Biasing

7th FLUKA Course

NEA Paris, Sept.29-Oct.3, 2008

Overview

General concepts:

Analog vs. biased Monte Carlo calculation

Biasing options

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

Importance biasing
Weight Windows
Leading particle biasing
Multiplicity tuning
Biased down-scattering
Non-analogue neutron absorption
Biasing mean-free paths - decay lengths biasing
- hadronic inelastic interaction lengths
User-written biasing

Analog 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

Analog 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**

Reduce variance or CPU time ?

A Figure of Merit

Computer cost of an estimator = $\sigma^2 \times t$

(σ^2 = Variance, t =CPU time per primary particle)

- some biasing techniques are aiming at reducing σ^2, others at reducing t
- often reducing σ^2 increases t, and *viceversa*
- therefore, minimizing $\sigma^2 x\ t$ means to reduce σ at a faster rate than increases or *viceversa*
- 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

Importance biasing - 1

Input card: **BIASING**

- it is the simplest, most "safe" 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 - 2

Input card: **BIASING**

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 replaced on average by $n = I_2/I_1$ identical particles with the same characteristics
- the weight of each "daughter" is multiplied by I_1/I_2

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

Input card: **BIASING**

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 is submitted to a random survival test: with a chance I_2/I_1 the particle survives with its weight increased by a factor I_1/I_2
- with a chance $(1-I_2/I_1)$ the particle is killed

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 - 4

Input card: **BIASING**

Note:

In FLUKA, for technical reasons, importances are internally stored as integers. Therefore, importances can only take values between 0.0001 and 100000. An input values 0.00015 is read as 0.0001, 0.00234 is read as 0.0023, *etc.*

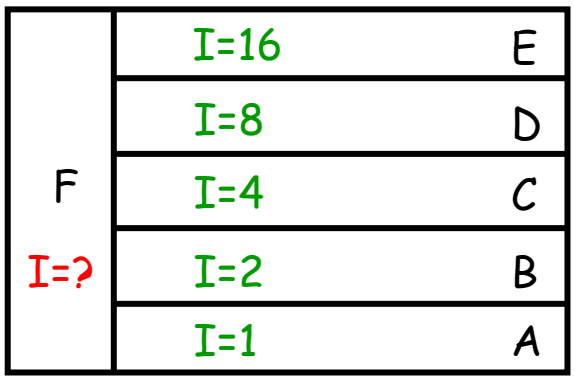
There is also a user routine **USIMBS** which allows to assign importances not only at boundaries, but at each step, according to any logic desired by the user (as a function of position, direction, energy,…).
Very powerful, but ***time-consuming*** (it is called at each step!). The user must balance the time gained by biasing with that wasted by calls.

Importance biasing - 5

Input card: **BIASING**

Problems

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



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. (Difficult to do in ducts and mazes)

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**	**8.**	**18.**	**2.PRINT**

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 10000.

Importance biasing - 7

Input card: **BIASING**

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

If WHAT(1) >= 0.0 :

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 = **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 gen.)

Importance biasing - 8

Input card: **BIASING**

```
BIASING          2.0       0.0      10.0       7.0      11.0       2.0
BIASING          2.0       0.0      15.0       8.0       9.0       0.0
BIASING         -1.0       0.0       3.0       4.0       0.0       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 index (Default: = 1.0)

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

WHAT(5) = step length in assigning particle indices (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

Weight Windows - 1

Input cards: **WW-FACTO**
WW-THRES
WW-PROFI

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

Input cards: **WW-FACTO**
WW-THRES
WW-PROFI

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

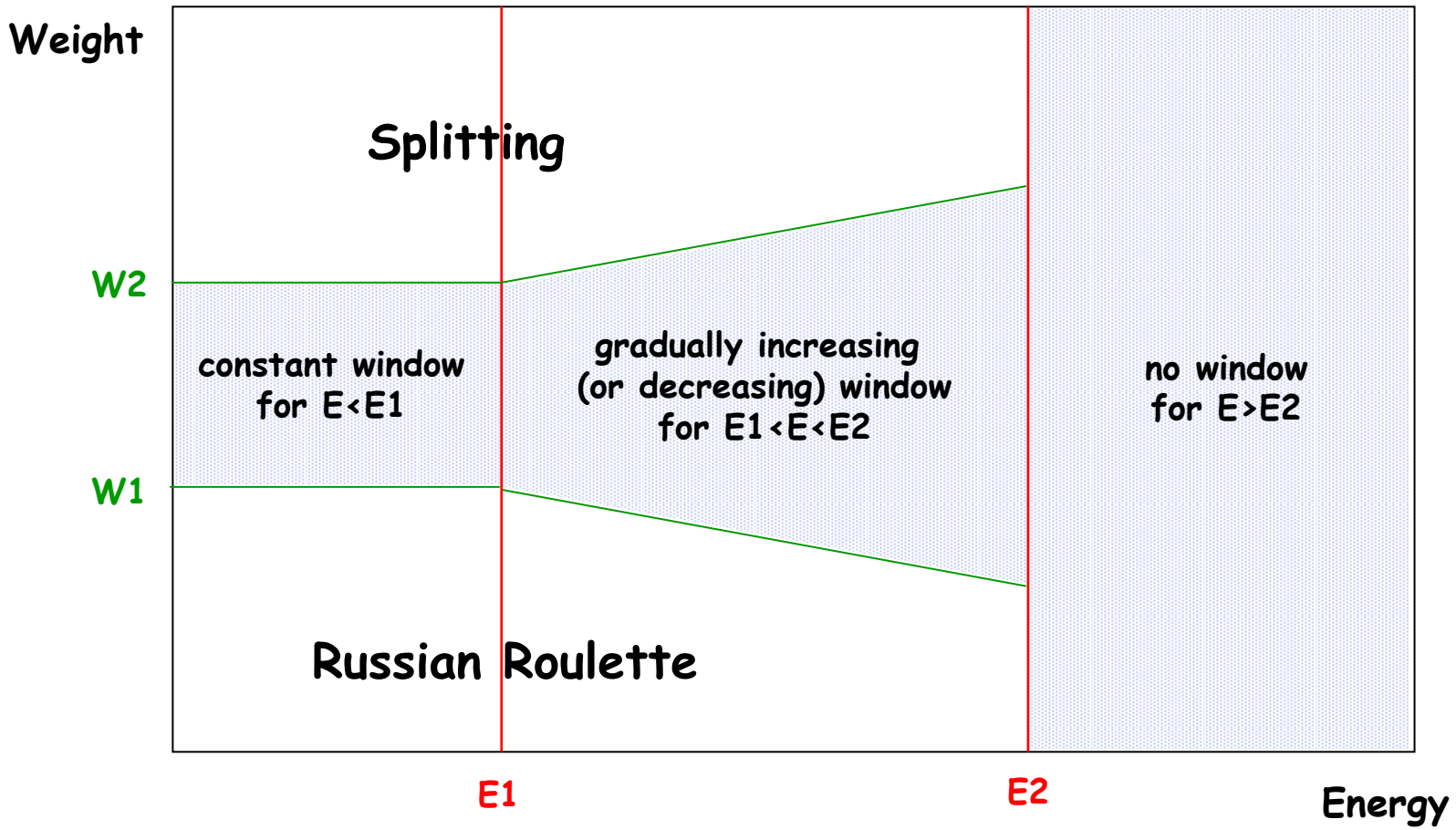
- *increases* ***t*** (in proportion to the number of additional particles to be followed)
- but at the same time *reduces* σ by avoiding large fluctuations in the contributions to scoring.

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

Input cards: WW-FACTO
WW-THRES
WW-PROFI



Weight Windows - 4

Input card: **WW-FACTO**

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

Input card: **WW-FACTO**

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

Input card: **WW-THRES**

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

Input card: **WW-THRES**

```
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)

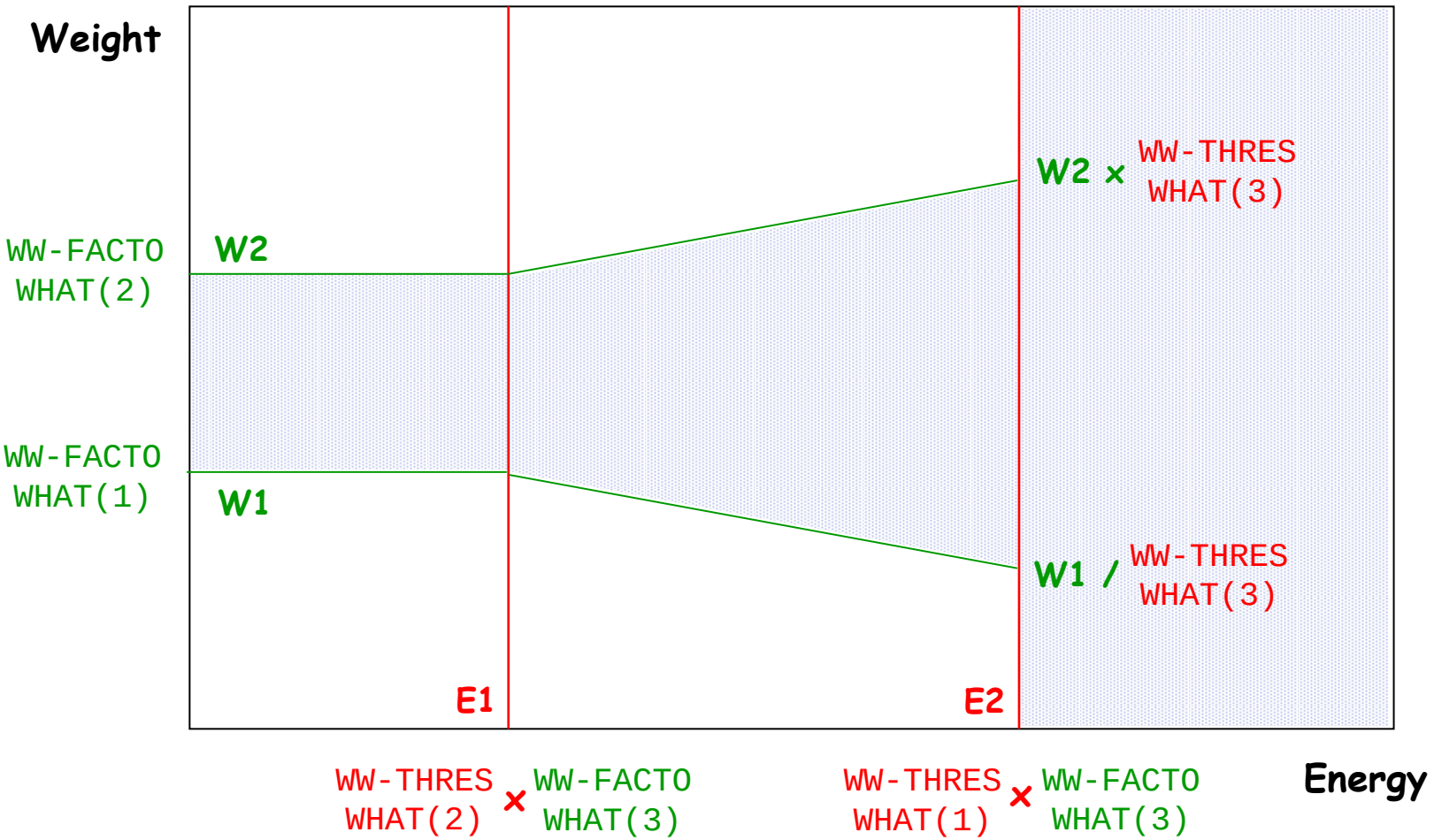
WHAT(6) = step length in assigning indices (Default = 1.0)

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

Weight Windows - 8

Input cards: **WW-FACTO**
WW-THRES

region dependent
particle type dependent



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

Leading particle biasing - 1

Input card: **EMF-BIAS**

- 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 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 this 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 large weight fluctuations. Therefore, it should nearly always be backed up by weight windows.

Leading particle biasing - 2

Input card: **EMF-BIAS**

EMF-BIAS	1022.	0.	5.E-4	16.	20.	2.LPBEMF

For SDUM = LPBEMF (default):

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

WHAT(1) = 2^0xb0 + 2^1xb1 + 2^2xb2 + 2^3xb3 + 2^4xb4 +
2^5xb5 + 2^6xb6 + 2^7xb7

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

Input card: **EMF-BIAS**

EMF-BIAS	1022.	0.	5.E-4	16.	20.	2.LPBEMF

WHAT(2) > 0.0: energy threshold below which LPB is played for 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 played at all energies, Default)
= 0.0: ignored

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)

Multiplicity tuning - 1

Input card: **BIASING**

- 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.
- Except for the leading particle, 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 which are kept and transported be adjusted so that the total weight is conserved *(but the leading particle must not be discarded).*
- The user can tune the average multiplicity in different regions of space by setting a region-dependent reduction factor *(in fact, it can even be > 1 ! But this possibility is seldom used).*

Multiplicity tuning - 2

Input card: **BIASING**

```
BIASING          1.0       0.7       1.0       8.       18.       1.
```

WHAT(1) specifies the particles to be biased

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

Non-analog neutron absorption - 1

Input card: **LOW-BIAS**

- Also called survival biasing
- It is implemented in most low-energy neutron transport codes, were the user must choose between two options:
  - at each neutron collision either analog scattering or absorption according to the actual physical probability σ_s/σ_T and $(1- \sigma_s/\sigma_T)$, respectively
  - systematic survival with a weight reduced by a factor σ_s/σ_T
- In FLUKA there is also a third possible choice: the user can force the neutron absorption probability to take an arbitrary value, pre-assigned on a region-by-region basis and as a function of energy.

Non-analog neutron absorption - 2

Input card: **LOW-BIAS**

When and how to use it ?

- A small survival probability is often assigned to thermal neutrons to limit the number of scatterings in non-absorbing media.
- It is also very useful in material with unusual scattering properties (*e.g.*, iron)
- Survival probabilities too small with respect to the physical one σ_s/σ_T may introduce large weight fluctuations due to the very different number of collisions suffered by individual neutrons: in these cases a Weight Window should be applied. *(Never set the absorption probability larger than a few times the physical one !!)*

Non-analog neutron absorption - 3

Input card: **LOW-BIAS**

LOW-BIAS	**60.0**	**47.0**	**0.95**	**5.0**	**19.0**	**0.0**

WHAT(1) > 0.0 : group cut-off (Default = 0.0, i.e., no cut-off)

Neutrons in energy groups with number >= WHAT(1) are not transported.

WHAT(2) > 0.0 : group limit for non-analogue absorption
(Default: depends on DEFAULTS setting)

Neutrons in energy groups >= WHAT(2) undergo non-analogue absorption. If WHAT(2)> NMGP (=72) analogue absorption is applied to all groups.

WHAT(3) > 0.0 : non-analogue survival probability 1 - ($\sigma_{abs}/\sigma_{tot}$)
(Default: depends on DEFAULTS setting)

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.)

Biasing mean free paths – 1

Input card: **LAM-BIAS**

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.	13.	16.	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** $W=(1 - P_u/P_b)$, where W is the current weight of the particle before the decay. Its daughters are given a weight $W=P_u/P_b$ (as in case WHAT(1) < 0.0).

Biasing mean free paths – 3

Input card: **LAM-BIAS**

LAM-BIAS	-0.5	1.	1.	13.	16.	0.

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=DECAY)* **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)

Biasing mean free paths – 4

Input card: **LAM-BIAS**

Interaction lengths:

- In a similar way, the hadron or photon mean free path for 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 that that for EM processes.

Biasing mean free paths – 5

Input card: **LAM-BIAS**

LAM-BIAS	**0.0**	**0.02**	**11.**	**7.**	**0.**	**0.INEPRI**

WHAT(1): *(see 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 point **Russian Roulette (i.e. random choice) decides whether the particle actually will survive or not** after creation of the secondaries products. The latter are created in any case and their weight adjusted taking into account the ratio between biased and physical survival probability.

> 0. : At the interaction point **the particle always survives with a reduced weight.** The secondaries are created in any case and their weight adjusted taking into account the ratio between biased and physical survival probability.

Biasing mean free paths – 6

Input card: **LAM-BIAS**

LAM-BIAS	**0.0**	**0.02**	**11.**	**7.**	**0.**	**0.**

WHAT(3) : If > 2.0 : number 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 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 - 1

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