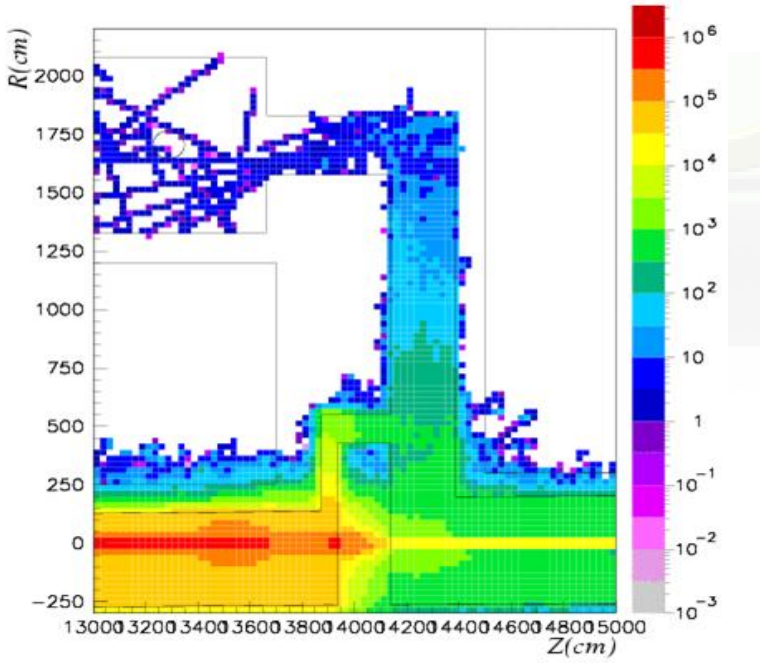
Simulation Optimization

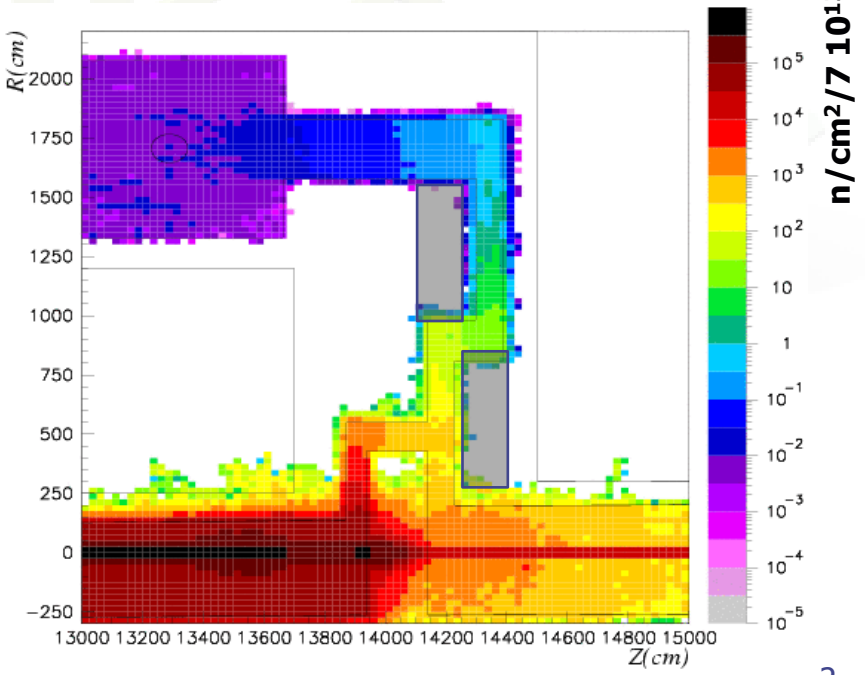
Advanced FLUKA Course

Concept

- Variance reduction techniques in Monte Carlo calculations reduce the computer time *or the opposite* to obtain results of sufficient precision in the phase-space region of interest.
- Remember: that precision is not the only requirement for a Good Monte Carlo calculation. Even a zero variance calculation cannot accurately predict natural behavior if other sources of error are not minimized.



No Bias and no maze



Region Biasing + maze

Monte Carlo Flavors

	Microscopic Analog	Microscopic Biased	Macroscopic Analog
Physics Models	Theoretical	Theoretical	Parameterizations
PDF sampling	Physical processes	Artificial distributions	Fits & Data
Predict Average	Yes	Yes	Yes
Predict Higher Moments	Yes	-	-
Preserves Correlations	Yes	-	-
Reproduces Fluctuations	Yes	-	-
Rare events	-	Yes	-
Predictability	Yes	Yes	-
Convergence	Slow	Fast privileged regions	Fast
Safe	Yes	Almost	-

Figure of Merit

- Computer cost of an estimator

$$FOM = \sigma^2 \times t$$

$$\sigma^2 = \text{Variance} \propto 1/N, \quad t = \text{CPU time} \propto N$$

- some biasing techniques are aiming at reducing the σ^2, others at reducing t
- often reducing σ^2 increase t, and vice versa
- therefore, minimizing $\sigma^2 \times t$ means to reduce σ at a faster rate than t increases, or vice versa
- $\Rightarrow$ the choice depends on the problem, and sometimes a combination of several techniques is most effective
- bad judgment, or excessive "forcing" on one of the two variables, can have catastrophic consequences on the other one, making computer cost "explode"

Biasing Techniques

FLUKA offers the following possibilities for biasing

- Importance Biasing (BIASING)
- Weight window (WW-FACTOr, WW-THRESh, WW-PROFIle)
- Leading Particle Biasing (EMF-BIAS)
- Multiplicity Tuning (BIASING)
- Biased down scattering for neutrons, *only for experts* (LOW-DOWN)
- Non analogue absorption (LOW-BIAS)
- Biasing Mean free paths (LAM-BIAS)
- User defined biasing (usbset.f, usimbs.f)

Other optimization checks

- CPU intensive consuming physics options and uses

Importance Biasing

- Importance biasing combines two techniques:
  - Surface splitting: Reduces σ but increases t
  - Russian roulette: which does the opposite
- It is the simplest, most safe and easiest to use of all biasing
- The user assigns a relative importance to each geometry region (the absolute value doesn't matter) based on:
  - expected fluence attenuation with respect to other regions
  - probability of contribution to score by particles entering the region
- Importance biasing is commonly used to maintain a constant particle population, compensating for attenuation due to absorption or distance.
- In FLUKA it can be tuned per type of particle

Importance Biasing

Surface Splitting

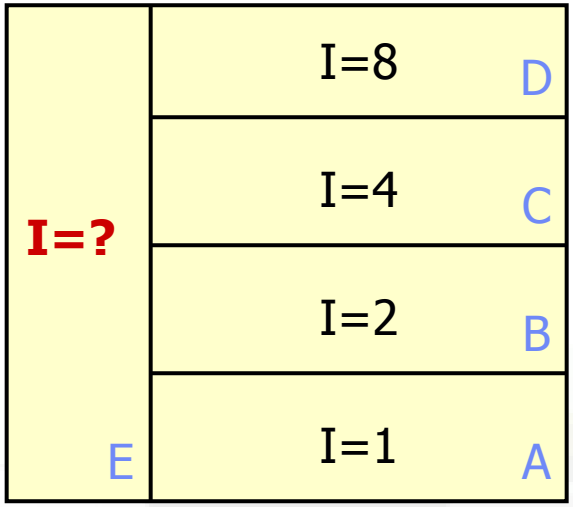
- If a particle crosses a region boundary coming from a region of important I_1 and enters a region of higher importance $I_2 > I_1$:
  - the particle is replaced on average by $n = I_2/I_1$ identical particles with the same characteristics
  - the weight of the daughter is multiplied by I_1/I_2
- If I_2/I_1 is too large, excessive splitting may occur, which will be prevented by FLUKA

Russian Roulette

- RR acts in the opposite direction: from higher importance I_1 to lower importance $I_2 < I_1$
  - The particle is submitted a random survival test with a chance of I_2/I_1 the particle survives with its weight increased by a factor I_1/I_2
  - with chance $1-I_2/I_1$ the particle is killed

Importance Biasing Problems

- Although important biasing is relatively easy and safe to use, there are a few cases where caution is recommended



Which importance shall we give to region E? Whatever value we choose we will get an inefficient splitting/RR at some of the boundaries.

- Another case is that of splitting in vacuum (or air). Splitting daughters are strongly correlated. It must be made sure that their further histories are differentiated enough to forget their correlation.
- The above applies in part also to muons (the differentiation provided by multiple scattering and by Landau *dE/dx* fluctuations is not always sufficient).

Leading Particle Biasing

- Leading Particle Biasing is available only for e^+, e^- and photons
- It is generally used to avoid the geometrical increase with energy of the number of particles in the electromagnetic shower
- It is a characteristic of EM interactions that 2 particles are present in the final state
- if LPB is activated only one is randomly retained and its weight is adjusted so as to conserve weight × probability
- The most energetic of the two particles is kept with higher probability
- LPB is reducing t, but increases σ by introducing large weight fluctuations. Therefore is should nearly always be backed up by WW
- Very useful for shielding calculations at both electron and proton accelerators

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

Weight Window

- The WW technique is a combination of splitting and RR, but it is based on the absolute value of the weight of each individual particle, rather than on relative region importance
- The user sets an upper and a lower weight limit, generally as a function of region, energy and particle
- Particles having a weight larger than the upper limit are split, those with weight smaller than the lower limit are submitted to RR $\Rightarrow$ killed or put back "inside the window"
- WW is a more powerful biasing tool than Importance Biasing, but it requires also more experience and patience to set it up correctly
  "*It is more an art than a science*" (From MCNP Manual)
- Use of the WW is essential whenever other biasing techniques generate large weight fluctuations in a given phase space region.
- For Low Energy neutrons the energy range should be defined through WW-PROFILe (instead of WW-THRESh)

Weight Windows - 2

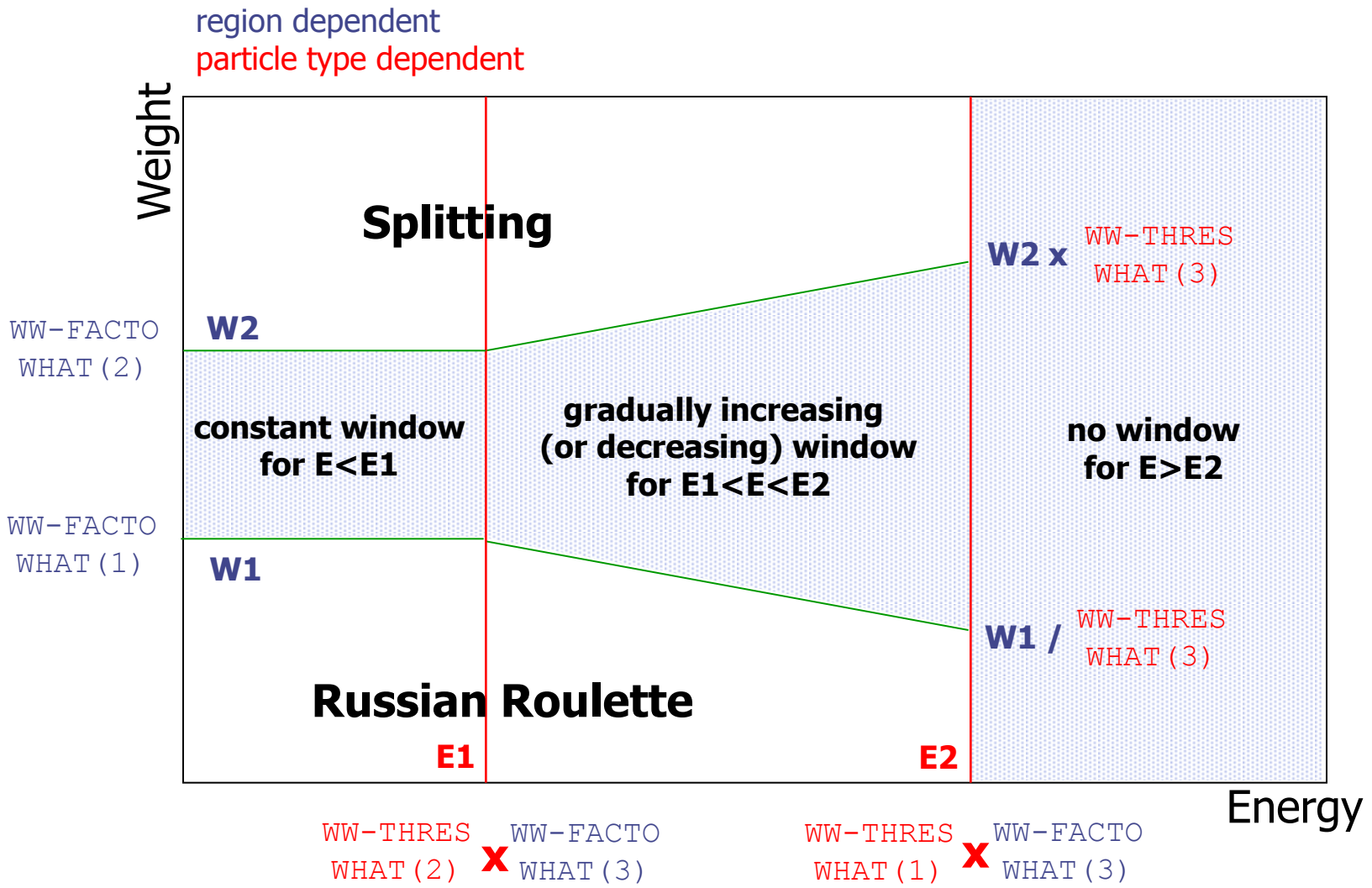
Killing a particle with a very low weight (with respect to the average for a given phase space region) decreases t but has very little effect on the score (and therefore on σ)

Splitting a particle with a large weight *increases* t (in proportion to the number of additional particles to be followed) but at the same time *reduces* σ by avoiding large fluctuations in the contributions to scoring.

The global effect is to reduce $\sigma^2 t$

A too wide window is of course ineffective, but also too narrow windows should be avoided. Otherwise, too much CPU time would be spent in repeated splitting / Russian Roulette. A typical ratio between the upper and the lower edge of the window is about 10. It is also possible to do Russian Roulette without splitting (setting the upper window edge to infinity) or splitting without Russian Roulette (setting the lower edge to zero)

Weight Windows - 3



Non analogue absorption

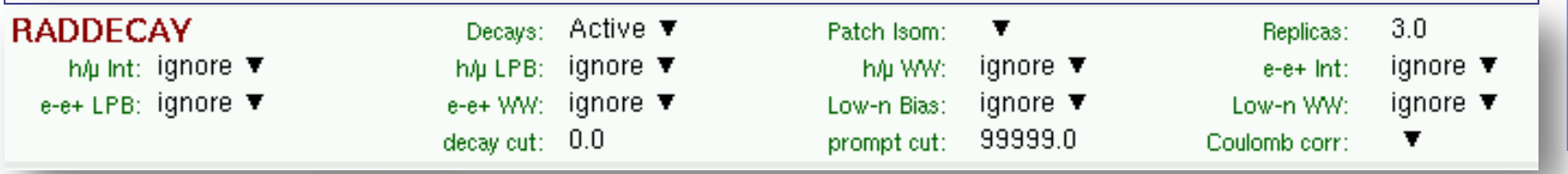
- Implemented in most low-energy neutron transport codes, where at each neutron collision the neutron always survives with its weight multiplied by the physical survival probability σ_s/σ_τ
- In FLUKA there is an additional choice: the user can force the neutron absorption probability to take an arbitrary value, pre-assigned on a region-by-region basis as a function of energy. The neutron weight is properly normalized by the code accordingly.

When and How

- A smaller survival probability is often assigned to thermal neutrons to limit the number of scatterings in non-absorbing media
- Also very useful in materials with unusual scattering properties (e.g. Iron)
- Survival probabilities too small with respect to the physical one σ_s/σ_τ may introduce large weight fluctuations due to the very different number of collisions suffered by individual neutrons. In these cases a WW should be applied.
- Card: LOW-BIAS *Also called survival biasing*

Card: RADDECAY [1/2]

```
* 1) request radioactive decays
RADDECAY          1.0          0          3.0          0000099999          0
```



WHAT(1) **= 1** **radioactive decays activated for requested cooling times**

Decays: Active "activation study case": time evolution calculated analytically for *fixed* (cooling) times. Daughter nuclei as well as associated radiation is considered at these (fixed) times

> 1 **radioactive decays activated in semi-analogue mode**

Semi-Analogue each radioactive nucleus is treated like all other unstable particles (random decay time, daughters and radiation), all secondary particles/nuclei carry time stamp ("age")

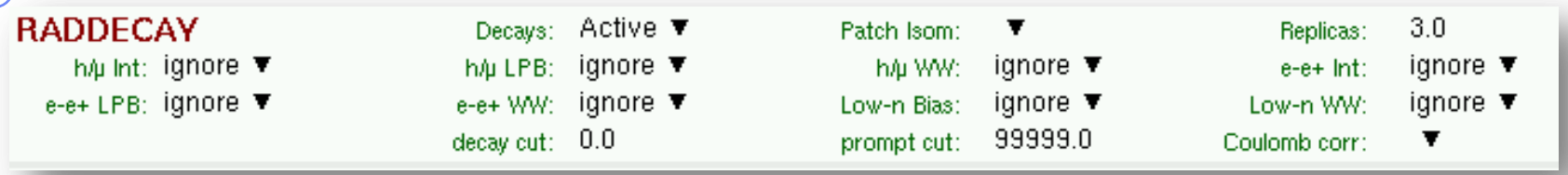
WHAT(2) **> 0** **isomer "production" activated**

Patch Isom: On

WHAT(3) **number of "replicas" of the decay of each individual nucleus**

Replicas: #

Card: RADDECAY [2/2]



WHAT(4) **switch for applying various biasing features only to prompt radiation or only to particles from radioactive decays**

h/m Int .. Low-n WW 9 digits, each responsible for a different biasing
Example:
5th digit, e+/e-/gamma leading particle biasing applied
000010000 to prompt radiation only
000020000 to decay radiation only
000030000 to both
Default: 111111111 (or blank as above)

WHAT(5) **multiplication factors to be applied to transport cutoffs**

decay cut: #
prompt cut: #

10 digits, first five for decay radiation, second five for prompt radiation (see manual)
Special cases:
0000099999 kill EM cascade for prompt radiation
9999900000 kill EM cascade for residual radiation

User written biasing

FLUKA offers the following routines for user-written biasing

- ubsset.f: User BiaSing SETting
  - called after reading in the input file and before the first event
  - allows to alter almost any biasing weight on a region-dependent basis
- usimbs.f: USer defined IMportance BiaSing
  - if activated, called at every particle step
  - allows to implement any importance biasing scheme based on region number and/or phase space coordinates
- udcdrl.f: User defined DeCay DiRection biasing and Lambda
  - only for neutrinos emitted in decays: bias the direction of emitted neutrino

Not biasing by itself, but it could be used for generating biased runs

- source.f: User written source
  - to sample primary particle properties from distribution in space, energy, time…

User Defined Importance Biasing

- Typical problem: Spend a lot of time to write the problem input, geometry and on the first runs you realize that statistics are not good
- First method (and safest) is to introduce region importance biasing. In FLUKA you can introduce it with two ways:
  - 1st Manually slice the geometry and increase the number of regions. Modifying an existing geometry to introduce biasing can be a very cumbersome process
  - 2nd Introduce the "importance biasing" information with a user fortran routine independent of the regions defined in the geometry
- Routine: usimbs.f

USer defined IMportance BiaSing

Allows to implement any importance biasing scheme based on region number and/or phase space coordinates

User Defined Importance Biasing

- Enable the call to USIMBS routine with the BIASING card:
  - WHAT(1) Particles to be biased
  - WHAT(2) and WHAT(3) ≠ 1.0 (Any value ≠ 1.0)
  - WHAT(4) Lower bound of region
  - WHAT(5) Upper bound of region
  - WHAT(6) Step
  - SDUM = USER

Remember:

- If WHAT(3)=1 for a region, the routine will not be called during tracking of particles inside that region
- Cannot have both normal importance BIASING with cards and the routine at the same time

usimbs.f – Routine

The routine is called on every particle step!

WARNING: can slow down the execution speed! Use with caution.

Input:

- Region information at the beginning and end of the step
- X,Y,Z coordinates through the TRACKR common.
  Beginning: X, Y, Ztrack(0)
  End: X, Y, Ztrack(Ntrack)
- Particle type and Energy could be used for even more advanced biasing schemes

Output:

The routine must return the importance ratio to the new position (end/beginning) in the variable FIMP < 10.0

Entry: USIMST

Split the particles step between interactions in half (or any other user defined value)

```
      SUBROUTINE USIMBS ( MREG, NEWREG, FIMP )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------
*     USer defined IMportance BiaSing:
*     Created on   02 july 2001    by    Alfredo Ferrari & Paola Sala
*                                                  Infn - Milan
*     Last change on 09-jul-01     by    Alfredo Ferrari
*     Input variables:
*                Mreg = region at the beginning of the step
*               Newreg = region at the end of the step
*     (thru common TRACKR):
*               Jtrack = particle id. (Paprop numbering)
*               Etrack = particle total energy (GeV)
*       X,Y,Ztrack(0) = position at the beginning of the step
*  X,Y,Ztrack(Ntrack) = position at the end of the step
*    Output variable:
*                Fimp = importance ratio (new position/original one)
*----------------------------------------------------------------------
      INCLUDE '(TRACKR)'
      FIMP   = ONEONE
      RETURN
      END
*======================================================================
*     Entry USIMST:
*     Input variables:
*                Mreg = region at the beginning of the step
*                Step = length of the particle next step
*    Output variable:
*                Step = possibly reduced step suggested by the user
*======================================================================
      ENTRY USIMST ( MREG, STEP )
      IF ( STEP .GT. ONEONE ) STEP = HLFHLF * STEP
      RETURN
*=== End of subroutine Usimbs =========================================
      END
```

usimbs.f – Important Notes

- The routine has only a relative effect on the weight of the particle
  ⇒ Beam particles can have any weight.
- Importance ratio will be limited by WW-THRESh card
- The Russian Roulette / Splitting will take place at the middle (defined by the ENTRY) of the step,
  and not on a fixed region boundary.
- The biasing position will be a little fuzzy depending on the particle step. This has a visible effect when it is applied to low density materials (i.e. air)
- Results will similar but not the same as with the manual region biasing.
- Is a great time saver for complex geometries, as well different biasing schemes
- Combined with the particle type and energy could become very powerful

usimbs.f: Function example

- Introduce an importance biasing assuming an exponential law of attenuation in the R direction $exp(-\lambda \times R)$, for R>1cm

```
RSTART = SQRT(XTRACK(0)**2 + YTRACK(0)**2 + ZTRACK(0)**2)
REND = SQRT(XTRACK(NTRACK)**2 + YTRACK(NTRACK)**2 +
       ZTRACK(NTRACK)**2)
IF (RSTART .LT. ONEONE) THEN
       FSTART = ONEONE
ELSE
       FSTART = EXP(-ALAMBDA * RSTART)
ENDIF
IF (REND .LT. ONEONE) THEN
       FEND = ONEONE
ELSE
       FEND = EXP(-ALAMBDA * REND)
ENDIF
FIMP = FSTART / FEND
```

usimbs.f: Sampling from array [1/4]

- Create a concentric cylindrical biasing with the weights sample from an array with various radii

- Define the variables:

```
      PARAMETER (NBIAS=5)
      PARAMETER (Xcenter=ZERZER)
      PARAMETER (Ycenter=ZERZER)
      DOUBLE PRECISION BIASR(NBIAS), BIASF(NBIAS)
* Radius
      DATA BIASR / 200.0, 250.0, 300.0, 400.0, 500.0 /
* Biasing factor
      DATA BIASF /  NBIAS * 2.0 /

      LOGICAL LFIRST
      DATA LFIRST / .TRUE. /
      SAVE LFIRST, BIASR, BIASF
```

usimbs.f: Sampling from array [2/4]

- Initialization – build the cumulative importance factor

```
      IF (LFIRST) THEN
            WRITE(LUNOUT,*) "*** User defined biasing ***"
            PREVBIAS = 1.0
            DO N=1,NBIAS
                  PREVBIAS = PREVBIAS * BIASF(N)
                  BIASF(N) = PREVBIAS
                  WRITE(LUNOUT,*) "Bias cylinder: ",N, BIASR(N), BIASF(N)
*                  convert the radius to square to avoid sqrt
                  BIASR(N) = BIASR(N)**2
            ENDDO
            LFIRST = .FALSE.
        ENDIF
```

usimbs.f: Sampling from array [3/4]

- Calculate the importance biasing

```
* Find square of radius for starting/ending position
      Rold = (Xtrack(0)-Xcenter)**2 + (Ytrack(0)-Ycenter)**2
      Rnew = (Xtrack(Ntrack)-Xcenter)**2 + (Ytrack(Ntrack)-Ycenter)**2
* Search index of the starting position
      Nold = NBinSearch(Rold, NBIAS, BIASR)
      IF (Nold.EQ.0) THEN
            BIASOLD = 1.0
      ELSE
            BIASOLD = BIASF(Nold)
      ENDIF
* Search index of the new position
      Nnew = NBinSearch(Rnew, NBIAS, BIASR)
      IF (Nnew.EQ.0) THEN
            BIASNEW = 1.0
      ELSE
            BIASNEW = BIASF(Nnew)
      ENDIF
      FIMP = BIASNEW / BIASOLD
```

usimbs.f: Sampling from array [4/4]

```
      INTEGER FUNCTION NBinSearch(x, N, VEC)
      INCLUDE '(DBLPRC)'
      DOUBLE PRECISION VEC(N)
      NLOW = 1
      NHIGH = N
      NBinSearch = 0
      IF (X.GE.VEC(NLOW) .AND. X.LE.VEC(NHIGH)) THEN
10          CONTINUE
                  MID = (NLOW+NHIGH)/2
                  IF (MID.EQ.NLOW) THEN
                        NBinSearch = MID
                        RETURN
                  ELSEIF (X .GT. VEC(MID)) THEN
                        NLOW = MID
                  ELSEIF (X .LT. VEC(MID)) THEN
                        HIGH = MID
                  ELSE
                        NBinSearch = MID
                        RETURN
                  ENDIF
            GOTO 10
      END IF
      END
```

Perform a binary search
Converge in $\log_2(N)$ steps

Neutrino Decay Biasing

- There is a special routine udcdrl.f where one can bias the direction of the emitted neutrino in decays.

DOUBLE PRECISION **FUNCTION** UDCDRL(IJ, KPB, NDCY, UDCDRB, VDCDRB, WDCDRB)

Input variables:

IJ	decaying particle
KPB	outgoing neutrino

Output variables:

U,V,W DCDRB	preferential outgoing direction for the neutrino
UDCDRL	Lambda for direction biasing (1-cos(θ))

The biasing expression is of the form:

$$e^{-(1-\cos\theta/\lambda)}$$

- Useful for neutrino applications like CNGS, Beta Beams…
- For a fixed direction the LAM-BIAS card with SDUM=DCY-DIRE could be used instead

Direction Biasing Example (n_TOF)

- Bias the direction of the neutrino in the pion decay so the daughter muon to be directed to the exp. area @(-120,0,18500). The direction is given in the lab-frame, and in this example the energies we are dealing are small, so safely we can assume that also in the lab-frame, the neutrino and muon go in opposite directions. The lambda is ¼ wide enough to cover the whole exp area.

```
      INCLUDE '(TRACKR)'
*
      UDCDRB = XTRACK (NTRACK) + 120.D+00
      VDCDRB = YTRACK (NTRACK) - ZERZER
      WDCDRB = ZTRACK (NTRACK) - 18500.D+00
      HLPHLP = SQRT ( UDCDRB**2 + VDCDRB**2 + WDCDRB**2 )
      UDCDRB = UDCDRB / HLPHLP
      VDCDRB = VDCDRB / HLPHLP
      WDCDRB = WDCDRB / HLPHLP
      UDCDRL = ONEFOU
*
```

Direction in the lab frame

Width [1-cos(θ)]

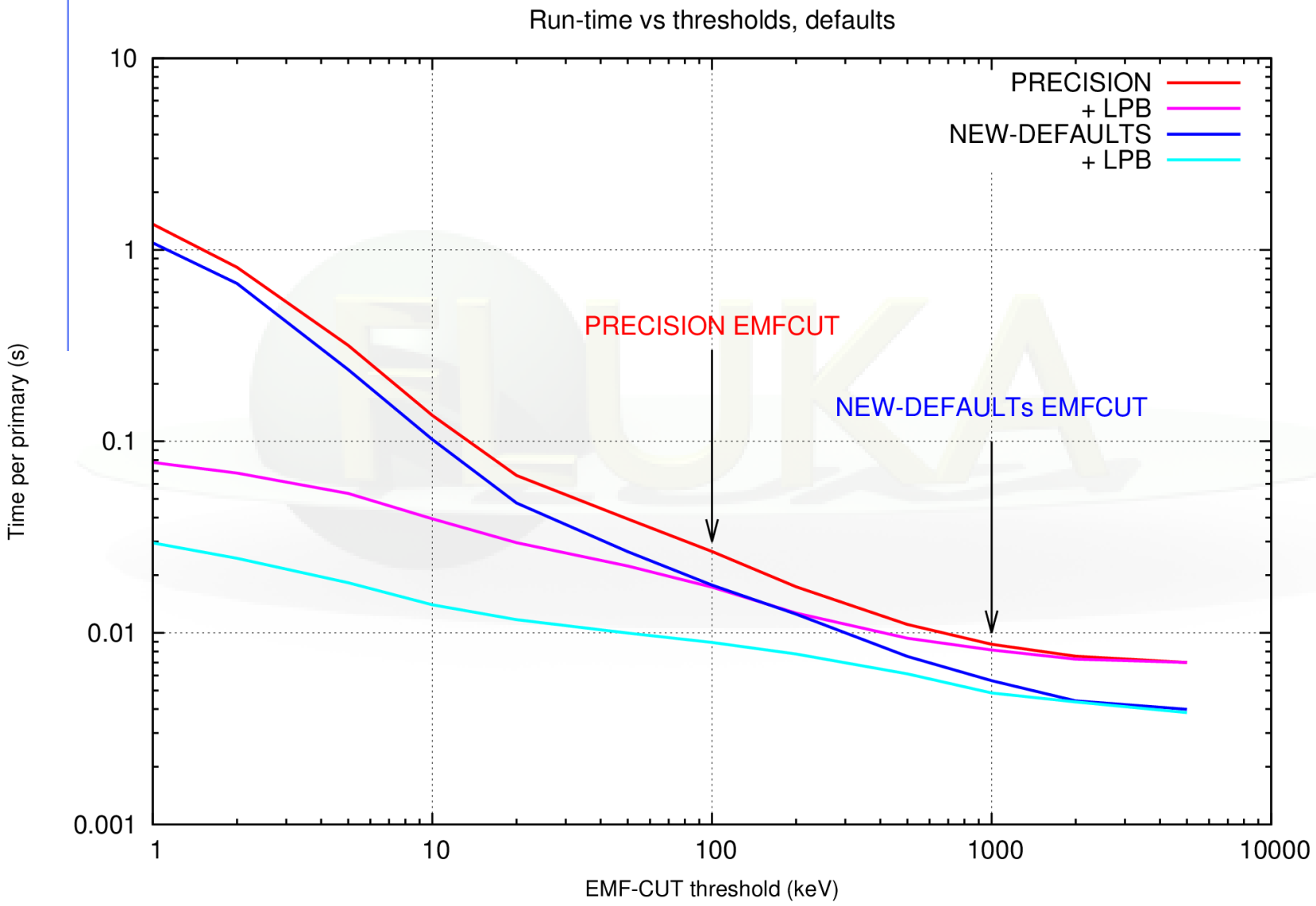
Other optimizations

- **Raising EMF thresholds (EMF-CUT)…**
  - in regions far from the scoring area or deep into the shielding
  - in heavy materials (at least 100-200 keV, unless in single scattering mode)
  - to GDR threshold (5-15 MeV) for neutron-dominated problems
  - to muon production threshold (>~100 MeV) for muon-dominated problems

  Notes:
  - These techniques will help get a faster estimate with relatively good accuracy
  - The sensitivity of results to EMF-CUT changes should be tested to assess biasing effect on results
  - In many cases further simulations with lower thresholds are not needed

  WARNING:
  - Rather than optimization, raising thresholds is OFTEN a user mistake as done irrespectively of the relevant physical processes

Example: CPU vs Thresholds and LPB



Other optimizations

- **Data initialization in user routines**: Avoid unnecessary repetition of global initialization steps at each call of a user routine
  - Use one-time check with logical variable, e.g.

```
IF (LFIRST) THEN
   ...
   LFIRST = .FALSE.
END IF
```

  - Example 1: number to name conversion: generate conversion table, save and use thereafter
  - Example 2: preliminary mathematical transformations before sampling in source routine

Other optimizations

- **Using symmetries for mesh scoring**, e.g. |x|, |y|, |z|, |r|, r-phi-z, r-phi-|z|, r-z, r-|z|

  Notes:

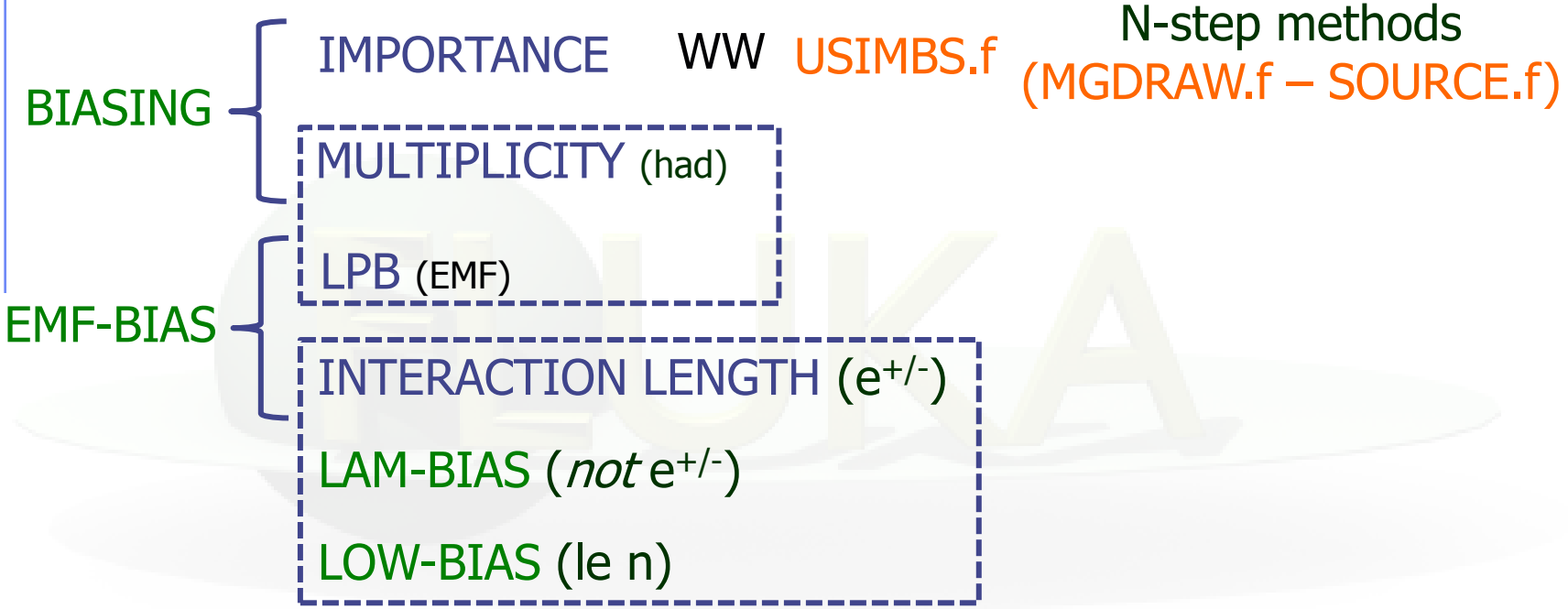
  - For example a symmetry over one plane saves 40% CPU time
  - Symmetry can also be forced through usrmed.f (activated by option MAT-PROP with SDUM USERDIRE).

- **Data writing**: dumping or writing large amount of data can considerably slow down calculations.
  - In large input files, and inherited setups, deactivate unnecessary legacy scoring, specially fine grid 3D histograms (e.g. USRBIN)
  - Avoid using 3D histograms (large size scoring), use instead 2D histograms (scoring in a plane) or 3D histograms with a coarse grid in one of the axis
  - Limit tape writing to necessary. Optimize the mathematical expressions in mgdraw.f
  - Tune-up scoring cards with short runs (good practice for several aspects) to optimize scoring scales

Other optimizations

- **Geometry optimization**
  - Adjust NAZ in region definition to make it equal or slightly larger than the number of neighboring regions. Check for for "GEOMETRY SEARCH ARRAY" warnings in output
  - If a region has too many neighboring cells, subdivide it in zones or regions to lower number of neighboring cells per zone
  - If there are components (with sub-structure) that are repeated many times in the geometry use LATTICE capabilities to reduce number of regions
- **More:** Biasing impurities in compounds, number of replica in activation/decay…
- **Other checks:** these are *not* ON by default, so check only if you are recycling an input file or you are using it for a different application *and* CPU is an issue:
  - Make sure single EMF scattering (MULSOPT) is not ON unless needed (low energy, thin geometry)
  - If not looking at activation, heavy ion evaporation does *not* need to be switched ON (*WHAT(2)* of PHYSICS card with SDUM EVAPORAT)

Summary Biasing Techniques



Reduce memory / check requirements

Use *physics process* and *thresholds* relevant to the problem

Warnings…

- **BIASING focuses the CPU onto a phase-space area of the problem, sacrificing other areas**
- **Reducing simulation time does not necessarily mean the simulation has been optimized**
- **Similarly, reducing variance does not necessarily mean the simulation has been optimized**
- **Small statistical error does not mean the error is actually low. This is specially true if splitting / two step methods, etc. are used. Correlations may be hidden.**
- **Biasing may reduce the error in the estimation of the *average* of a variable, but the higher momenta of that variable will not be obtained (e.g. the variance of the PDF distribution for that variable cannot be estimated).**
- **Simulation optimization may safe CPU time, but it can cost considerable manpower**

FLUKA

Weight Windows - 4

WW-FACTO	13.0	120.0	1.5	27.0	31.0	2.0

Defines Weight Windows in selected regions

WHAT(1) >= 0.0 : Window "bottom" weight

< 0.0 : resets to -1.0 (no Russian Roulette, Default)

Weight below which *Russian Roulette* is played at the lower energy threshold (set by WW-THRES).

WHAT(2) > 1.7 * WHAT(1) : Window "top" weight

= 0.0 : ignored

=< 1.7 * WHAT(1) : resets to infinity

(no Splitting, Default)

Weight above which *Splitting* is applied at the lower energy threshold (set by WW-THRES).

WHAT(3) > 0.0 : Multiplicative factor (Default: 1.0)

= 0.0 : ignored

< 0.0 : resets to 1.0

Factor to be applied to the two energy thresholds for Russian Roulette / Splitting (set by WW-THRES)

Weight Windows - 5

WW-FACTO	13.0	120.0	1.5	27.0	31.0	2.0

WHAT(4) = lower bound of the region indices (Default = 2.0)

WHAT(5) = upper bound of the region indices (Default = WHAT(4))

WHAT(6) = step length in assigning indices (Default = 1.0)

SDUM : a number from 1.0 to 5.0 in any position, indicating the **low-energy neutron weight-window profile** to be applied in the regions selected (see WW-PROFI). (Default = 1.0)
= blank, zero or non numerical: ignored
< 0.0 : resets to 1.0

Attention: Option WW-FACTO alone is not sufficient to define a weight window. One or more WW-THRES cards are also necessary in order to activate the window.

Weight Windows - 6

WW-THRES	2.0	0.05	2.4	3.0	7.0	0.0

Defines the energy limits and particle-dependent modification factors

WHAT(1) > 0.0: upper kinetic energy threshold (GeV)
Low-energy neutrons: lower group number (included)
= 0.0: ignored
< 0.0: any previously selected threshold is cancelled

WHAT(2) >= 0.0 and < WHAT(1): lower kinetic energy threshold (GeV)
Low-energy neutrons: upper group number (included)
< 0.0 or > WHAT(1): WHAT(2) is set = WHAT(1)

WHAT(3) > 0.0: amplification factor to define the weight window width at the higher energy threshold represented by WHAT(1).

The weight window at the higher energy threshold is obtained by multiplying by WHAT(3) the upper weight limit and by dividing by the same factor the lower weight limit. (Default = 10.0)

< 0.0: |WHAT(3)| multiplication factor for the lower and upper weight limits for the particles selected by WHAT(4-6) (Default = 1.0)

Weight Windows - 7

WW-THRES	2.0	0.05	2.4	3.0	7.0	0.0

WHAT(4) = lower bound of the particle indices (Default = 1.0)

Note that particle index 40 indicates low-energy neutrons (for this purpose only!). Particle index 8 indicates neutrons with energy > 19.6 MeV.

WHAT(5) = upper bound of the particle indices
(Default = WHAT(4) if WHAT(4) > 0, all particles otherwise)

WHAT(6) = step length in assigning indices (Default = 1.0)

SDUM = PRIMARY: the weight window applies also to primary particles (default)
= NOPRIMARy: the weight window doesn't apply to primaries

Selecting Weight Windows - 1

```
BIASING          0.0        0.0        4.64         8.         18.         2.PRINT
```

FLUKA output file:

```
Hadron importance RR/Splitting counters


Reg. #  N. of RR  <Wt>  in  <Wt> kil  Reg. #  N. of RR  <Wt>  in  <Wt> kil  Reg. #  N. of RR  <Wt>  in  <Wt> kil
     1  0.00E+00  0.00E+00  0.00E+00        2  0.00E+00  0.00E+00  0.00E+00        3  1.15E+05  9.31E-01  4.70E-02

Reg. #  N. of Sp  <Wt>  in  <Wt> out  Reg. #  N. of Sp  <Wt>  in  <Wt> out  Reg. #  N. of Sp  <Wt>  in  <Wt> out
     1  0.00E+00  0.00E+00  0.00E+00        2  0.00E+00  0.00E+00  0.00E+00        3  0.00E+00  0.00E+00  0.00E+00

Reg. #  N. of RR  <Wt>  in  <Wt> kil  Reg. #  N. of RR  <Wt>  in  <Wt> kil  Reg. #  N. of RR  <Wt>  in  <Wt> kil
     4  1.36E+04  4.66E-01  1.47E-01        5  8.97E+03  3.22E-01  1.06E-01        6  6.03E+03  2.16E-01  7.10E-02

Reg. #  N. of Sp  <Wt>  in  <Wt> out  Reg. #  N. of Sp  <Wt>  in  <Wt> out  Reg. #  N. of Sp  <Wt>  in  <Wt> out
     4  1.01E+05  9.99E-01  7.64E-01        5  9.25E+04  6.80E-01  5.23E-01        6  8.24E+04  4.65E-01  3.55E-01
…
```

"N. of RR" --> Number of FLUKA particles entering a region and which are not split (i.e., particles undergoing Russian Roulette as well as neither Russian Roulette nor splitting)

"<Wt> in" --> Average weight of these particles

"<Wt> kil" --> Average weight of particles killed after being submitted to Russian Roulette

"N. of Sp" --> Number of FLUKA particles entering the region and which are split

"<Wt> in" --> Average weight of these particles

"<Wt> out" --> Average weight of particles after being submitted to splitting

Selecting Weight Windows - 2

where

A = "N. of RR" + "N. of Sp"
= *total number of particles entering the region*

B = ("<Wt> in"_RR * "N. of RR") + ("<Wt> in"_Sp * "N. of Sp")
= *total weight of the particles entering the region*

B/A = *average weight of the particles entering the region*

Note -1: RR and splitting arising from Weight-Window biasing (options WW-FACTOR, WW-THRESh, WW-PROFI) or from multiplicity biasing (WHAT(2) in option BIASING) are not accounted for in the counters.

Note – 2: Separate counters are printed for hadrons/muons, electrons/photons and low-energy neutrons (referring to importance biasing requested by BIASING, respectively, with WHAT(1) = 1.0, 2.0 and 3.0, or = 0.0 for all).

Selecting Weight Windows - 3

Strategy:

1. run without any biasing and print counter, *e.g.,*

```
BIASING       0.0       1.0       1.0       1.        9.          PRINT
```

2. analyze counter and adjust region importance biasing, *e.g.,* according to the inverse of the attenuation in shielding, add other biasing, *e.g.,* leading particle biasing run and print counter again

```
BIASING       0.0       1.0       1.0       1.        9.          PRINT
BIASING       0.0       1.0      1.47       4.
BIASING       0.0       1.0      2.15       5.
BIASING       0.0       1.0      3.16       6.
BIASING       0.0       1.0      4.64       7.
BIASING       0.0       1.0      4.64       8.
```

3. analyze counter, select Weight Windows (WW-THRES, WW-FACTO) around average weights and perform final (high-statistics) run

Selecting Weight Windows - 4

Alternative Strategy:

1. Include USERDUMP card in input file to activate call to **mgdraw.f**

```
USERDUMP     100.0       25.0        4.0         1.
```

2. Edit mgdraw.f (bxdraw) to print energy and weight of particles (filter by particle if wanted) entering specific areas of the geometry, e.g.

```
IF ( .NOT. LFCOPE) THEN
   LFCOPE = .TRUE.
   OPEN (UNIT = 86, FILE = "tuneWW.dat", STATUS = "UNKNOWN")
END IF
. . .
IF (MRGNAM.eq."OLDCELL".and.NRGNAM.eq."NEWCELL".and.jtrack.eq.3) THEN
   write(86,*)wtrack, etrack
END IF
. . .
```

3. Produce scatter plot weight(E) by loading file "tuneWW.dat" in **gnuplot**

```
gnuplot> plot 'tuneWW.dat' using 2:1
```

3. Tune WW accordingly

usimbs.f: Simple Example

- Biasing a factor of 2 every 50 cm on the Z direction from 100cm to 500cm

```
ZSTART = ZTRACK(0)
IF (ZSTART .LT. 100.0D0) THEN
        FSTART = ONEONE
ELSE IF (ZSTART .GT. 500.0D0) THEN
        FSTART = TWOTWO ** NINT((500.0D0-100.0D0)/50.0D0)
ELSE
        FSTART = TWOTWO ** NINT((ZSTART-100.0D0)/50.0D0)
ENDIF
ZEND = ZTRACK(NTRACK)
* Similarly calculate the FEND from ZEND
...
FIMP = FEND / FSTART
```

Initial position

Final position

Importance Ratio