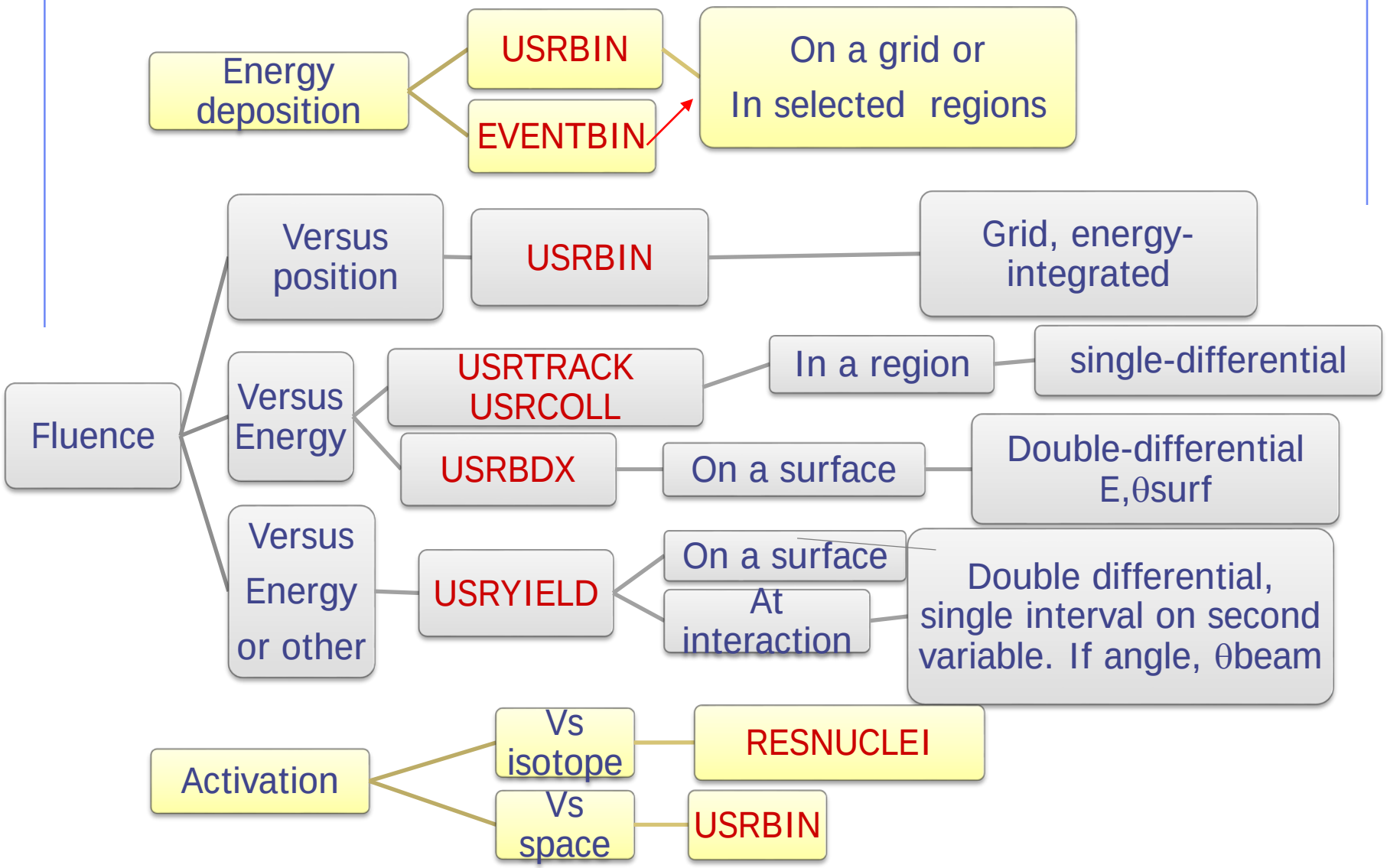
FLUKA Scoring

FLUKA Advanced Course

FLUKA Scoring & Results - Estimators

- It is often said that Monte Carlo (MC) is a "mathematical experiment" The MC equivalent of the result of a real experiment (*i.e.*, of a measurement) is called an estimator.
- Just as a real measurement, an estimator is obtained by sampling from a statistical distribution and has a statistical error (and in general also a systematic one).
- There are often several different techniques to measure the same physical quantity: in the same way the same quantity can be calculated using different kinds of estimators.
- FLUKA offers numerous different estimators, *i.e.*, directly from the input file the users can request scoring the respective quantities they are interested in.
- As the latter is implemented in a very complete way, users are strongly encouraged to preferably use the built-in estimators with respect to user-defined scoring
- For additional requirements FLUKA user routines are provided

Fluka estimators (see beginner course)



Reaction Rate and Cross Section [1/3]

- We call mean free path $\lambda[cm]$ the average distance travelled by a particle in a material before an interaction. Its inverse, $\Sigma\,[cm^{-1}]$ is the probability of interaction per unit distance, and is called macroscopic cross section. Both λ and Σ depend on the material and on the particle type and energy.

- For N identical particles, the number of reactions R occurring in a given time interval will be equal to the total distance travelled Nl times the probability per unit distance Σ: $R = Nl\Sigma$

- The reaction rate will be $\dot{R} = N\mathrm{d}l/\mathrm{d}t\ \Sigma = Nv\Sigma$, where v is the average particle velocity.

Reaction Rate and Cross Section [2/3]

- Assume now that $n(\boldsymbol{r},v)=dN/dV\ [cm^{-3}]$ be the density of particles with velocity $v=dl/dt\ [cm/s]$, at a spatial position $\boldsymbol{r}$. The reaction rate inside the volume element $\mathrm{d}V$ will be: $\mathrm{d}\dot{R}/\mathrm{d}V = n(\boldsymbol{r},v)v\Sigma$

- The quantity $\dot{\Phi}(\boldsymbol{r},v) = n(\boldsymbol{r},v)v$ is called fluence rate or flux density and has dimensions $[cm^{-3}\ cm\ s^{-1}]=[cm^{-2}\ s^{-1}]$.

- The time integral of the flux density $\Phi(\boldsymbol{r},v) = n(\boldsymbol{r},v)\mathrm{d}l$ is the fluence $[cm^{-2}]$

- Fluence is measured in particles per cm^2 but in reality it describes the **density of particle tracks**

- The number of reactions inside a volume V is given by the formula: $R = \Sigma\Phi V$ (where the product $\Sigma\Phi$ is integrated over energy or velocity)

Reaction Rate and Cross Section [3/3]

- Dividing the macroscopic cross section by N_0, the number of atoms per unit volume, one obtains the microscopic cross section $\sigma\,[barn=10^{-24}cm^2]$.

$$\frac{probability/cm}{atoms/cm^3} = \frac{probability\ x\ cm^2}{atoms} = atom\ effective\ area$$

i.e., the area of an atom weighted with the probability of interaction (hence the name "cross section").

- But it can also be understood as the probability of interaction per unit length, with the length measured in atoms/cm^2 (the number of atoms contained in a cylinder with a 1 cm^2 base).
- In this way, both microscopic and macroscopic cross section are shown to have a similar physical meaning of "probability of interaction per unit length", with length measured in different units. Thus, the number of interactions can be obtained from both, by multiplying them by the corresponding particle track-length.

Fluence estimation [1/2]

- Track length estimation:

USRTRACK

$$\dot{\Phi}(v)\,dt = n(v)\,v\,dt = \frac{dN(v)}{dV}\frac{dl(v)}{dt}dt = \lim_{\Delta V \to 0}\frac{\sum_i l_i(v)}{\Delta V}$$

- Collision density estimation (NOT IN VACUUM!):

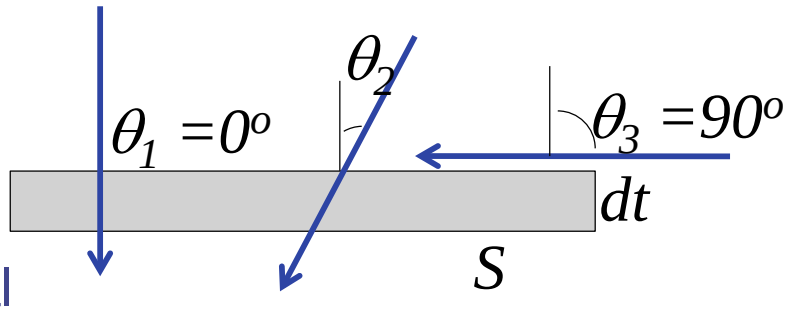
USRCOLL

$$\dot{\Phi}(v) = \frac{d\dot{R}(v)}{dV}\lambda(v)$$

$0 \times \infty$

Fluence estimation [2/2]

Surface crossing estimation



- Imagine a surface having an infinitesimal thickness dt A particle incident with an angle θ with respect to the normal of the surface S will travel a segment $dt/cos\theta$.
- Therefore, we can calculate an average surface fluence by adding $dt/cos\,\theta$ for each particle crossing the surface, and dividing by the volume $S\,dt$

$$\Phi = \lim_{dt \to 0} \frac{\sum_i \frac{dt}{\cos\theta_i}}{S\,dt}$$

- While the current J counts the number of particles crossing the surface divided by the surface:

$$J = dN/dS$$

The fluence is independent of the orientation of the surface S, while the current is NOT!

In an *isotropic field* it can be easily seen that for a flat surface $J = \Phi/2$

“Unusual” Scoring cards

- EVENTBIN is like USRBIN, but prints the binning output after each event instead of an average over histories
- ROTPRBIN sets the storage precision (single or double) and assigns rotations/translations for a given user-defined binning (USRBIN or EVENTBIN). Quite useful in case of LATTICE
- USERDUMP defines the events to be written onto a “collision tape” file Coupled to the **mgdraw** user routine
- AUXSCORE defines filters and conversion coefficients
- TCQUENCH sets scoring time cut-offs and/or Birks quenching parameters for binnings (USRBIN or EVENTBIN) indicated by the user
- DETECT scores energy deposition in coincidence or anti-coincidence with a trigger, separately for each “event" (primary history). Dedicated post-processing routine available.

Lattice Related Scorings

EVENTBIN or USRBIN with WHAT(1)=**8** :

Special user-defined 3D binning. Two variables are discontinuous (*e.g.,* region number), the third one is continuous, but not necessarily a space coordinate.

Variable	Type	Default	Override Routine
1st	integer	region number	MUSRBR
2nd	integer	lattice cell number	LUSRBL
3rd	float	Before used as η, now set to zero*	FUSRBV

* In the past it scored: $n=-ln(tan(0.5\ arctan(sqrt(x^2+y^2)/z))$

ROTPRBIN can assign rotations/translations (as defined by ROT-DEFI) for a given user-defined binning (USRBIN or EVENTBIN):

- this allows *e.g.*, defining a 'normal' scoring around a prototype and then 'replicating' the scoring to the respective lattices

Dose-Equivalent (not Dose)

For some quantities, there is the possibility to get built-in conversions, without the need for user routines, rather through dedicated generalized particles. The most commonly used is *dose equivalent* (ambient dose equivalent or effective dose):

DOSE-EQ	Dose Equivalent [pSv]
DOSEQLET	Dose Equivalent via Q(LET) – unrestricted LET in water – according to ICRP60 [GeV/g]

!!!! Different from :::

DOSE	total absorbed dose in GeV/g
DOSE-EM	as above but electromagnetic contribution only

DOSE-EQ is calculated by folding *particle fluences* with conversion coefficient sets, selected by the user among a list (see manual) through AUXSCORE. The default set (not requiring the AUXSCORE association) is "AMB74".

WARNING : in case of DOSE-EQ no coefficients available for heavy ions (ok for DOSEQLET) !!!

“FILTER” : AUXSCORE

There is the possibility to filter the estimators, restricting the scoring to a selected subset of particles.

Warning: filtering energy deposition (or similar) by particle type has small physical meaning, because the result depends on the delta ray threshold

New: possibility to filter activity with respect to parent isotope:

```
USRBIN         1.0  ACTIVITY   -87.       10.0                  1.0Co60
USRBIN         0.0       0.0     0.       10.0        0.0      10.0&
AUXSCORE     USRBIN -6002700.              Co60
```

The requested ion is coded in WHAT(2)= - (100*__Z__ + 100000*__A__ + __m__*100000000)
according to its **A**, **Z** and (optionally) isomeric state **m**
with 0==all, i.e. -2700 == all Cobalt isotopes
M=0➔ ground + isomeric states. M=9 ➔ only ground state

(the coding is the same as for filtering other estimators)

Reminder: Normalisation

- Beware! The MonteCarlo code does NOT know the intensity of your beam. It only knows how many events were run
- Normalization in all FLUKA intrinsic scoring is “per primary event”, or better, “per unit primary weight” if the source is biased
- The normalization to experimental conditions has to be done by the user
- Exception : Activation, just because the beam intensity is provided by the user in the IRRPROFI card.
- Exception, of course, event-by-event scoring, and DETECT
- Volume/area : Quantities are normalized per unit volume or per unit area **only for regular meshes** . For all others (i.e. USRBDX) area has to be provided by the user, or normalized offline

Statistical Errors [1]

- Can be calculated for single histories (not in FLUKA), or for batches of several histories
- Distribution of scoring contributions by single histories can be very asymmetric (many histories contribute little or zero)
- Scoring distribution from batches tends to Gaussian for $N \rightarrow \infty$, provided $\sigma^2 \neq \infty$ (thanks to Central Limit Theorem)
- The standard deviation of an estimator calculated from batches or from single histories is an estimate of the standard deviation of the actual distribution ("error of the mean")
- How good is such an estimate depends on the type of estimator and on the particular problem (but it converges to the true value for $N \rightarrow \infty$)

Statistical Errors [2]

- The variance of the mean of an estimated quantity x (e.g., fluence), calculated in N batches, is:

$$\sigma^2_{<x>} = \frac{1}{N-1}\left[\frac{\sum_1^N n_i x_i^2}{n} - \left(\frac{\sum_1^N n_i x_i}{n}\right)^2\right]$$

mean of squares – square of means

N – 1

where:

n_i = number of histories in the i^{th} batch

$n = \Sigma n_i$ = total number of histories in the N batches

x_i = average of x in the i^{th} batch: $x_i = \sum_{j=1}^{n_i} \frac{x_{ij}}{n_i}$

x_{ij} is the contribution to x of the j^{th} history in the i^{th} batch

In the limit $N = n$, $n_i = 1$, the formula applies to single history statistics

Statistical Errors [3]

Practical tips:

- Use always at least 5-10 batches of comparable size (it is not at all mandatory that they be of equal size)
- Never forget that the variance itself is a stochastic variable subject to fluctuations
- Be careful about the way convergence is achieved: often (particularly with biasing) apparent good statistics with few isolated spikes could point to a lack of sampling of the most relevant phase-space part
- Plot 2D and 3D distributions! Looking at them the eye is the best tool in judging the quality of the result

Statistical Errors [4]

from an old version of the MCNP Manual:

Relative error	Quality of Tally
50 to 100%	Garbage
20 to 50%	Factor of a few
10 to 20	Questionable
< 10%	Generally reliable

Note: max error is set to 99% in post-processing utilities

- Why does a 30% σ mean an uncertainty of a "factor of a few"?
  Because: σ is a stochastic variable
  Its evaluation is valid only in the Gaussian approximation
  If large ➔ probably not gaussian
- The MCNP guideline is empirically based on experience, not on a mathematical proof. But it has been generally confirmed as working also with other codes
- Small penetrations and cracks are very difficult to handle by MC, because the "detector" is too small and too few non-zero contributions can be sampled, even by biasing

Reminder: post-processing

- Post-processing utilities: are provided in the FLUKA distribution. They:
  - Sum the results from different cycles
  - Calculate statistical errors
  - Provide a summed file, that contains averages and statistics, and can be re-used to sum other cycles
  - Provide human-readable output and gnuplot-readable output
- These utilities are used by flair in the "Data process" tab
- Can be used directly from command line (indeed, they were born well before flair)
- All of them are in the $FLUPRO/flutil directory
- All of them are fortran codes
- Built automatically by the makefile

List of post-processing codes

- flutil/detsuw.f DETECT: sum and provide tab_lis
- flutil/gplevbin.f USRBIN: prepare 2D or 1D for plot
- flutil/usbrea.f USRBIN: convert to ascii
- flutil/usbsuw.f USRBIN: sum
- flutil/usrsuwev.f RESNUC: offline evolution
- flutil/usrsuw.f RESNUC: sum
- flutil/ustsuw.f USRTRACK: sum and provide tab_lis
- flutil/usxrea.f USRBDX: read summed file
- flutil/usxsuw.f USRBDX: sum and provide tab_lis
- flutil/usysuw.f USRYIELD: sum and tab_lis

Systematic Errors

- physics: codes are based on physics models, which cannot be perfect (especially in nuclear physics). Model quality is best shown by benchmarks at the microscopic level (e.g. thin targets)
- artifacts: due to imperfect algorithms, e.g., energy deposited in the middle of a step*, inaccurate path length correction for multiple scattering*, missing correction for cross section and *dE/dx* change over a step*, etc. Algorithm quality is best shown by benchmarks at the macroscopic level (thick targets, complex geometries)
- data uncertainty: results can never be better than available experimental data!

- material composition: not always well known. In particular *concrete/soil* composition (how much water content?). *Air* contains humidity and pollutants, has a density variable with pressure
- presence of additional material, not well defined (cables, supports...)
- geometries cannot be reproduced exactly (or would require too much effort)

Is it worth doing a very detailed simulation when some parameters are unknown or badly known?

Routines associated to FLUKA scoring

- comscw.f — weighting energy deposition and star production
- fluscw.f — weighting fluence, current and yield
- mgdraw.f — *general event interface*
- usrrnc.f — intercepting produced residual nuclei (at the end of their path)
- endscp.f — shifting energy deposition
- fldscp.f — shifting fluence
- musrbr.f — special USRBIN binning (lattice): returns region #
- lusrbl.f — special USRBIN binning (lattice): returns lattice #
- fusrbv.f — special USRBIN binning (lattice): returns zero

- mdstck.f
- stuprf.f
- stupre.f

} intercepting particle stack

comscw.f [1]

weighting energy deposition or star production

Argument list (all variables are input only)

IJ	:	particle type (1 = proton, 8 = neutron, etc.: see code in 5.1)
XA,YA,ZA	:	current particle position
MREG	:	current geometry region
RULL	:	amount to be deposited (unweighted)
LLO	:	particle generation
ICALL	:	internal code calling flag (not for general use)

Activated by option USERWEIG with WHAT(6) > 0.0.

Energy and stars obtained via SCORE, USRBIN & EVENTBIN and production of residual nuclei obtained via RESNUCLEi are **multiplied** by the value returned by this **function**.

If the logical flag LSCZER is set to .TRUE. , no amount will be scored (**filtering**)

comscw.f [2]

The user can implement any desired logic according to the argument list (*particle type, position, region, amount deposited, particle generation*) and information available in the included COMMONs.

COMMON SCOHLP provides the binning number JSCRNG [printed in the output file between the estimator type and the detector name] and the type ISCRNG of scored quantity:

ISCRNG = 1 → Energy density binning
ISCRNG = 2 → Star density binning
ISCRNG = 3 → Residual nuclei scoring
ISCRNG = 4 → Momentum transfer
ISCRNG = 5 → Activity density binning
ISCRNG = 6 → Net charge
ISCRNG = 7 → Residual nuclei density
ISCRNG = 8 → Pulse height detector (DETECT)
ISCRNG = 9 → Annihilation at rest

comscw.f [3]

Note that the same JSCRNG number can correspond to different detectors if of different type ISCRNG (use both to discriminate)

COMMON TRACKR gives current particle's properties and COMMON SOUEVT gives *current source particle's* ones

COMMON FLKMAT allows to access data concerning the current material, identified by the index
MEDFLK(MREG,IPRODC) where
IPRODC=1 for prompt and IPRODC=2 for radioactive decay particles)

fluscw.f [1]

weighting fluence, current and yield

Argument list (all variables are input only)

IJ	:	particle type
PLA	:	particle momentum (if > 0.0) or -PLA = kinetic energy (if < 0.0)
TXX, TYY, TZZ	:	particle current direction cosines
WEE	:	particle weight
XX, YY, ZZ	:	particle position
NREG	:	current region (after boundary crossing)
IOLREG	:	previous region (before boundary crossing). Useful only with boundary crossing estimators (for other estimators it has no meaning)
LLO	:	particle generation
NSURF	:	internal code calling flag (not for general use)

Activated by option USERWEIG with WHAT(3) > 0.0.

Yields obtained via USRYIELD, fluences calculated with USRBDX, USRTRACK, USRCOLL, USRBIN, and currents calculated with USRBDX are multiplied by the value returned by this **function**.

If the logical flag LSCZER is set to .TRUE. , no amount will be scored.

fluscw.f [2]

ISCRNG = 1 → Boundary crossing estimator
ISCRNG = 2 → Track-length binning
ISCRNG = 3 → Track-length estimator
ISCRNG = 4 → Collision density estimator
ISCRNG = 5 → Yield estimator

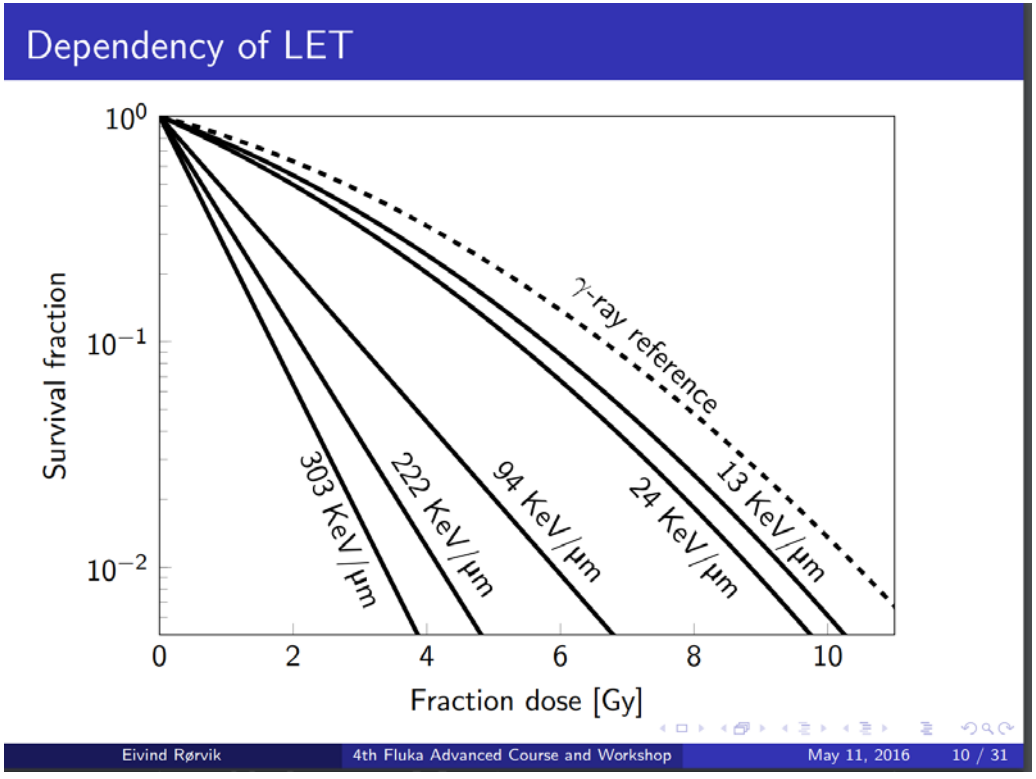
To act for a given region, it's useful to convert its name to the respective number as the routine is accessed the first time

CALL **GEON2R**('myregion', MYREG, IERR) Region Name to Region #

Warning: 'myregion' must be 8 characters. If shorter, pad with blanks:
'myr '

save it and compare it to NREG (MREG in case of comscw) runtime

Example of fluscw for particle therapy



Relative
Biological
Effectivness
Depends on
LET

Definition

Linear Energy Transfer is the energy locally imparted to the medium by a charged particle ($\mathrm{d}E$) traversing a distance $\mathrm{d}l$ in the medium.

$$\mathrm{LET} = \frac{\mathrm{d}E}{\mathrm{d}l}$$

Relevant “variations” of LET:

Averages of LET Spectrum

Fluence weighted LET (Track LET)

$$f(L) = \frac{\Phi(L)}{\int_0^\infty \Phi(L)\mathrm{d}L} \qquad \int_0^\infty f(L)\,\mathrm{d}L = 1$$

$$LET_f = \int_0^\infty L f(L)\,\mathrm{d}L$$

Dose weighted LET

$$d(L) = \frac{L}{\mathrm{LET_f}} f(L) \qquad \int_0^\infty d(L)\,\mathrm{d}L = 1$$

HOW? ? Multiply fluence by let in fluscw/comscw, using the getlet routine to get the LET

Getlet routine usage:

FLUKA Advanced Course

Getlet calling sequence:

```
DOUBLE PRECISION FUNCTION GETLET ( IJ, EKIN, PLA, TDELTA, MATLET )

* Input variables:
*
*     Ij = particle index (Paprop)
*   Ekin = particle kinetic energy (GeV)
*    Pla = particle momentum (GeV/c)
* Tdelta = maximum secondary electron energy (GeV)
*          (unrestricted if =< 0)
* Matlet = material index for which LET is requested
*
* Output variables:
*
* Getlet = (un)restricted LET (keV/(um g/cm3))
```

Ij must be a standard Fluka particle number, in particular "-2" or "<-6" for an ion heavier than α. In that case, the current particle must be indeed that ion, in order to have its properties properly set

…using variables from common TRACKR 1/2

Use the **FLUSCW** or **COMSCW** user routines for fluence and dose averaged LET "weighting" respectively. The following lines allow to obtain the LET of the current particle, if **200>JTRACK >-7**, **NTRACK=1**, where BEGLET and ENDLET are the LET's at the beginning and at the end of the current step respectively, **DTRACK(1)** is the energy deposited in the step

```
      INCLUDE '(TRACKR)'
      INCLUDE '(PAPROP)'
      INCLUDE '(FHEAVY)'
      IJ    = JTRACK
      EKIN  = ETRACK - AM (JTRACK)
      PLA   = PTRACK
      ENDLET= GETLET(IJ,EKIN,PLA,ZERZER,MATLET)
      EKIN  = EKIN + DTRACK(1)
      PLA   = SQRT(EKIN*(EKIN+TWOTWO*AM(JTRACK)))
      BEGLET= GETLET(IJ,EKIN,PLA,ZERZER,MATLET)
```

…using variables from common TRACKR 2/2

- If **JTRACK ≤-7, NTRACK=1**, substitute AM (JTRACK) with AMNHEA (JTRACK) and proceed as per the previous slide.
- If **JTRACK >200, NTRACK=1**, substitute JTRACK with JOTRCK and then proceed according to the previous instructions
- If **NTRACK =0**, it is a spot-like energy deposition, eg a particle below threshold (it should never occur inside FLUSCW)

mgdraw.f [1]

general event interface

Argument list (all variables are input only)

ICODE : FLUKA physical compartment originating the call
- = 1: call from subroutine KASKAD (hadrons and muons)
- = 2: call from subroutine EMFSCO (e^-, e^+ and photons)
- = 3: call from subroutine KASNEU (low-energy neutrons)
- = 4: call from subroutine KASHEA (heavy ions)
- = 5: call from subroutine KASOPH (optical photons)

MREG : current region

Subroutine mgdraw is activated by option USERDUMP with WHAT(1) ≥ 100.0,
usually writes a "collision tape", i.e., a file where all or selected transport events are recorded.
The default version (unmodified by the user) offers several possibilities, selected by WHAT(3)

mgdraw.f [2]

The different ENTRY points of mgdraw

MGDRAW called at each step, for trajectory drawing and recording dE/dx energy deposition
BXDRAW called at boundary crossings (no record)
EEDRAW called at event end (no record)
ENDRAW for recording point energy deposition events
SODRAW for recording source particles

One can remove their default writing and/or customize them.

Additional flexibility is offered by the user entry USDRAW, interfaced with the most important physical events happening during particle transport.

mgdraw.f [3]

All six entries can be activated at the same time by setting USERDUMP WHAT(3) = 0.0 and WHAT(4) ≥ 1.0.

They constitute a complete interface to the entire FLUKA transport. Therefore, mgdraw can be used not only to write a collision tape, but to do any kind of complex analysis (*e.g.*, event by event output as in HEP applications).

When mgdraw should be used with care

- When biasing is requested (non-analogue run). In this case, mgdraw should NOT be used for event-by-event scoring
- Whenever low-energy neutrons (E<20 MeV) are involved, unless one has a deep knowledge of the peculiarities of their transport and quantities (*i.e.*, kerma, etc)

mgdraw.f: the MGDRAW entry

MTRACK:	number of energy deposition events along the track
JTRACK:	type of particle
ETRACK:	total energy of the particle
WTRACK:	weight of the particle
NTRACK:	values of XTRACK, YTRACK, ZTRACK: end of each track segment
MTRACK:	values of DTRACK: energy deposited at each deposition event
CTRACK:	total length of the curved path

Other variables are available in TRACKR (but not written by MGDRAW unless the latter is modified by the user: particle momentum, direction cosines, cosines of the polarisation vector, age, generation, etc. see a full list in the comment in the INCLUDE file).

mgdraw.f: the BXDRAW entry

called at *boundary crossing*

Argument list (all variables are input only)

`ICODE` : physical compartment originating the call, as in the `MGDRAW` entry

`MREG` : region from which the particle is exiting

`NEWREG` : region the particle is entering

`XSCO, YSCO, ZSCO` : point where the boundary crossing occurs

mgdraw.f: the EEDRAW entry

called at the *event end*

Argument list (all variables are input only)		
`ICODE`	:	physical compartment originating the call, as in the `MGDRAW` entry

mgdraw.f: the ENDRAW entry

called at point-like energy deposition

(for example: stopping particles, photoelectric effect, ...)

Argument list (all variables are input only)

ICODE : type of event originating energy deposition
ICODE = 1x: call from subroutine KASKAD (hadrons and muons);
= 10: elastic interaction recoil
= 11: inelastic interaction recoil
= 12: stopping particle
= 14: particle escaping (energy deposited in blackhole)
ICODE = 2x: call from subroutine EMFSCO (electrons, positrons and photons)
= 20: local energy deposition (i.e. photoelectric)
= 21 or 22: particle below threshold
= 23: particle escaping (energy deposited in blackhole)
ICODE = 3x: call from subroutine KASNEU (low-energy neutrons)
= 30: target recoil
= 31: neutron below threshold
= 32: neutron escaping (energy deposited in blackhole)
ICODE = 4x: call from subroutine KASHEA (heavy ions)
= 40: ion escaping (energy deposited in blackhole)
ICODE = 5x: call from subroutine KASOPH (optical photons)
= 50: optical photon absorption
= 51: optical photon escaping (energy deposited in blackhole)
MREG : current region
RULL : energy amount deposited
XSCO, YSCO, ZSCO : point where energy is deposited

Mgdraw for energy deposition?

- Only if really needed. Warnings/tricks:
- Both the MGDRAW and ENDRAW entries must be used
- Particle WEIGHT must be taken into account if bias is used
- Particle steps might be longer than the accuracy you need (i.e. homogeneous regions that you wish to divide in cells)➔: do NOT deposit the energy in the middle of the step, rather DISTRIBUTE it along the step (already done for you in USRBIN..)
- Why is the step in /TRACKR/ subdivided in sub-steps? It happens for tracking in magnetic field, where the trajectory is approximated with arc segments (see lecture).
- Quenching of the signal (recombination in scintillators/ ionization) can be accounted for: activate with

  USERDUMP SDUM UDQUENCH

mgdraw.f: the SODRAW entry

Argument list
No arguments

It writes by default, for each source particle:

NCASE: number of primaries followed so far (with a minus sign to identify SODRAW output), from **COMMON CASLIM**

NPFLKA: stack pointer, in **COMMON FLKSTK**

NSTMAX: highest value of the stack pointer encountered so far, in **COMMON FLKSTK**

TKESUM: total kinetic energy of the primaries of a user written source, in **COMMON SOURCM**, if applicable. Otherwise = 0.0

WEIPRI: total weight of the primaries handled so far, in **COMMON SOURCM**

`NPFLKA` times: (all variables in `COMMON FLKSTK`)

`ILOFLK`:	type of source particle
`TKEFLK + AM`:	total particle energy (kinetic+mass)
`WTFLK`:	source particle weight
`XFLK, YFLK, ZFLK`:	source particle position
`TXFLK, TYFLK, TZFLK`:	source particle direction cosines

mgdraw.f: the USDRAW entry

called *after each particle interaction*
(requested by USERDUMP WHAT(4) ≥ 1.0)

Argument list (all variables are input only)

ICODE : type of event
ICODE = 10x: call from subroutine KASKAD (hadron and muon interactions);
= 100: elastic interaction secondaries
= 101: inelastic interaction secondaries
= 102: particle decay secondaries
= 103: delta ray generation secondaries
= 104: pair production secondaries
= 105: bremsstrahlung secondaries
ICODE = 20x: call from subroutine EMFSCO (electron, positron and photon interactions)
= 208: bremsstrahlung secondaries
= 210: Møller secondaries
= 212: Bhabha secondaries
= 214: in-flight annihilation secondaries
= 215: annihilation at rest secondaries
= 217: pair production secondaries
= 219: Compton scattering secondaries
= 221: photoelectric secondaries
= 225: Rayleigh scattering secondaries
ICODE = 30x: call from subroutine KASNEU (low-energy neutron interactions)
= 300: neutron interaction secondaries
ICODE = 40x: call from subroutine KASHEA (heavy ion interactions)
= 400: delta ray generation secondaries

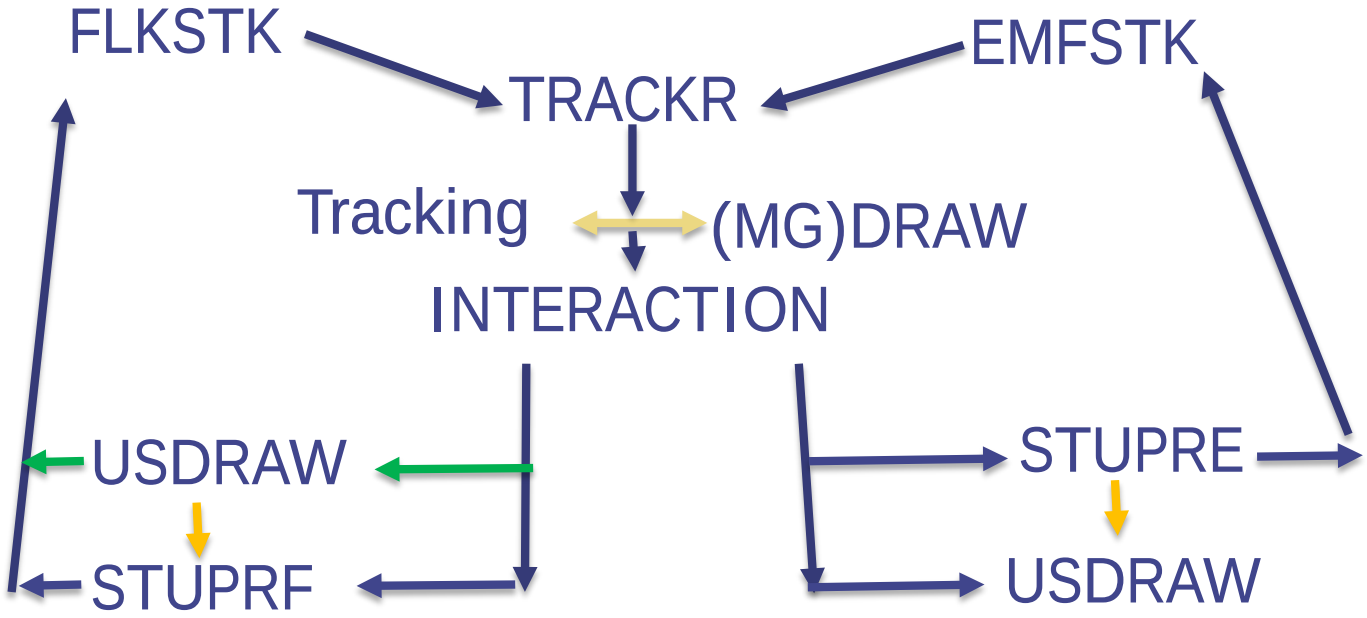
MREG : current region
XSCO, YSCO, ZSCO : interaction point

USDRAW: where to look?

- Secondaries are put on GENSTK common (kp=1,np)
- The surviving primary (if any) is also in GENSTK
- Exception: KASHEA delta ray generation where only the secondary electron is present and stacked on FLKSTK common for kp=npflka.
- Heavy fragments are put on FHEAVY common (kp=1, NPHEAV)
- But: heavy fragments from ion-ion interactions are in GENSTK (just to make things more difficult...)
- Residual nucleus (if any) (IBRES, ICRES etc)is in the RESNUC common (not included by default)
- Target nucleus (ICHTAR, IBTAR variables) is in RESNUC
- EMF particles: the code places them temporarily in GENSTK before calling USDRAW

Reminder: "tracking": variables

Correspondence	FLKSTK	EMFSTK		TRACKR
integer variable:	LOUSE	LOUEMF	$\longrightarrow$	LLOUSE
integer array:	ISPARK	IESPAK	$\longrightarrow$	ISPUSR
double precision array:	SPAREK	ESPARK	$\longrightarrow$	SPAUSR



Stuprf is called for each secondary

Stupre is called only once, loop is inside

Retrieve particle properties

- In Fluka, to each particle corresponds a numerical identifier, reported in the manual, coded in JTRACK, KPART(kp) , ILOFLK(NP)...
- With this, particle properties (mass, charge...) can be retrieved from common PAPROP.
- Warning: for recoils or kerma, JTRACK can contain a *generalized particle* code. In this case, the real particle code is in J0TRCK. For standard cases, J0TRCK = 0.
- This is not always true for ions/ fragments:
- Up to alphas: standard (although negative ID)

Retrieve particle properties :ions

- The currently TRANSPORTED ion has JTRACK=-2 or JTRACK <-6 (if appox ion transport) . Properties can be retrieved from
  - PAPROP with index –2 or
  - FHEAVY with index -JTRACK
- At the exit from interactions, ions can be in GENSTK, RESNUC, FHEAVY
- GENSTK: |KPART(KP)| > 10000 coded as

  KPART = -(1000000 * ism +100000 * Z +100 *A +IONID)

  ISM = isomeric state, presently always 0

  IONID = index in FHEAVY, if mass needed: AMNHEA(IONID).
- FHEAVY: Properties can be obtained from FHEAVY using ID = KHEAVY(KP)
- RESNUC: only zero or one, filled : IBRES>0

Retrieve particle properties :ions -II

- In the Fluka Stack (FLSTCK): ILOFLK (NP)

  ILOFLK = -(1000000 * ism +100000 * Z +100 *A +IONID)

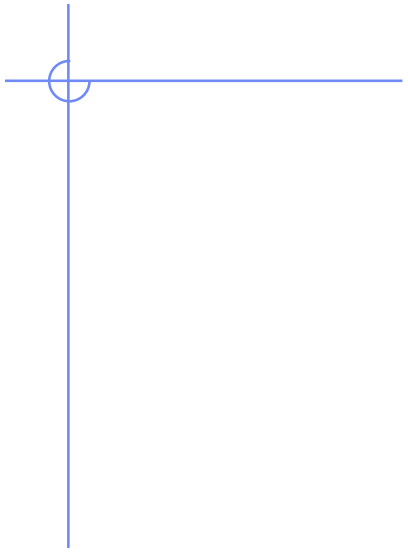
  ISM = isomeric state, presently always 0

  IONID = Ignore! Properties can be superseded at any moment

- Utility to "unpack" ILOFLKA and "large" KPART :

CALL USRDCI (ILOFLKA(NP), IONA , IONZ, IONM)

Gives back A, Z, isomeric state

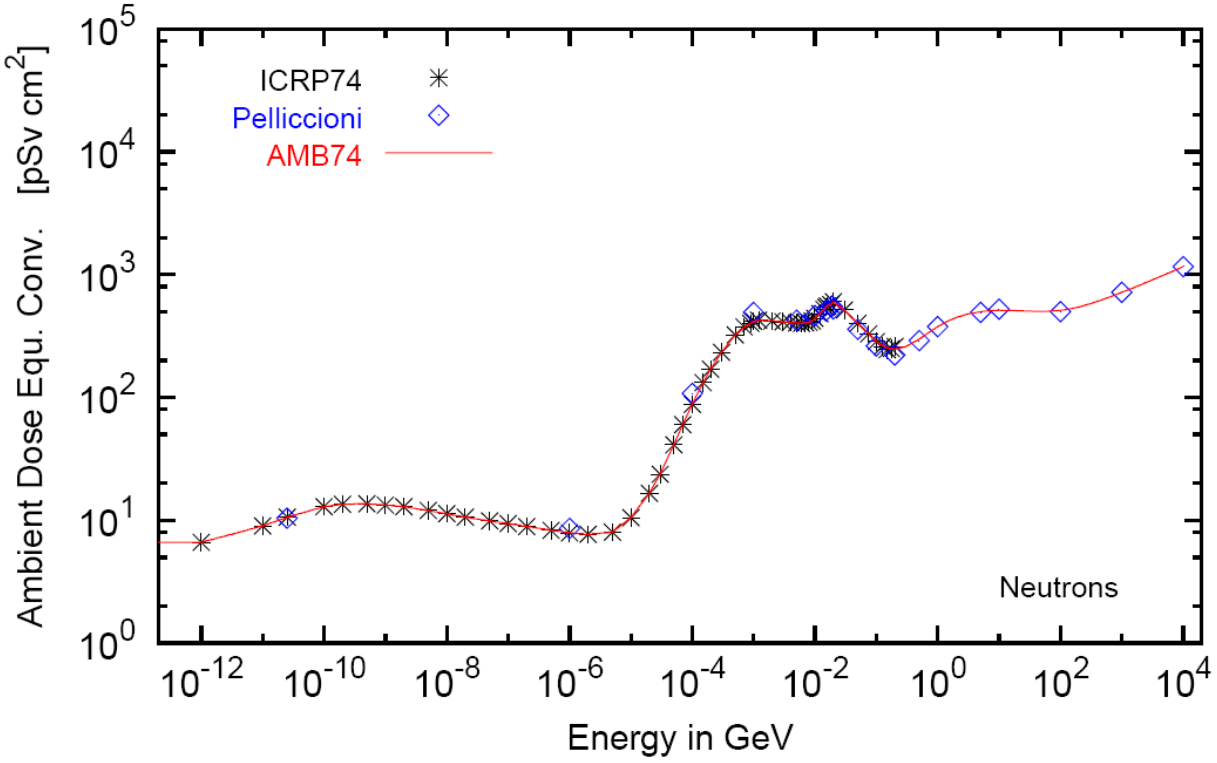


END

Backup

Conversion Coefficients

Conversion coefficients from fluence to ambient dose equivalent are based on ICRP74 values and values calculated by M.Pelliccioni. They are implemented for protons, neutrons, charged pions, muons, photons, electrons (conversion coefficients for other particles are approximated by these). AMB74 is the default choice for dose equivalent calculation.

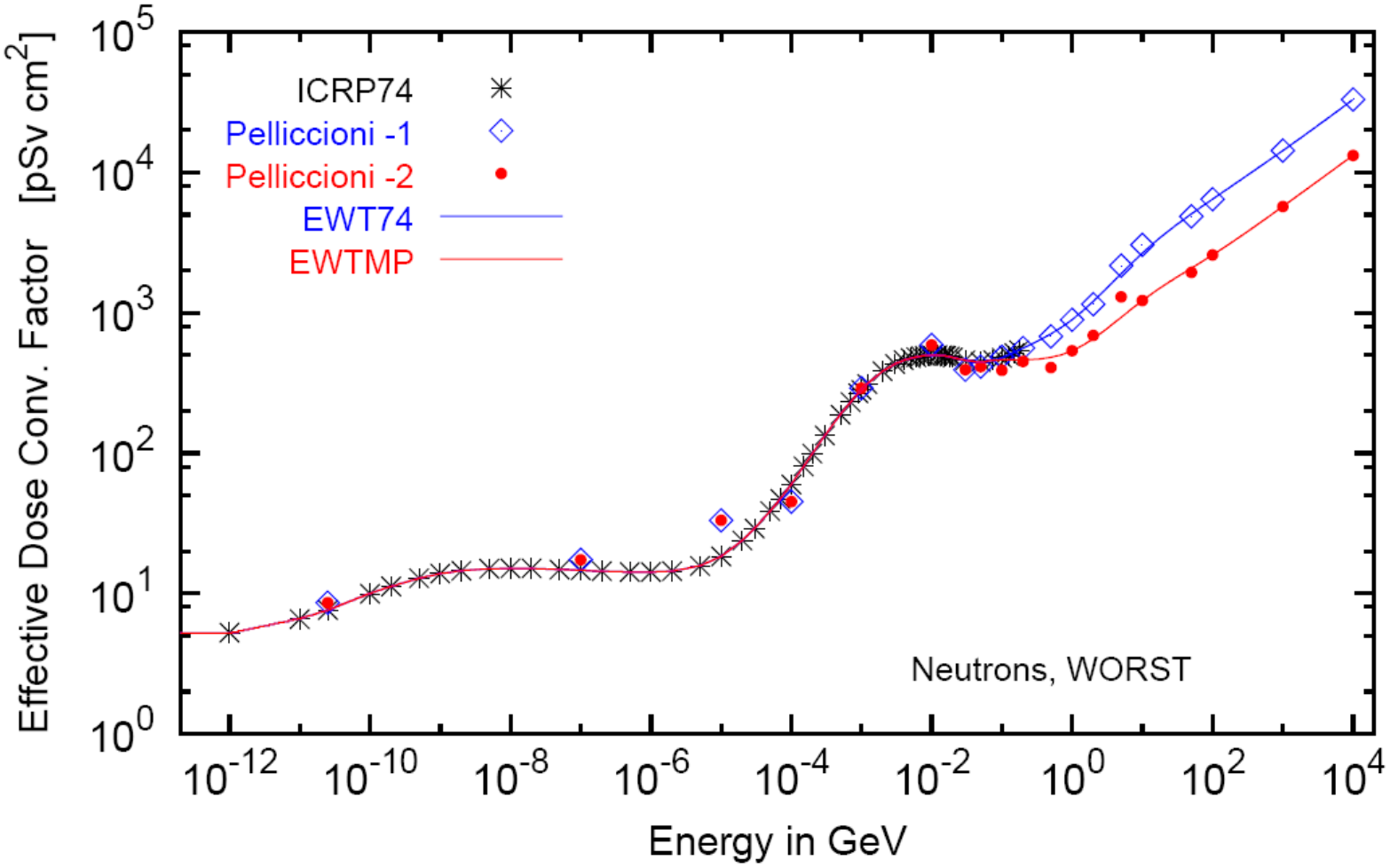


For more info: http://cern.ch/info-fluka-discussion/download/deq2.pdf

Fluence to effective dose coefficients

- Conversion coefficients from fluence to effective dose are implemented for three different irradiation geometries:
  - anterior-posterior
  - rotational
  - WORST ("Working Out Radiation Shielding Thicknesses") is the maximum coefficient of anterior-posterior, posterior-anterior, right-lateral and left-lateral geometries. It is recommended to be used for shielding design.
- Implemented for radiation weighting factors recommended by ICRP60 (e.g., SDUM=ETW74) and recommended by M.Pelliccioni (e.g., SDUM=EWTMP). The latter anticipate the 2007 recommendations of ICRP.
- Implemented for protons, neutrons, charged pions, muons, photons, electrons (conversion coefficients for other particles are approximated by these)
- Zero coefficient is applied to all heavy ions

Fluence to effective dose coefficients



USRYIELD

- Scores a double-differential particle yield around an extended or a point target.
- "Energy-like" quantities

Kinetic energy , total momentum , total energy , longitudinal momentum in the lab frame ,
longitudinal momentum in the c.m.s. frame , LET

- "Angle-like" quantities (in degrees or radians)

Rapidity in the lab frame , rapidity in the c.m.s. frame , pseudorapidity in the lab frame ,
pseudorapidity in the c.m.s. frame , Feynman-x in the lab frame ,
Feynman-x in the c.m.s. frame , transverse momentum , transverse mass ,
polar angle (*) in the lab frame , polar angle (*) in the c.m.s. frame ,
square transverse momentum , charge , weighted angle in the lab frame ,
weighted transverse momentum

Biasing Mean Free Paths

Multiplicity Tuning — BIASING

- Multiplicity tuning is meant to be to hadrons what LPB is for electrons and photons.
- A hadronic nuclear interaction at LHC energies can end in hundreds of secondaries. Except for the leading particle, many secondaries are of the same type and have similar energies and other characteristics
- The user can tune the average multiplicity in different regions

Interaction Length — LAM-BIAS

- Mean life / average decay length of unstable particles can be artificially shortened
- Can increase generation rate of decay products without discarding the parent
- For hadrons the mean free path for nuclear inelastic interactions can be artificially decreased. Useful for very thin targets, and also for photonuclear reactions where the cross section is relatively small

Warnings [I]

- USRBIN scoring algorithm:

  By selecting **WHAT(1)>=10**, *energy deposition, dose,* ... are distributed along the particle track (recommended!)

*** Activity/fission/neutron balance binnings cannot be track-length!!!

Point-wise quantities have to be scored at a point (select **WHAT(1)<10**)

- Badly defined USRBIN limits

******* Fluka stopped in Usrbin: "usr/eventbin" n. 1 *******
******* with zero width 0.000 for axis R ******

- Never use unit numbers smaller than 20 or higher than 99
  <20 reserved by FLUKA >99 FORTRAN limitation

- Never mix the output of different scoring cards in the same unit

- Verify that you didn't merge cycles referring to different input versions (change the name of the input file for every new problem!)

Example

- Thin window with low-E (5MeV) electron beam
- Energy deposition profile in the window
  (for radiation damage studies)
- Observation of ‘strange peaks’

- **Trying to understand:** lower e^--thresholds help
- **Real-Problem:** point-wise scoring requested