Biasing techniques

21th FLUKA Beginners' Course
ALBA Synchrotron (Spain)
April 8-12, 2019

Overview

General concepts:

Analog vs. biased Monte Carlo calculation
Statistical weight

Biasing options
(only the most important / common options available in FLUKA)

Importance biasing
Leading particle biasing
Multiplicity tuning
Biasing mean-free paths - decay lengths biasing
                                 - hadronic inelastic interaction lengths

Additional information:
 User-written biasing
  Weight Windows

Preliminaries

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
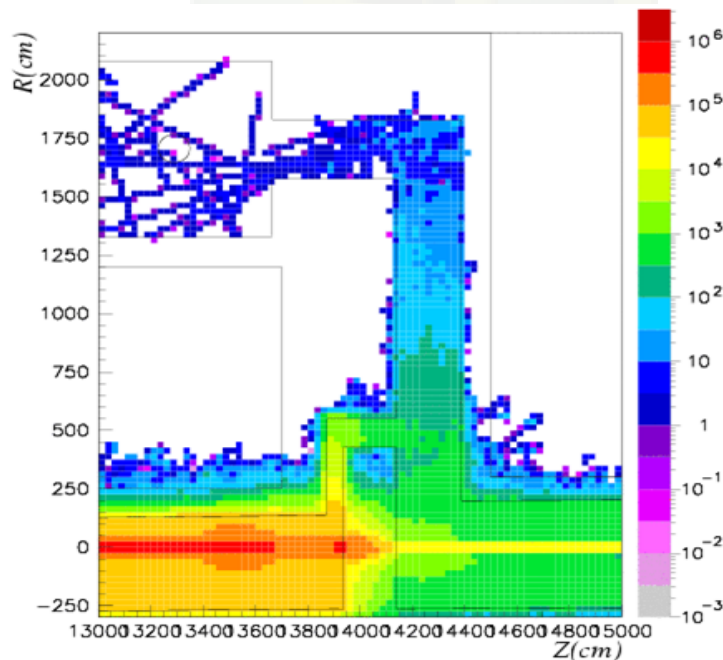
No Bias and no maze
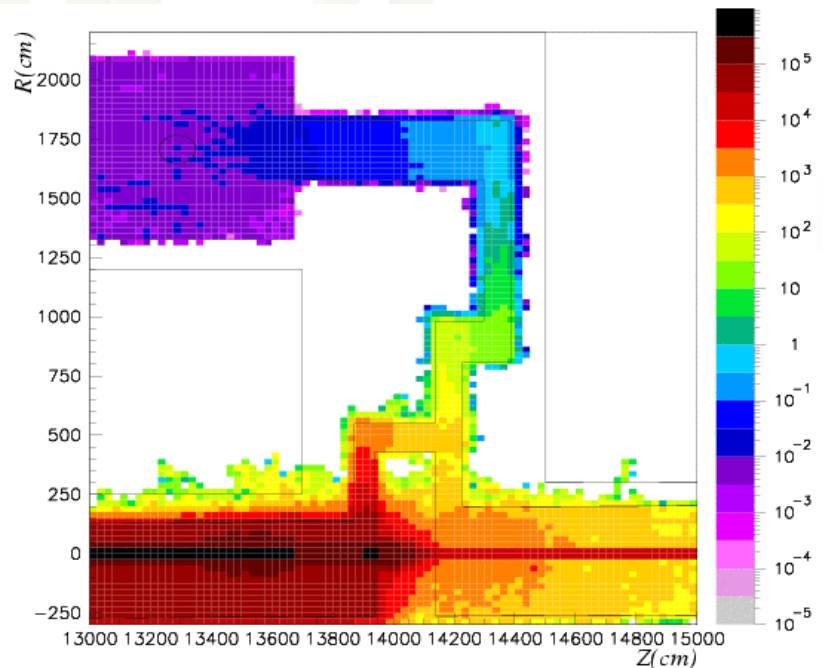
Region Biasing + maze

Is optimization necessary?

- Visual inspection of histograms (especially 2D maps) is a powerful way to determine if simulations have **NOT** converged in the area of interest. Big fluctuations between neighboring bins is a warning of uncertainties and potential error

- However, even in smooth 2D histograms lack of convergence may be present, e.g.:
  - If the scoring scale spans over many orders of magnitude significant differences may are hard to spot
  - 2-step simulations may give a false sense of convergence if first step has not converged but 2nd step has. Plot also maps for the 1st step!
  - Relatively smooth maps may have some 'hard'-tracks overlaid typically indicating strong energy dependent weight dispersion

Analogue vs Biased – 1

Analog Monte Carlo

- samples from actual phase space distributions
- predicts average quantities and all statistical moments
    of any order
- preserves correlations and reproduces fluctuations
    (provided the physics is correct…)
- is *(almost)* safe and can *(sometimes)* be used as "black box"

BUT

- is inefficient and converges very slowly
- fails to predict important contributions due to rare events

Analogue vs Biased – 2

Biased Monte Carlo

- samples from artificial distributions and applies a weight to the particles to correct for the bias
- predicts average quantities, but not the higher moments
  (on the contrary, its goal is to minimize the second moment)
- same mean with smaller variance, i.e., faster convergence

BUT

- cannot reproduce correlations and fluctuations
- requires physical judgment, experience and a good understanding of the problem (it is not a "black box"!)
- in general, a user does not get the definitive result after the first run, but needs to do a series of test runs in order to optimize the biasing parameters

→ **balance between user's time and CPU time**

Figure of Merit

- Computer cost of an estimator

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

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

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

Other optimization checks
- CPU intensive consuming physics options and uses

Importance biasing

Helpful in:

- Large geometries
- Problems with strong attenuation

Importance biasing – 1

- it is the simplest, "safest" and easiest to use of all biasing techniques

- importance biasing combines *two techniques*:

  Surface Splitting   (reduces σ but increases t)
  Russian Roulette   (does the opposite)

- the user assigns a relative importance to each geometry region (the actual absolute value doesn't matter), based on

  1. expected fluence attenuation with respect to other regions
  2. probability of contribution to score by particles entering the region

- Importance biasing is commonly used to maintain a uniform particle population, compensating for attenuation due to absorption or distance
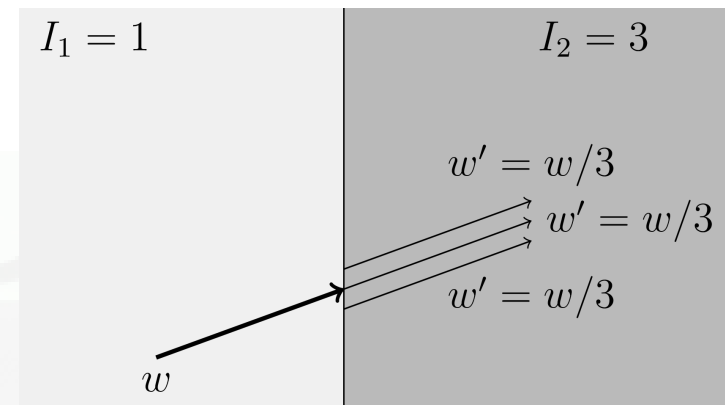
- In FLUKA it can be tuned per particle type

Importance biasing - 2

Surface Splitting

A particle crosses a region boundary, coming from a region of importance I_1 and entering a region of **higher** importance $I_2 > I_1$:

- the particle is split in
  $n = I_2/I_1$ identical particles
  with the same characteristics

- the weight of each "daughter"
  is multiplied by I_1/I_2

$I_1 = 1$ $I_2 = 3$

$w' = w/3$

$w' = w/3$

$w' = w/3$

w

If I_2/I_1 is too large, excessive splitting may occur with codes which do not provide an appropriate protection .

An internal limit in FLUKA prevents excessive splitting if I_2/I_1 is too large (> 5), a problem found in many biased codes.
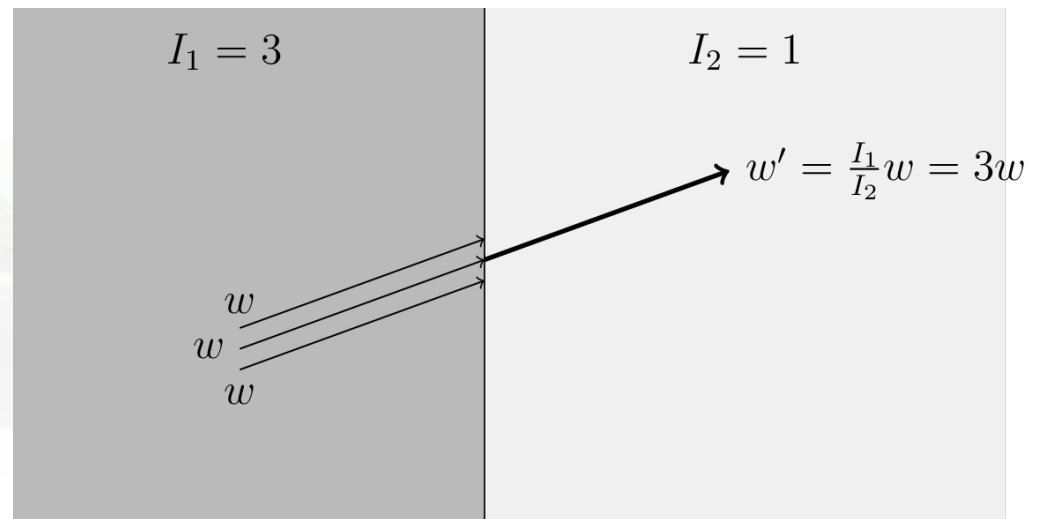
Importance biasing - 3

Russian Roulette

A particle crosses a region boundary, coming from a region of importance I_1 and entering a region of *lower* importance $I_2 < I_1$:

- the particle survives with probability I_2/I_1 and weight increased by I_1/I_2

- the particle is killed with probability $(1 - I_2/I_1)$

$I_1 = 3$

$I_2 = 1$

$w' = \frac{I_1}{I_2}w = 3w$

w

w

w

I : importance, w: particle weight
Particle survives with probability $I_2/I_1 = 1/3$
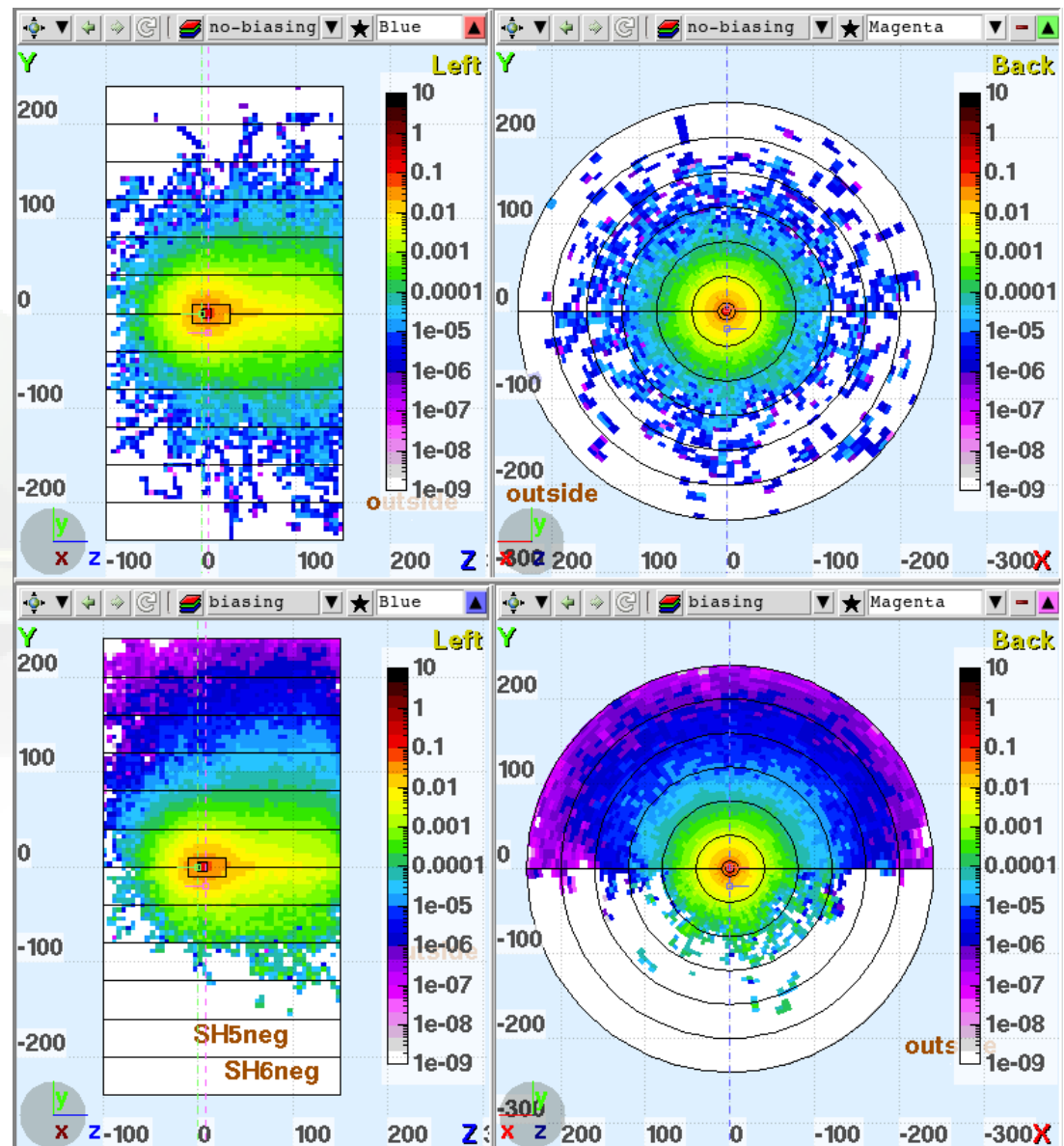Surviving particle weight increased by $I_1/I_2 = 3$

Importance biasing - 4

Example of region
importance biasing
capability

Top:
no biasing applied

Bottom:
Surface splitting &
Russian roulette
both applied

Importance biasing – 5

Problems

Although importance biasing is relatively easy and safe to use, there are a few cases where caution is recommended, *e.g.:*

F I=?	I=16	E
	I=8	D
	I=4	C
	I=2	B
	I=1	A

Which importance shall we assign to region F? Whatever value we choose, we will get inefficient splitting/RR at 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

This applies in part also to muons: the differentiation provided by multiple scattering and by dE/dx fluctuations is not always sufficient

Importance biasing - 6   Input card: **BIASING**

BIASING	0.0	0.0	4.64	Region1	Region2	2.PRINT

```
BIASING              Type: All particles ▼  RR: 0.0              Imp: 4.64
  Opt: PRINT ▼         Reg: Region1 ▼    to Reg: Region2 ▼      Step: 1.
```

The meaning of WHAT(1)…WHAT(6) and SDUM is different depending on the sign of WHAT(1):

If WHAT(1) >= 0.0 :

> **WHAT(1) specifies the particles to be biased**

> > = 0.0 : all particles
> > = 1.0 : hadrons and muons
> > = 2.0 : electrons, positrons and photons
> > = 3.0 : low energy neutrons

> **WHAT(2) =** *(see multiplicity tuning)*

> **WHAT(3) = region importance** (Default = 1.0)
> > Allowed values range from 0.0001 to 100000.

16

Importance biasing - 7

BIASING	0.0	0.0	4.64	Region1	Region2	2.PRINT

```
BIASING              Type: All particles ▼  RR: 0.0          Imp: 4.64
    Opt: PRINT ▼        Reg: Region1 ▼      to Reg: Region2 ▼   Step: 1.
```

If WHAT(1) >= 0.0 :

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

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

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

SDUM   **= PRINT :**   importance biasing counters are printed
             = NOPRINT: counters are not printed
                       (cancels any previous PRINT request)
             = USER: importance biasing
                       according to the user defined routine USIMBS
             = NOUSER: reset to default (cancels any previous USER request)
             = RRPRONLY: multiplicity biasing for primary particles only
             = (blank):   ignored

(Default: NOPRINT, NOUSER, multiplicity biasing for all generations)

Importance biasing – 8   Input card: **BIASING**

BIASING		-1.0	0.0	ELECTRON	POSITRON	0.0	0.0

```
BIASING              Type:All regions ▼ Mod. M0.0
  Opt: ▼              Part:ELECTRON ▼  to Part:POSITRON ▼    Step:0.0
```

If WHAT(1) < 0.0 :

      WHAT(1) : flag indicating that all region importances shall be modified by a particle-dependent factor

      WHAT(2) >= 0.0 : modifying factor M
           < 0.0 : M is reset to the default value 1.0

      WHAT(3) = lower bound of the particle indexes/names (Default: = 1.0)

      WHAT(4) = upper bound of the particle indexes/names
           (Default: = WHAT(3) if WHAT(3) > 0, all particles otherwise)

      WHAT(5) = step length in assigning particle indexes (Default: 1.0)

      WHAT(6) = not used

      SDUM   = PRIMARY :   importance biasing is applied also to primaries
             NOPRIMARy : importance biasing is applied only to secondaries
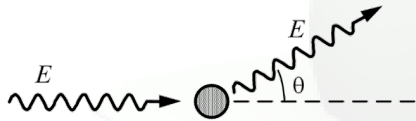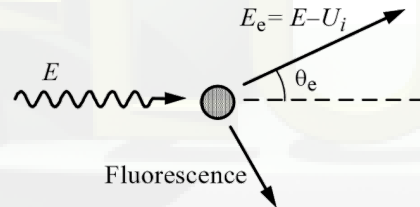   Default = PRIMARY
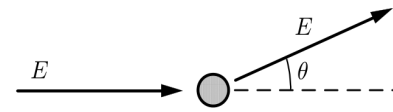
Leading particle biasing

Leading particle biasing - 1

- 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 an electromagnetic shower
- It is characteristic of EM interactions that two particles are present in the final state (at least in the approximation made by most MC codes)
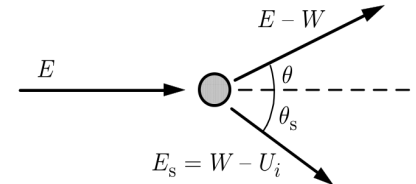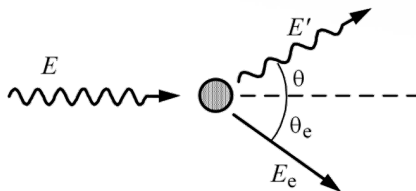
Rayleigh scattering	Photoelectric absorption	**Elastic scattering**	**Inelastic scattering**
Compton scattering	Pair production	**Bremsstrahlung emission**	**Positron annihilation**
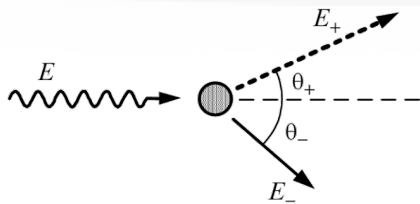
20
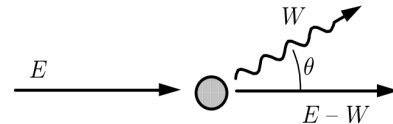
Leading particle biasing - 2

- 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 an electromagnetic shower
- It is characteristic of EM interactions that two particles are present in the final state (at least in the approximation made by most MC codes)
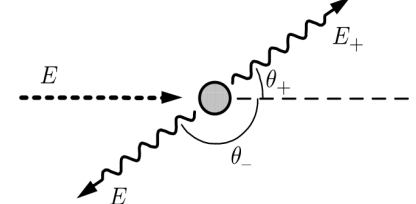- Only one of the two is randomly retained (probability is proportional to energy) and its weight is adjusted so as to conserve weight x probability
- The most energetic of the two particles is kept with higher probability (as it is the one which is more efficient in propagating the shower)
- Leading particle biasing is very effective at reducing t but increases σ by introducing weight fluctuations. If its application is not limited below a suitable energy threshold, it should be backed up by weight windows

21

Leading particle biasing - 3

```
EMF-BIAS        1022.        0.      5.E-4  Region1  Region20        2.LPBEMF
```


For SDUM = LPBEMF (default):

WHAT(1) > 0.0: leading particle biasing (LPB) is activated

$$\text{WHAT}(1) = 2^0 \times b0 + 2^1 \times b1 + 2^2 \times b2 + 2^3 \times b3 + 2^4 \times b4 + 2^5 \times b5 + 2^6 \times b6 + 2^7 \times b7$$

```
(b0 = 1 : LPB for bremsstrahlung and pair production)
 b1 = 1 : LPB for bremsstrahlung
 b2 = 1 : LPB for pair production
 b3 = 1 : LPB for positron annihilation at rest
 b4 = 1 : LPB for Compton scattering
 b5 = 1 : LPB for Bhabha & Moller scattering
 b6 = 1 : LPB for photoelectric effect
 b7 = 1 : LPB for positron annihilation in flight
```

Note: WHAT(1) = 1022 activates LPB for all physical effects
(values larger than 1022 are converted to 1022)

< 0.0: leading particle biasing is switched off
= 0.0: ignored

Leading particle biasing – 3

```
EMF-BIAS       1022.        0.      5.E-4  Region1  Region20        2.LPBEMF
```



```
EMF-BIAS                Type:LPBEMF ▼  Ethr e-e+:0              Ethr γ:5.e-4
bremss.:offBremsstrahlung:On ▼      Pair Prod.:On ▼    e+ ann @rest:On ▼
Compton:On ▼Bhabha&Moller:On ▼ Photo-electric:On ▼  e+ ann @flight:On ▼
                 Reg:Region1 ▼      to Reg:Region20 ▼      Step:2.
```

WHAT(2) > 0.0: energy threshold below which LPB is applied to electrons and positrons
    (electrons: kinetic energy
    positrons: total energy plus rest mass energy)
< 0.0: resets any previously defined threshold to infinity
(i.e., LPB is played at all energies, Default)
= 0.0: ignored

WHAT(3) > 0.0: energy threshold below which LPB is played for photons
< 0.0: resets any previously defined threshold to infinity
(i.e., LPB is applied at all energies, Default)
= 0.0: ignored

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

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

WHAT(6) = step length in assigning indices/names (Default = 1.0)
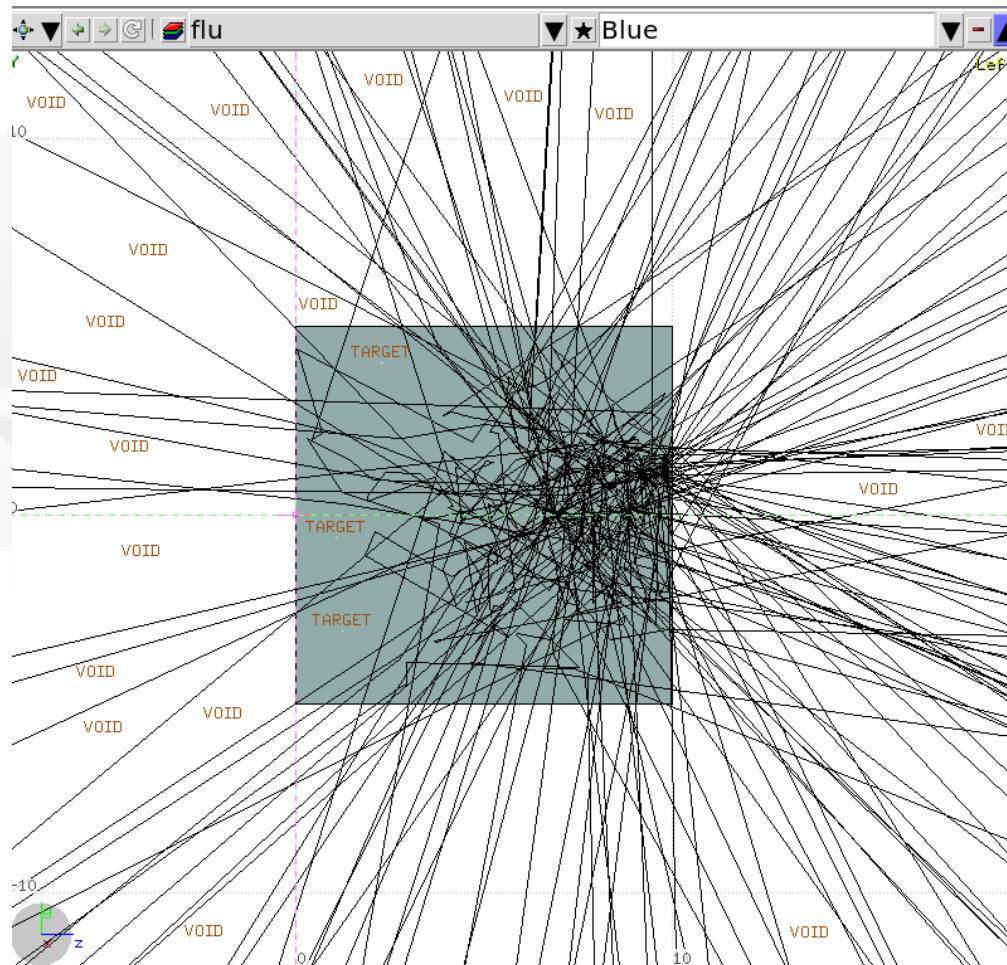
Multiplicity tuning

Multiplicity tuning – 1

Neutrons from a **_single_** simulated history of a 200 GeV p in W

Not shown: photons, protons, pions, kaons

Multiplicity tuning - 2

- Multiplicity tuning is meant to be for hadrons what Leading Particle Biasing is for electrons and photons

- A hadronic nuclear interaction at LHC energies can end in hundreds of secondaries. Thus, to simulate a whole hadronic cascade in bulk matter may take a lot of CPU time

- Contrary to the leading particle biasing case, many secondaries are of the same type and have similar energies and other characteristics

- Therefore, it is possible to discard a predetermined average fraction of them, provided the weight of those kept and transported is adjusted so that the total weight is conserved *(but the leading particle is never discarded)*

- The user can tune the average multiplicity in different regions of space by setting a region-dependent reduction factor *(in theory, it can even be > 1 ! But this possibility is seldom used)*

Input card: **BIASING**

BIASING	1.0	0.7	1.0	Region1	Region20	2.

```
BIASING              Type: Hadrons & Muons  ▼RR: 0.7          Imp: 1.0
  Opt: ▼              Reg: Region1 ▼   to Reg: Region20 ▼     Step: 2.
```

WHAT(1) specifies the particles to be biased with WHAT(3) *(see before)*

　　　　　= 0.0 : all particles
　　　　　= 1.0 : hadrons and muons
　　　　　= 2.0 : electrons, positrons and photons

WHAT(2) = RR (or splitting) factor by which the average number of secondaries produced in a collision should be reduced (or increased). (Default = 1.0)

WHAT(3) = *(see importance biasing)*

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

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

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

SDUM   = RRPRONLY: multiplicity biasing for primary particles only
**   = (blank):** ignored (Default)

Biasing mean free paths

Interaction lengths:

• In a similar way, the hadron or photon mean free path for non-elastic nuclear interactions can be artificially decreased by a predefined particle- or material-dependent factor.

• This option is useful for instance to increase the probability for beam interaction in a **very thin target** or in a material of very low density.

• It is also necessary to simulate photonuclear reactions with acceptable statistics, the photonuclear cross section being much smaller than that for EM processes.

Biasing mean free paths – 2

Input card: **LAM-BIAS**

LAM-BIAS		0.0	0.02	TUNGSTEN	PHOTON	0.	0.

```
LAM-BIAS          Type: ▼      × mean life: 0.0   × λ inelastic: 0.02
  Mat: TUNGSTEN ▼  Part: PHOTON ▼   to Part: ▼              Step: 0.0
```

WHAT(1): *(more relevant for decay length biasing)*

WHAT(2) : biasing factor for hadronic inelastic interactions

The hadronic inelastic interaction length of the particle is reduced by a factor |WHAT(2)| (must be <= 1.0)

< 0. : At the interaction, a **Russian Roulette decides whether the particle actually will survive or not**

> 0. : T**he particle always survives after the interaction but with a reduced weight**

In either case, the secondaries are generated and their weight is adjusted taking into account the ratio between the interaction and biased probabilities

LAM-BIAS	0.0	0.02	TUNGSTEN	PHOTON	0.	0.

```
LAM-BIAS          Type: ▼      × mean life:0.0  × λ inelastic:0.02
  Mat:TUNGSTEN ▼  Part:PHOTON ▼   to Part:  ▼              Step:0.0
```

WHAT(3) : If > 2.0 : number or name of the material to which the inelastic
biasing factor has to be applied.
< 0.0 : resets to the default a prev. assigned value
= 0.0 : ignored if a value has been previously assigned to a specific
material, otherwise all materials
0.0 < WHAT(3) =< 2.0 : all materials
(Default = 0.0)

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

WHAT(5) = upper bound of the particle indices/names
(Default = WHAT(4) if WHAT(4) > 0, 46 otherwise)

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

31

Summary of main input cards

BIASING

    1) region importance biasing (surface splitting or Russian Roulette)

    2) multiplicity tuning at hadronic interactions

EMF-BIAS

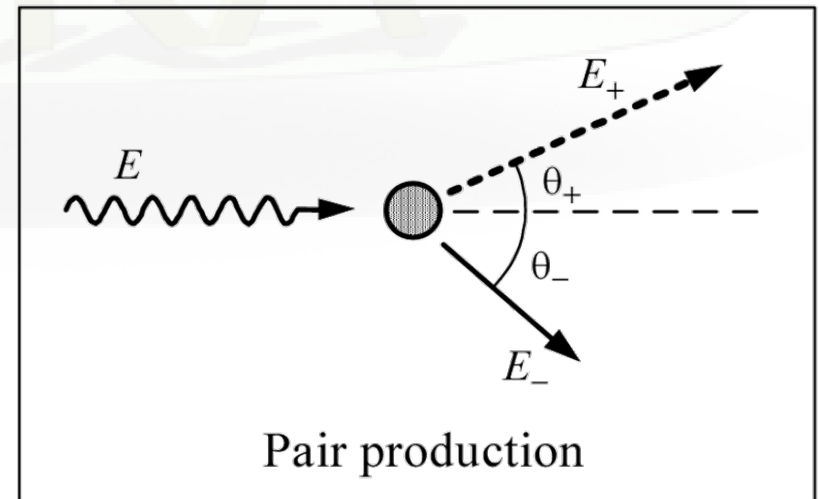   leading particle biasing for e+, e- and photon interactions

LAM-BIAS

    mean free path biasing

        (decay length biasing, hadronic interaction length biasing)

END

Aftermath of a photon interaction

Rayleigh scattering — incoming photon E, scattered photon E at angle θ.

Photoelectric absorption — incoming photon E, electron $E_e = E - U_i$ at angle θ_e, Fluorescence, +Auger

Compton scattering — incoming photon E, scattered photon E' at angle θ, electron E_e at angle θ_e.

Pair production — incoming photon E, positron E_+ at angle θ_+, electron E_- at angle θ_-.
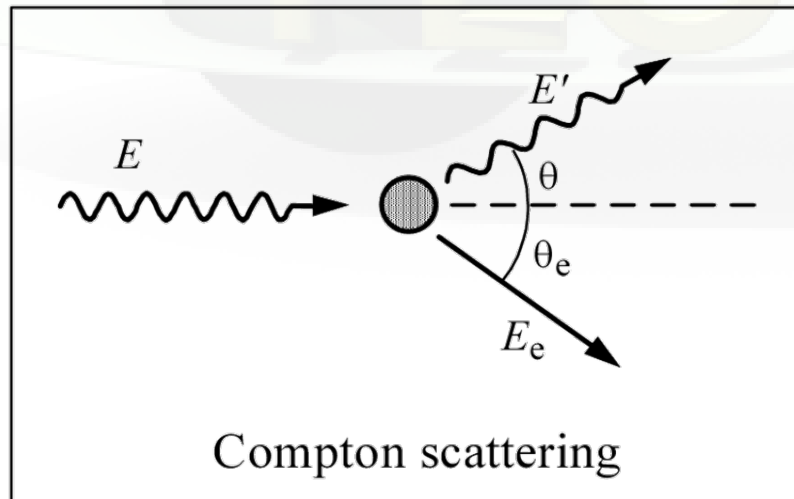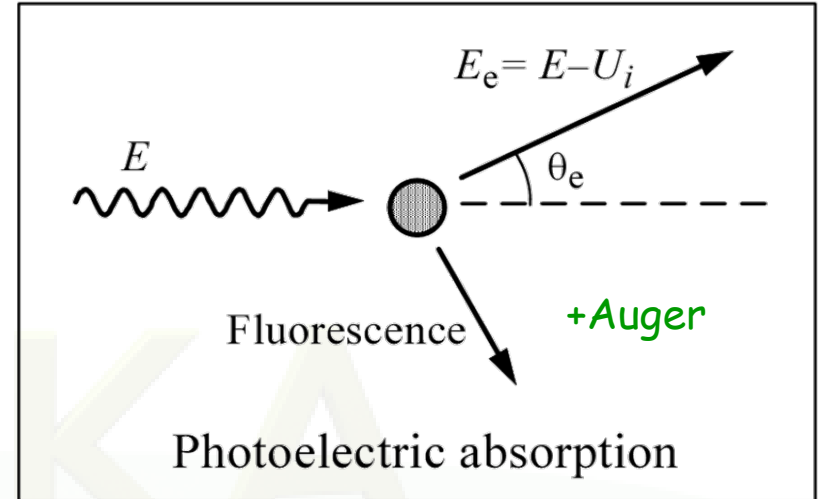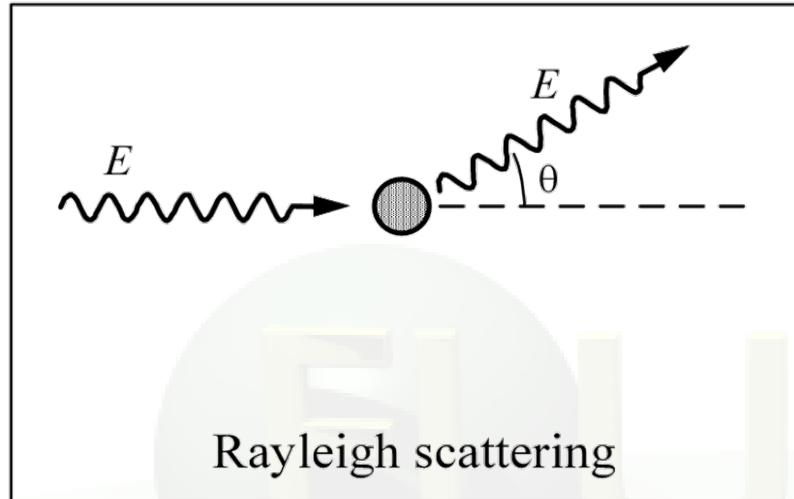
+photo-nuclear processes   +photo-muon production

Aftermath of an e-/e+ interaction
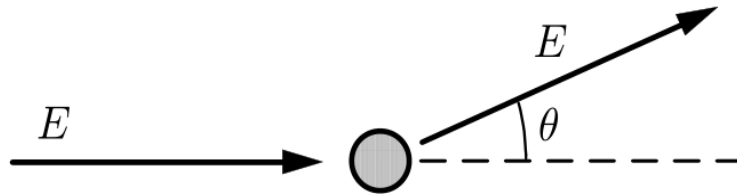
Elastic scattering
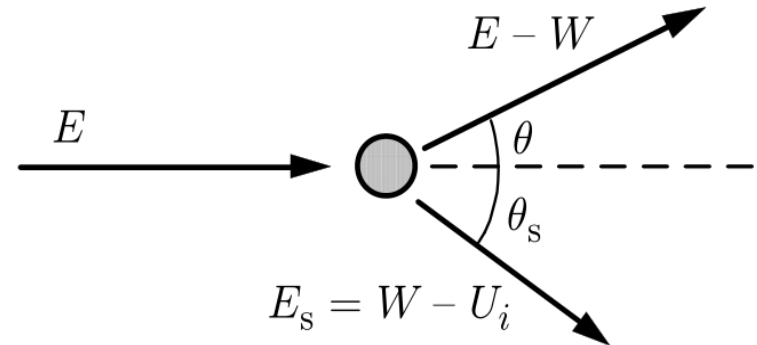
Inelastic scattering
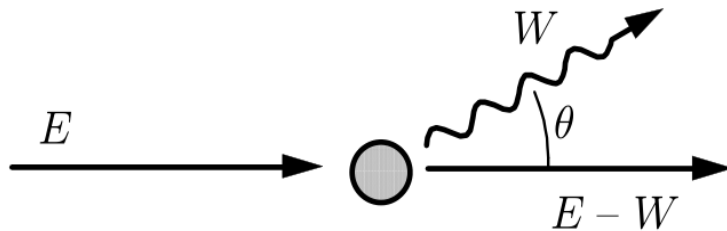
Bremsstrahlung emission
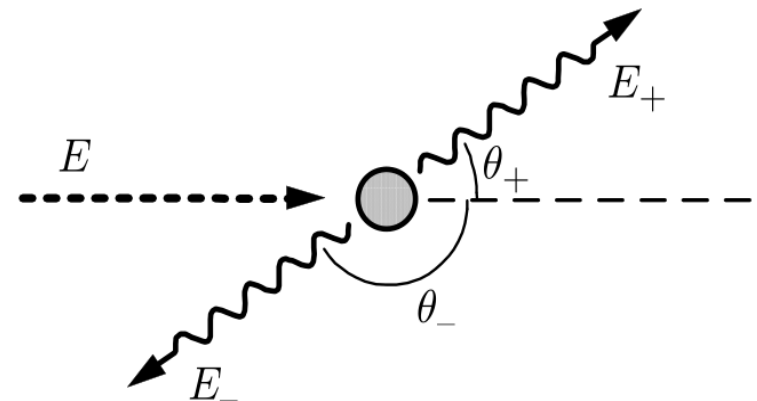
Positron annihilation

Biasing mean free paths – 1

Decay lengths:

- The mean life/average decay length of unstable particles can be artificially shortened.

- It is also possible to increase the generation rate of decay products without the parent particle actually disappearing.

- Typically used to enhance statistics of muon or neutrino production.

- The kinematics of the decay can also be biased (decay angle).

Biasing mean free paths – 2

Input card: **LAM-BIAS**

LAM-BIAS	**-3.E+3**	**1.**	**1.**	**PION+**	**KAON-**	**0.GDECAY**

for SDUM = GDECAY:

WHAT(1) : **mean decay length (cm) of the particle in the laboratory frame** is set = |WHAT(1)| if smaller than the physical decay length (otherwise it is left unchanged).

< 0.0 : At the decay point **Russian Roulette (i.e. random choice) decides whether the particle actually will survive or not** after creation of the decay products. The latter are created in any case and their weight adjusted taking into account the ratio between biased and physical survival probability.

> 0.0 : Let P_u = unbiased probability and P_b = biased probability: at the decay point **the particle always survives with a reduced weight W=(1-P_u/P_b).** Its daughters are given a weight **W=P_u/P_b** (as in case WHAT(1) < 0.0).

LAM-BIAS	-0.5	1.	1.	PION+	KAON-	0.

◇LAM-BIAS	Type: ▼	x mean life:	x λ inelastic:
Mat: ▼	Part: ▼	to Part: ▼	Step:

for SDUM = blank:

WHAT(1) : the mean life of the particle in its rest frame is reduced by a factor |WHAT(1)| (must be <= 1.0)

< 0.0 : *(as for SDUM=GDECAY)* **Russian Roulette**

> 0.0 : *(as for SDUM=GDECAY)* **the particle always survives with a reduced weight**

for SDUM = blank or GDECAY

WHAT(2) and WHAT(3) : *(see interaction length biasing)*

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

WHAT(5) = upper bound of the particle index (Default = WHAT(4) if WHAT(4) > 0, 46 otherwise)

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

User-written biasing

ubsset.f : User BiaSing SETting

- called after reading in the input file and before first event
- allows to alter almost any biasing weight on a region-dependent basis

For example: region number   region importances

```
SUBROUTINE UBSSET (      IR,  RRHADR,  IMPHAD,  IMPLOW,  IMPEMF,
     &                     IGCUTO,  IGNONA,  PNONAN,  IGDWSC,  FDOWSC,
     &                     JWSHPP,   WWLOW,    WWHIG,   WWMUL,  EXPTR ,
     &                     ELECUT,  GAMCUT,   LPEMF,  ELPEMF,  PLPEMF )
```

usimbs.f: User defined IMportance BiaSing

- called at *every* particle step (!)
- allows to implement any importance biasing scheme based on region number and/or phase space coordinates
- enabled with `BIASING/SDUM=USER`

region number at beginning and end of step   ratio of region importances

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

udcdrl.f: User defined DeCay DiRection biasing and Lambda

- only for neutrinos emitted in decays: bias on direction of emitted neutrino

39

Weight Windows - 1

The weight window technique is a combination of splitting and Russian Roulette, 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 weights smaller than the lower limit are submitted to Russian Roulette (killed or put back "inside the window").

Weight windows are a more powerful biasing tool than importance biasing, but they require also more experience and patience to set it up correctly. "It is more an art than a science" (Quote from the MCNP manual)

The use of weight windows is essential whenever other biasing techniques generate large weight fluctuations in a given phase space region.

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

Splitting

W2

constant window
for E<E1

gradually increasing
(or decreasing) window
for E1<E<E2

no window
for E>E2

W1

Russian Roulette

E1

E2

Energy
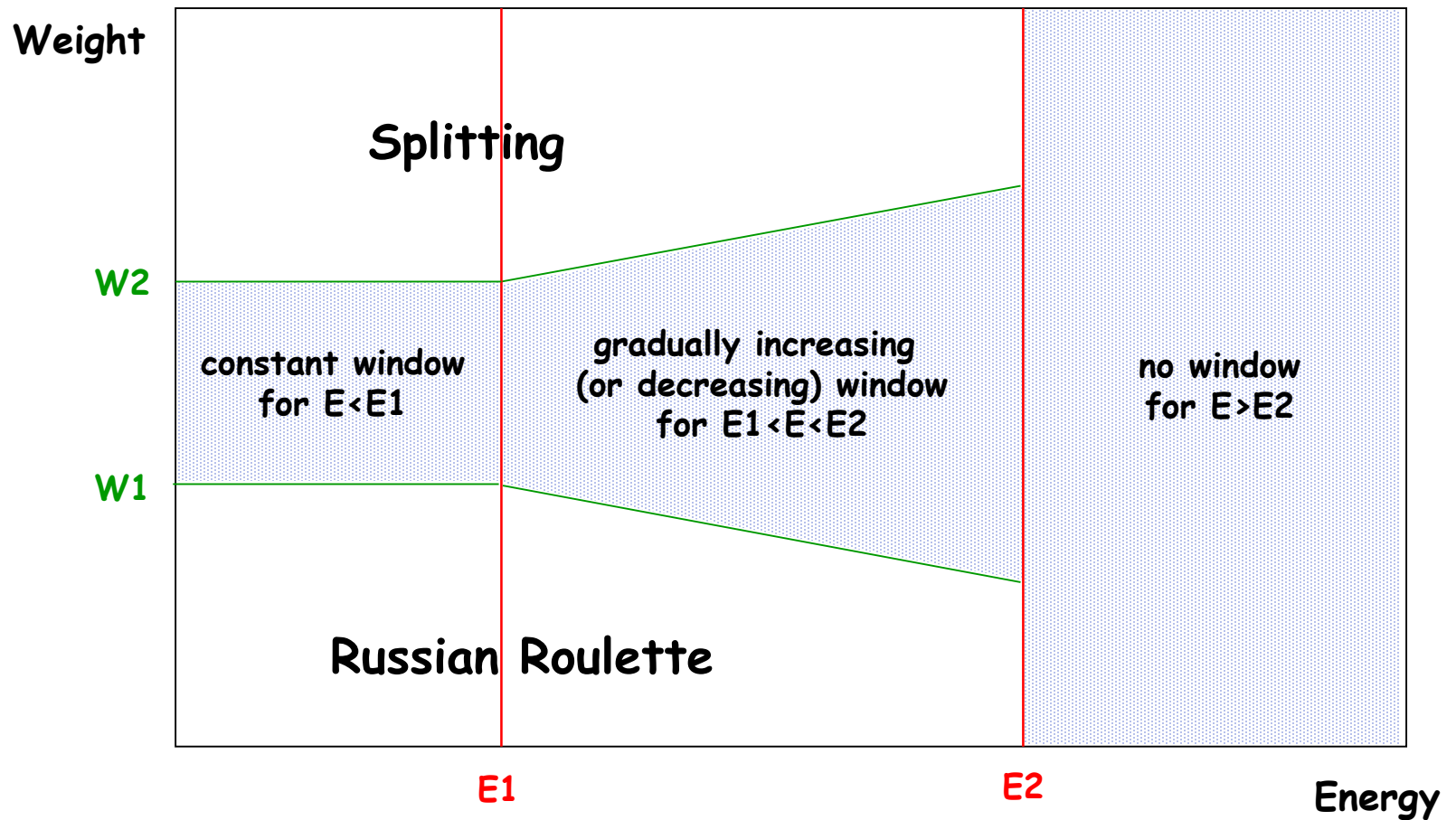
Weight Windows - 4

region dependent
particle type dependent

Weight

W2 × WW-THRES WHAT(3)

WW-FACTO WHAT(2) W2

WW-FACTO WHAT(1) W1

W1 / WW-THRES WHAT(3)

E1

E2

WW-THRES WHAT(2) × WW-FACTO WHAT(3)

WW-THRES WHAT(1) × WW-FACTO WHAT(3)

Energy

43

| WW-FACTO | 13.0 | 120.0 | 1.5 | Region27 | Region31 | 2.0 |

```
WW-FACTO                   RR:                    Split:                    mult f:
  LowE n: ▼                Reg: ▼                 to Reg: ▼                  Step:
```

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)

44

WW-FACTO	13.0	120.0	1.5	Region27	Region31	2.0

```
WW-FACTO                    RR:                    Split:                    mult f:
LowE n: ▼                   Reg: ▼                 to Reg: ▼                 Step:
```

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

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

WHAT(6) = step length in assigning indices/names (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.

WW-THRES	2.0	0.05	2.4	ELECTRON	PHOTON	0.0

```
WW-THRES              E upper:                    E lower:                    amp f:
   Opt: ▼             Part: ▼                     to Part: ▼                  Step:
```

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)

WW-THRES	2.0	0.05	2.4	**ELECTRON**	**PHOTON**	0.0

WW-THRES	E upper:	E lower:	amp f:
Opt: ▼	Part: ▼	to Part: ▼	Step:

WHAT(4) = lower bound of the particle indices/names (Default = 1.0)
Note that particle index 40 indicates low-energy neutrons (for this purpose only!). Particle index 8 indicates neutrons with energy > 20 MeV.

WHAT(5) = upper bound of the particle indices/names
(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    Region8   Region18        2.PRINT
```

```
BIASING              Type: ▼              RR:              Imp: 0.000189
   Opt: PRINT ▼        Reg: ▼           to Reg: ▼          Step:
```

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

48

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   Region1   Region9          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   Region1   Region9          PRINT
BIASING           0.0        1.0       1.47   Region4
BIASING           0.0        1.0       2.15   Region5
BIASING           0.0        1.0       3.16   Region6
BIASING           0.0        1.0       4.64   Region7
BIASING           0.0        1.0       4.64   Region8
```

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

END