeric string you like)

Example

LATTICE	Reg: TARGR1 ▼	to Reg: ▼	Step:
Id: 1tra ▼	Lat: 1.0	to Lat: 1.0	Step: 1.0

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
LATTICE       6.00000  19.00000              101.0000    114.00
```

Region # 6 to 19 are the "placeholders" for the first set replicas. We assign to them lattice numbers from 101 to 114

```
LATTICE        TARGR1                          TargRep                    1tra
```

TARGR1 is the container region using transformation *1tra*

ROT-DEFIni

The ROT-DEFIni card defines roto-translations that can be applied, in addition to bodies, to i.) USRBIN & EVENTBIN and ii.) LATTICE. It transforms the position of the tracked particle i.) before scoring with respect to the defined binning or ii.) into the prototype with the order:

- First applies the translation
- followed by the rotation on the azimuthal angle
- and finally by the rotation on the polar angle.

$$X_{new} = M_{polar} \times M_{az} \times (X + T)$$

WHAT(1): assigns a transformation index and the corresponding rotation axis **I + J * 100** or **I * 1000 + J**

I = index of rotation (WARNING: NOTE THE SWAP OF VARIABLES)

J = rotation with respect to axis (1=X, 2=Y, 3=Z)

WHAT(2): Polar angle of the rotation ($0 \leq \vartheta \leq 180^o$ degrees)

WHAT(3): Azimuthal angle of the rotation ($-180 \leq \Phi \leq 180^o$ degrees)

WHAT(4), WHAT(5), WHAT(6) = X_{offset}, Y_{offset}, Z_{offset} offset for the translation

SDUM: name for the transformation

ROT-DEFI	Id: 1	Axis: Z ▼	Name: 1tra
	Polar: 0.0	Azm:	
	Δx:	Δy:	Δz: -10.0

How ROT-DEFI works:

θ = polar angle, Φ = azimuthal angle

The transformation matrix is:

$R_{ij} = T_{ik}P_{kj}$:

$$\begin{vmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{vmatrix} = \begin{vmatrix} \cos\phi \times \cos\theta & \sin\phi \times \cos\theta & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \\ \cos\phi \times \sin\theta & \sin\phi \times \sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} X_{old} + X_{offset} \\ Y_{old} + Y_{offset} \\ Z_{old} + Z_{offset} \end{vmatrix}$$

Auxiliary routines – Name <-> number conv

Conversion of region name to number

```
CALL GEON2R ( REGNAM, NREG, IERR )
```

Input variable:

REGNAM = region name (CHARACTER*8)

Output variables:

NREG = region number

IERR = error code (0 on success, 1 on failure)

Conversion of region number to name

```
CALL GEOR2N ( NREG, REGNAM, IERR )
```

Input variable:

NREG = region number

Output variables:

REGNAM = region name (CHARACTER*8)

IERR = error code (0 on success, 1 on failure)

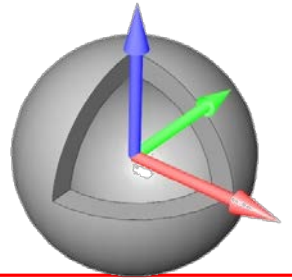
Volume sources

Extended sources – *Volume sources*

Input card: BEAMPOS

If **SDUM** = **SPHE-VOL**:

defines a spatially extended source in a spherical shell



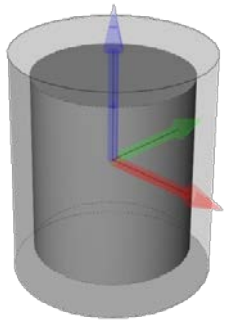
Example

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAMPOS          0.0       0.0       0.0       0.0       0.0       0.0
BEAMPOS          0.0       1.0       0.0       0.0       0.0       0.0SPHE-VOL
```

If **SDUM** = **CYLI-VOL**:

defines a spatially extended source in a cylindrical shell

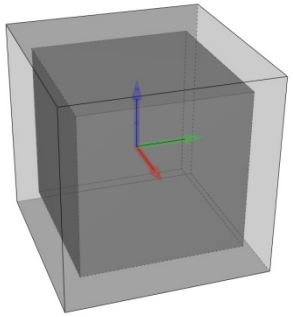
with the height parallel to the z-axis of the beam frame



If **SDUM** = **CART-VOL**:

defines a spatially extended source in a Cartesian shell

with the sides parallel to the beam frame axes



Extended sources – *Spherical surface uniform isotropic source*

Input card: BEAMPOS

If **SDUM** = **FLOOD**:

defines a source distribution on a spherical surface

Example

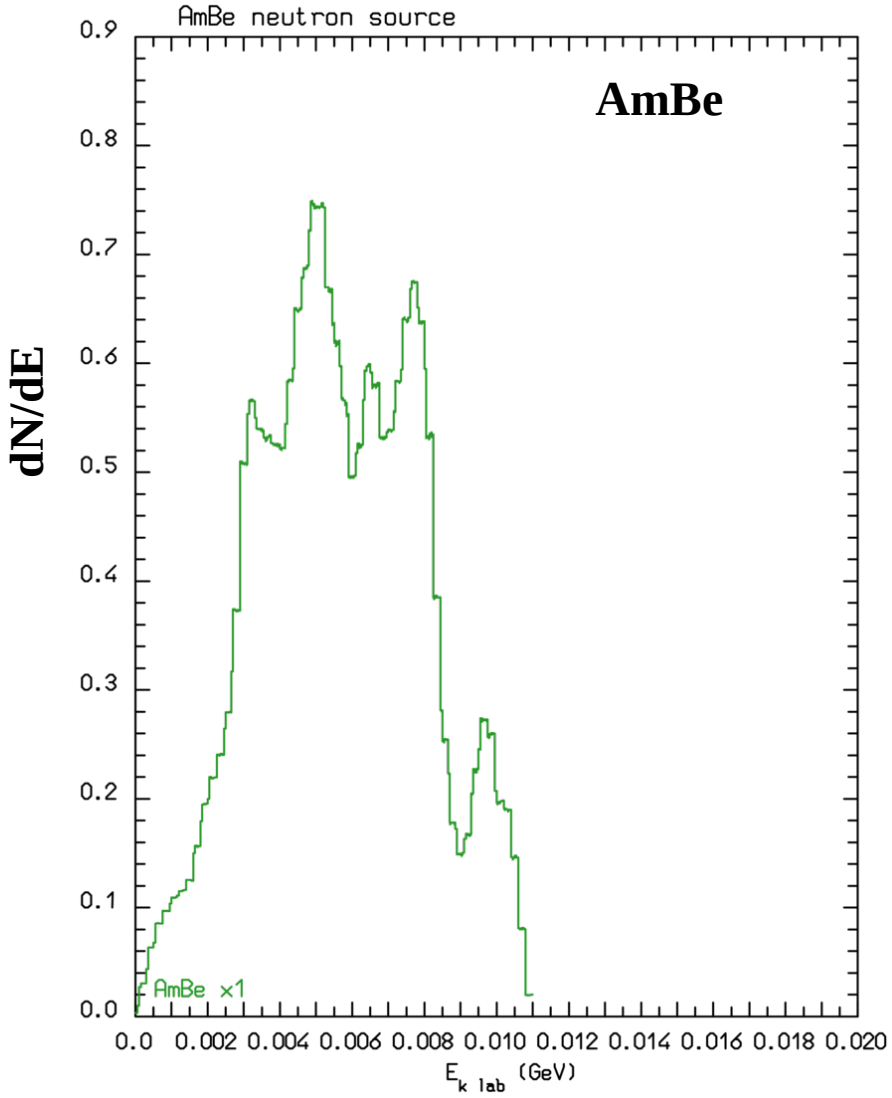
```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAMPOS          0.0       0.0       0.0       0.0       0.0       0.0
BEAMPOS          1.0       0.0       0.0       0.0       0.0       0.0FLOOD
```

- radius (in cm) of the sphere: 1.0 cm [**WHAT(1)**]
- **WHAT(2)** - **WHAT(6)** are not used !

The surface is centred at the (x,y,z) point defined by another BEAMPOS card with **SDUM** = blank (or = **NEGATIVE**). The particle direction is sampled according to a diffusive distribution so as to generate a uniform and isotropic fluence equal to $1/(\pi R^2)$ inside the sphere (in absence of materials)

Built-in neutron sources

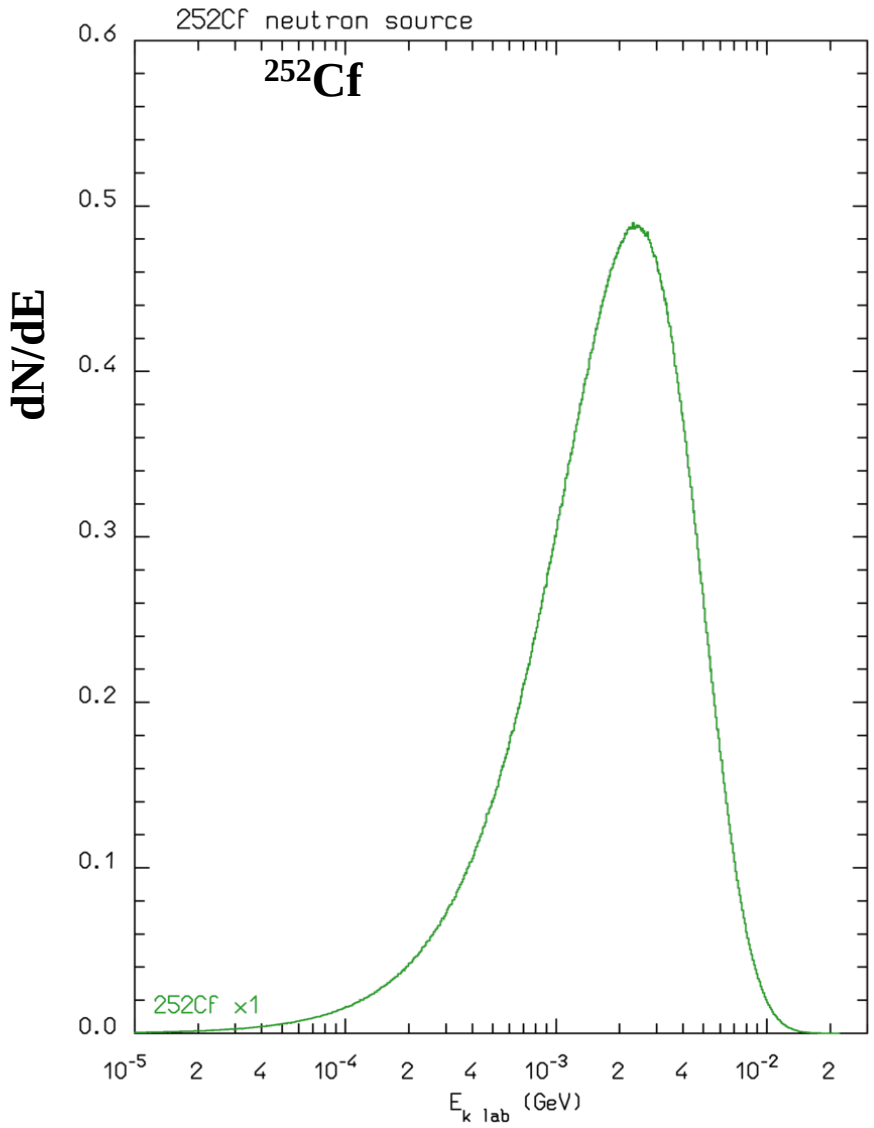
Built-in neutron sources: AmBe, AmB, ^{252}Cf, D-D, D-T



With the BEAM card the following built-in neutron sources can be activated

- AmBe: SDUM=AMBE (left, ISO/DIS 8529-1)
- AmB : SDUM=AMB (ISO/DIS 8529-1)
- ^{252}Cf : SDUM=252CF (right, std. param.)
- D-D : SDUM=D-D
- D-T : SDUM=D-T

Please read the manual for further details!!!



Special source routines

Special source routines: SPECSOUR

Special pre-set source routines are invoked setting a card:

SPECSOUR	**inp_1**	**inp_2**	**inp_3**	**inp_4**	**inp_5**	**inp_6**	**SDUM**

SPECSOUR	**inp_7**	**inp_8**	**inp_8**	**inp_{10}**	**inp_{11}**	**inp_{12}**	**&**
SPECSOUR	**inp_{13}**	**inp_{14}**	**inp_{15}**	**inp_{16}**	**inp_{17}**	**inp_{18}**	**&&**

Where SDUM can be:

- **BEAMSPOT** : using predefined beamlets (demonstrated yesterday)
- **PPSOUR** : colliding beams
- **GCR-AMS** : GCR spectra
- **SPE-2003** : Solar Particle Event (2003)
- **SPE-2005** : Solar Particle Event (2005)
- **BIN-SOUR** : USRBIN-like distributed source (eg activity 3D of a radiopharmaceutical)
- **SYNC-RAD** : Synchrotron radiation source (spare slides of the EM lecture)

Each special source can accept up to 18 different parameters, their meanings are different for each case and are explained in the manual

Special source routines: SPECSOUR … BEAMSPOT

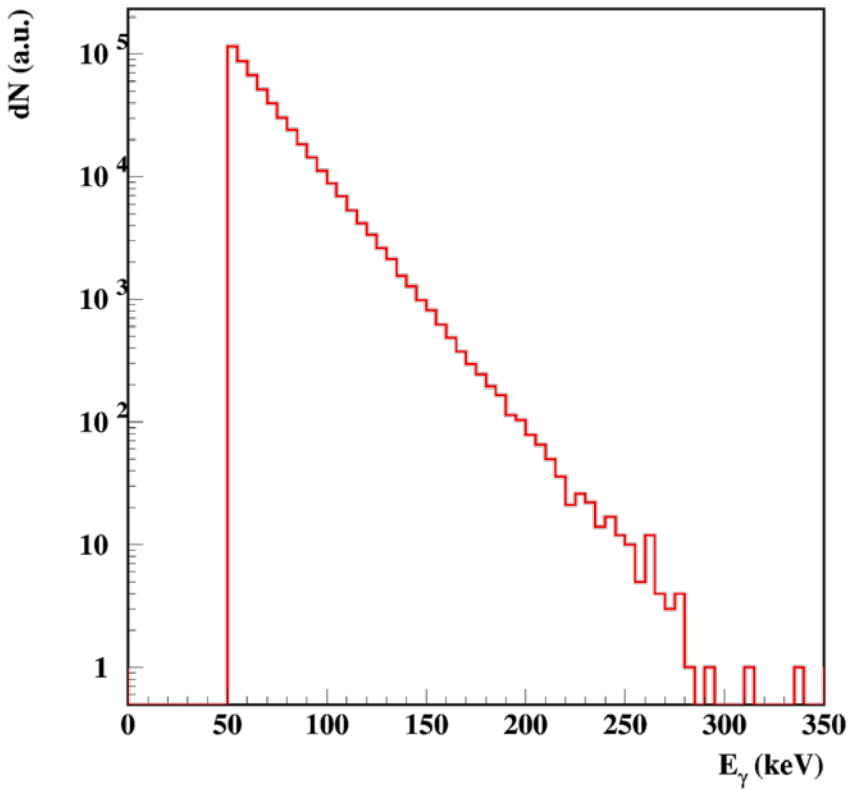
Special options to define “beamlets” eg for hadrontherapy:

- **SPOTBEAM** : it defines a beamlet energy/projectile, energy and angular spread
- **SPOTDIR** : it defines a beamlet direction, and the beamlet reference frame
- **SPOTPOS** : it defines a beamlet position and σ_x, σ_y
- **SPOTTIME** : it defines a beamlet time
- **SPPOTTRAN**: it defines a possible roto-translation to be applied to a beamlet

Generic source routine

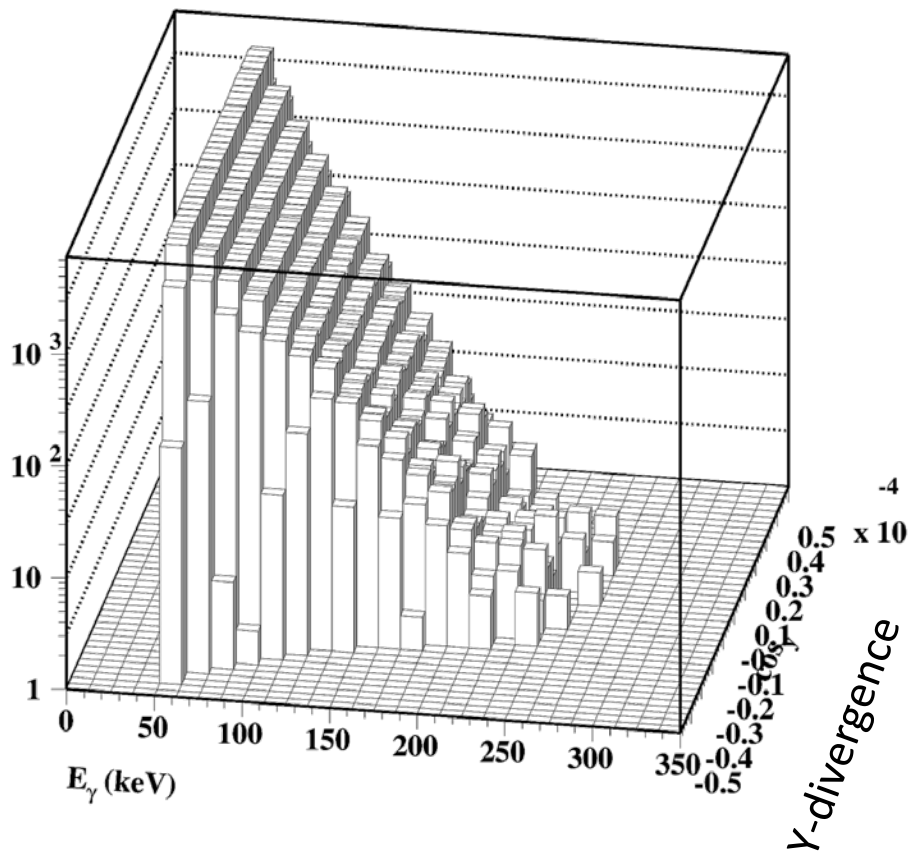
Implementing customized beam distributions

Example: synchrotron radiation spectrum read from external file with energy, x and y divergences, and polarization all correlated



← Energy spectrum

Y-divergence correlated → with energy



Implementing customized beam distributions - 2

Input card: SOURCE Template user routine: $FLUPRO/usermvax/source.f

source.f must be linked in one's own executable !

It is possible to sample beam particle position, direction, and energy from an external file or a distribution

It is possible to assign different weights to primary particles

It is possible to load reaction products in the same primary history

several sampling routines already exist in the FLUKA code (e.g. Gaussian)

Input parameters can be passed via the SOURCE card

a BEAM card with a momentum/energy higher than the maximum one is still needed for initialization purposes (to define the tabulation limit)

A look into the source routine prototype;


```
      NPFLKA = NPFLKA + 1
*  Wt is the weight of the particle
      WTFLK  (NPFLKA) = ONEONE
      WEIPRI = WEIPRI + WTFLK (NPFLKA)
*  Particle type (1=proton.....). Ijbeam is the type set by the BEAM
*  card, the numbering scheme is the Paprop (user) one:
*  +-------------------------------------------------------------------*
*  |  (Radioactive) isotope:
      IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
         …
*  |
*  +-------------------------------------------------------------------*
*  |  Heavy ion:
      ELSE IF ( IJBEAM .EQ. -2 ) THEN
         …
*  |
*  +-------------------------------------------------------------------*
*  |  Normal particle:
      ELSE
         IONID = IJBEAM
*  |  Possible excitation energy wrt ground state (0 in this case):
         EEXSTK (NPFLKA) = EXENRG (IJBEAM)
*  |  Possible mean life of an excited state (meaningless in this case):
         TMNSTK (NPFLKA) = TMNLF  (IJBEAM)
*  |  Number (see Paprop) of a possible excited state (0 in this case):
         ILVSTK (NPFLKA) = IEXLVL (IJBEAM)
*  |  Particle id:
         ILOFLK (NPFLKA) = IJBEAM
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
*  |  Group number for "low" energy neutrons, set to 0 anyway
         IGROUP (NPFLKA) = 0
*  |  Parent radioactive isotope:
         IRDAZM (NPFLKA) = 0
*  |  Particle age (s)
         AGESTK (NPFLKA) = +ZERZER
```

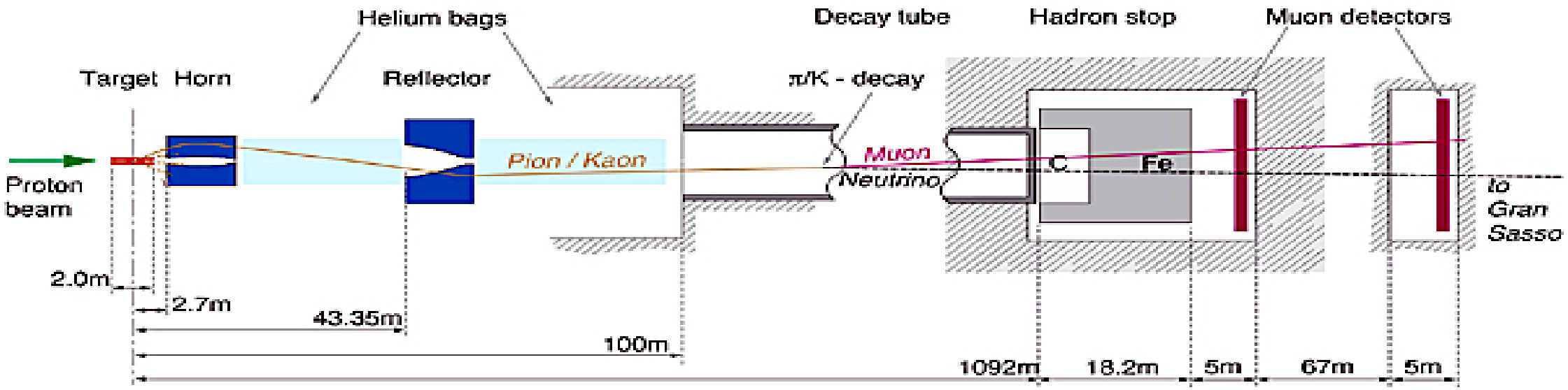
```
*  |  Kinetic energy of the particle (GeV)
         TKEFLK (NPFLKA) = PBEAM**2
     &                 / ( SQRT ( PBEAM**2 + AM (IONID)**2 ) + AM (IONID))
*  |  Particle momentum
         PMOFLK (NPFLKA) = PBEAM
*        PMOFLK (NPFLKA) = SQRT ( TKEFLK (NPFLKA) * ( TKEFLK (NPFLKA)
*    &                           + TWOTWO * AM (IONID) ) )
*  |  +-----------------------------------------------------------------*
*  |  |  Check if it is a neutrino, if so force the interaction
*  |  |  (unless the relevant flag has been disabled):
         IF ( LISNUT (IJBEAM) .AND. LNUFIN ) THEN
            LFRPHN (NPFLKA) = .TRUE.
*  |  |
*  |  +-----------------------------------------------------------------*
*  |  |  Not a neutrino:
         ELSE
            LFRPHN (NPFLKA) = .FALSE.
         END IF
*  |  |
*  |  +-----------------------------------------------------------------*
      END IF
*  |
```

```
•  +-------------------------------------------------------------------*
*  Direction cosines (tx,ty,tz):
      TXFLK  (NPFLKA) = UBEAM
      TYFLK  (NPFLKA) = VBEAM
      TZFLK  (NPFLKA) = WBEAM
*     TZFLK  (NPFLKA) = SQRT ( ONEONE - TXFLK (NPFLKA)**2
*    &                        - TYFLK (NPFLKA)**2 )
*  Polarization cosines:
      TXPOL  (NPFLKA) = -TWOTWO
      TYPOL  (NPFLKA) = +ZERZER
      TZPOL  (NPFLKA) = +ZERZER
*  Particle coordinates:
      XFLK   (NPFLKA) = XBEAM
      YFLK   (NPFLKA) = YBEAM
      ZFLK   (NPFLKA) = ZBEAM
```

Implementing magnetic and electric fields

Implementing magnetic field - 1

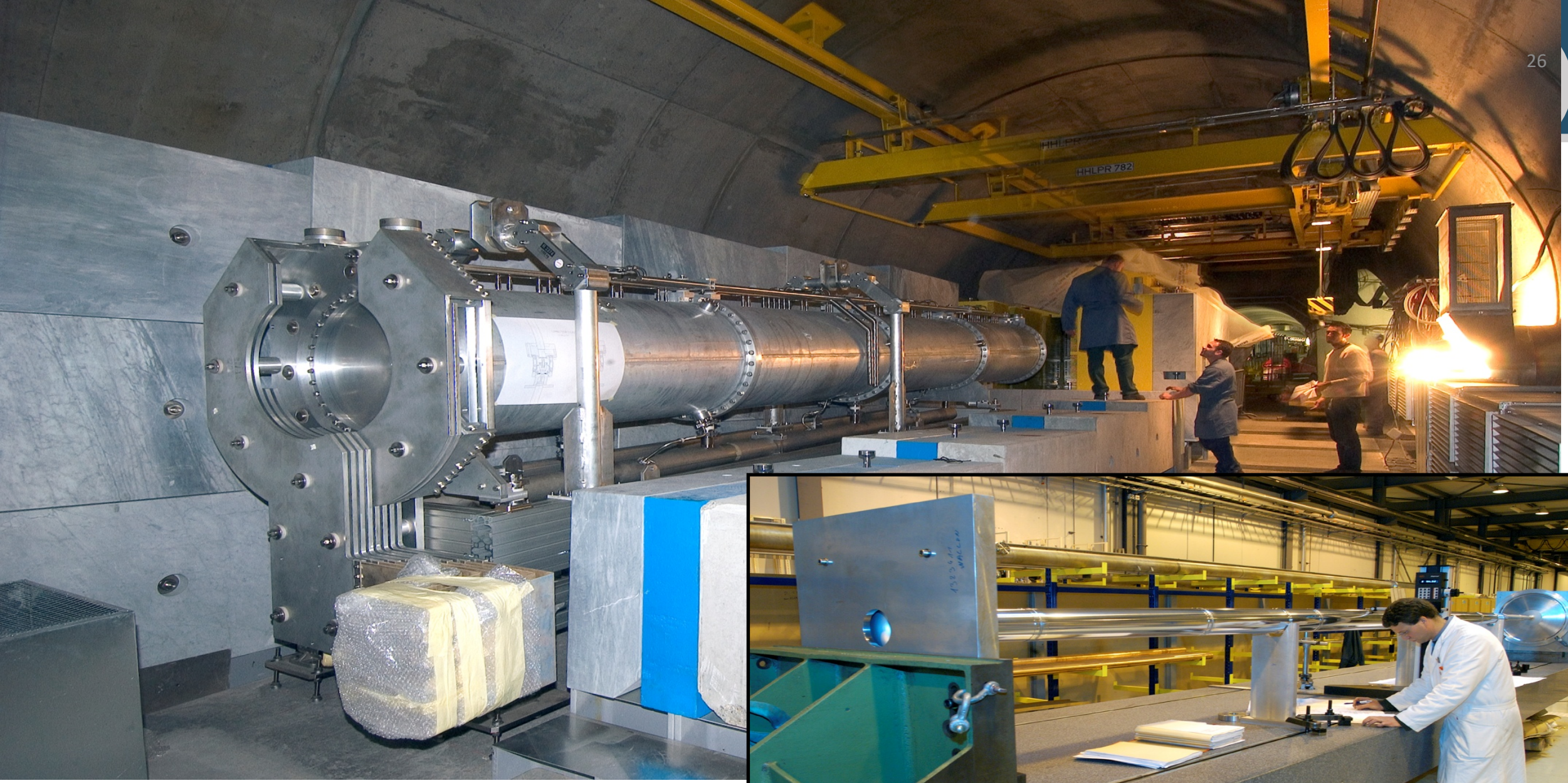
CERN Neutrino to Gran Sasso

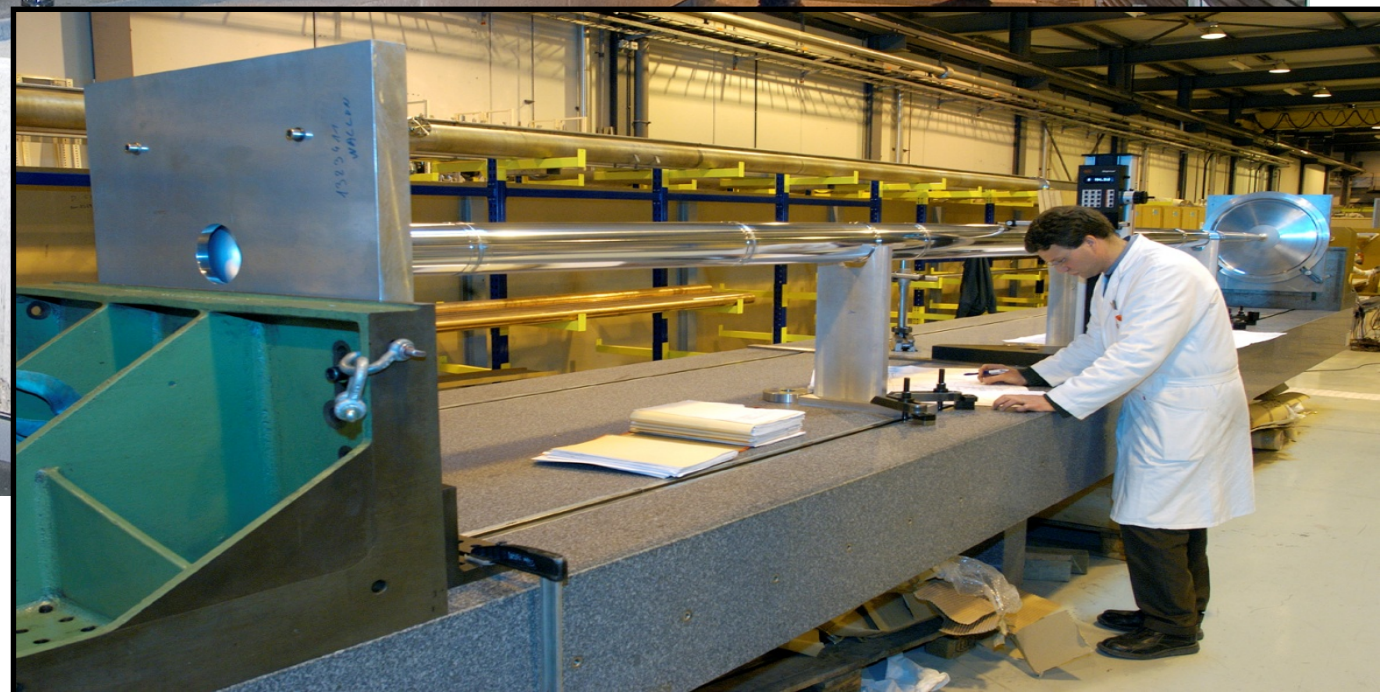


The two magnetic lenses (blue in the sketch) align positive mesons towards the Decay tunnel, so that neutrinos from the decay are directed to Gran Sasso, 730~km away

Negative mesons are deflected away

The lenses have a finite energy/angle acceptance


HHLPR 782



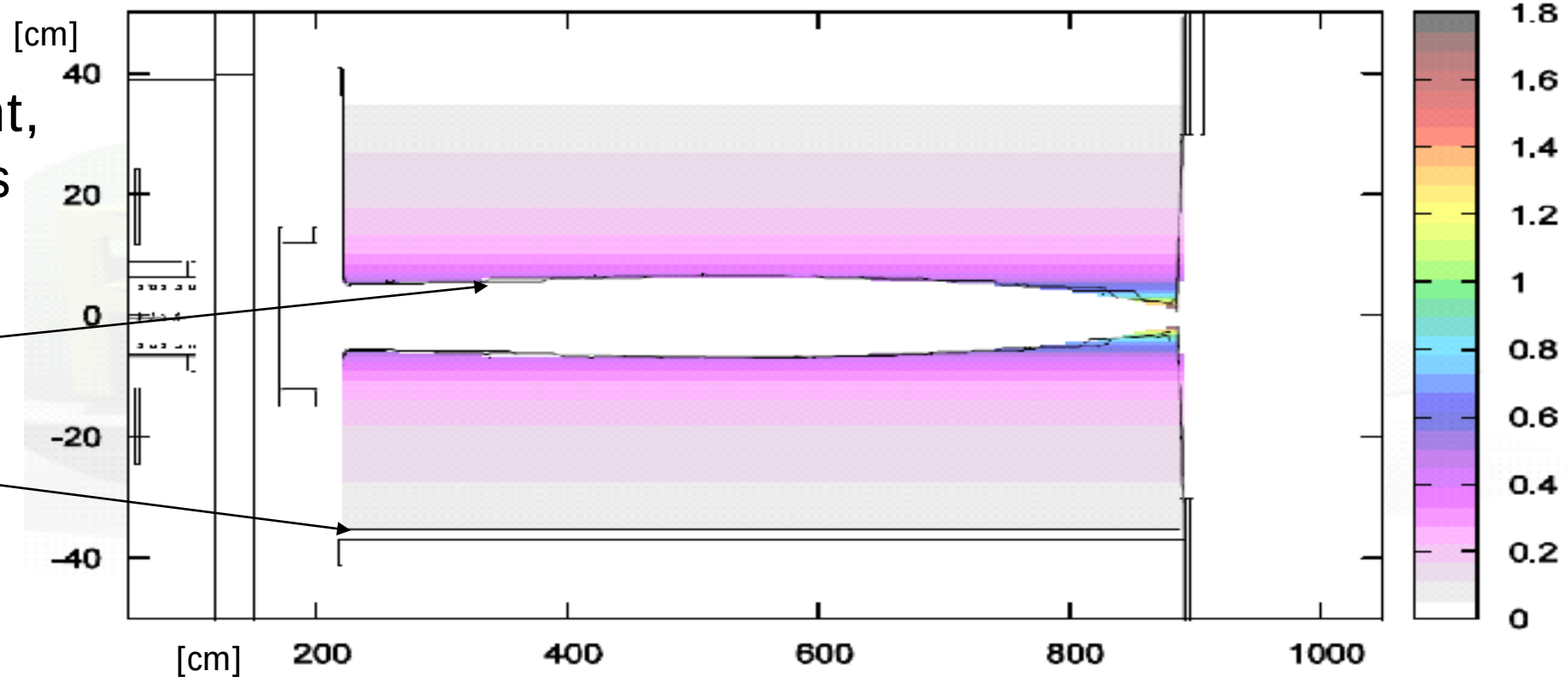
Implementing magnetic field - 2

Input card: ASSIGNMA Template user routine: $FLUPRO/usermvax/magfld.f

magfld.f must be linked in one's own executable !

Magnetic field intensity in the CNGS horn

A ≈150kA current, pulsed, circulates through the Inner and Outer conductors The field is toroidal, B÷1/R

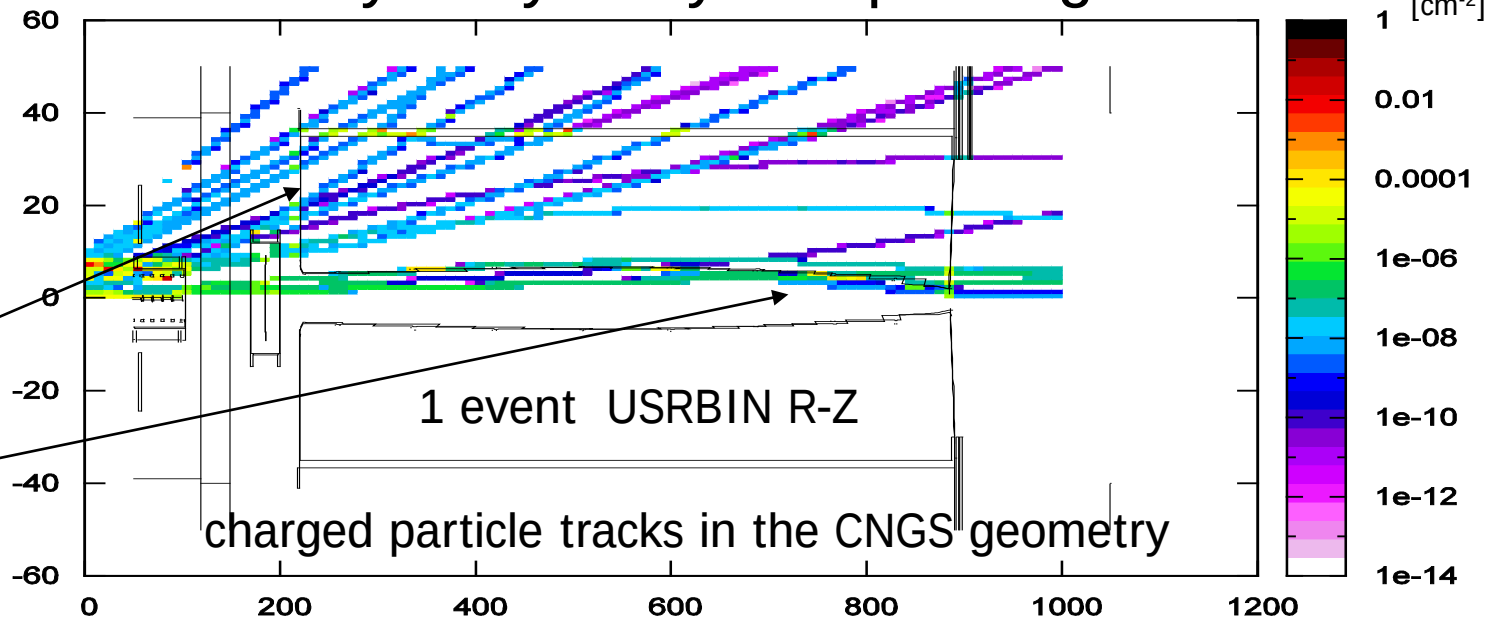


Implementing magnetic field - 3


```
SUBROUTINE MAGFLD ( X, Y, Z, T, BTX, BTY, BTZ, B, NREG, IDISC )
 …
IF ( NREG .EQ. NRHORN ) THEN
      RRR = SQRT ( X**2 + Y**2 )
      B   = 2.D-07 * CURHOR / 1.D-02 / RRR
      BTX =-Y / RRR
      BTY = X / RRR
      BTZ = ZERZER
END IF
```

X,Y,Z,T = current particle position and time (input)
Nreg = current region number (input)
BTX,Y,Z = magnetic field direction cosines (output)
B = magnetic field intensity (Tesla) (output)
IDISC = if set to 1 the particle is discarded (output)

Possible to define the magnetic field analytically or by interpolating an external map



De-focused
focused
Many Escaping

Implementing electric field


```
SUBROUTINE ELEFLD ( X, Y, Z, T, ETX, ETY, BTZ, E, NREG, IDISC )
…
RSPHER = 10.D+00
RRR = SQRT ( X**2 + Y**2 +Z**2)
IF ( RRR > RSPHER ) THEN
      E    = 1.D+00 * ( RSPHER / RRR )
      ETX = Y / RRR
      ETY = X / RRR
      ETZ = Z / RRR
ELSE
       E = 0.D+00
       ETX = 1.D+00
       ETY = 0.D+00
       ETZ = 0.D+00
END IF
```

Possible to define the electric field analytically or by interpolating an external map. In this case it is the E field generated by an uniformly charged conductor sphere of R=10 cm, with surface field = 1 MV/m

Very similar to the magnetic field case

Please take into account that electric fields are supported through Runge-Kutta-Gill 4th order tracking of the associated differential equations and that RKG is supported only in gas or vacuum regions

Please remember that electric/magnetic field can be also time varying!!

Miscellaneous

Access detailed informations

Input card: USERDUMP Template user routine: $FLUPRO/usermvax/mgdraw.f

mgdraw.f must be linked in one's own executable !

It is possible to get particle trajectories and (continuous and local) energy losses

It is possible to access information at each boundary crossing, particle step, energy deposition event, interaction

It is possible to look at reaction products

Implementing customized scoring

Input card: USERWEIG

Template user routines: $FLUPRO/usermvax/fluscw.f (for fluence scoring)

$FLUPRO/usermvax/comscw.f (for density-like scoring)

???scw.f must be linked in one's own executable !

It is possible to apply a user defined weight (even zero) to deposited energy, residual nuclei, etc.

It is possible to extract information about the particles involved (and dump it into a file)

Implementing customized "tags"

No input card

Template user routines: $FLUPRO/usermvax/stupre.f (for e.m. particles)

$FLUPRO/usermvax/stuprf.f (for other particles)

stupr?.f must be linked in one's own executable !

It is possible to keep track of particles properties like its origin

identification of ancestors of muon-neutrinos reaching Gran Sasso

ancestor as exiting the target

parent decaying in ν_μ

Implementing region independent importance biasing

Input card: BIASING Template user routine: $FLUPRO/usermvax/usimbs.f

usimbs.f must be linked in one's own executable !

It is call at every step in user-selected regions and allows to importance biasing of a particle independently from the region

It doesn't require to segment the geometry in many regions to use region importance biasing

It can be time-consuming in implementation and CPU-time

Thanks for your attention!