Biasing techniques in FLUKA

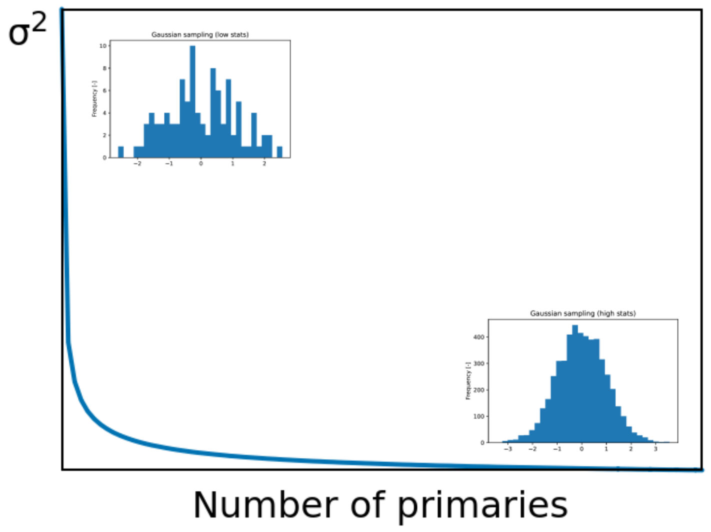
Beyond basics

What is biasing?

- It is the use of “*Variance Reduction Technique*(s)” that…

  distort distributions and apply weights to particles to correct for the bias

- VRTs aim at reducing variance σ^2 or CPU time t



- Usually, reducing one quantity increases the other

- Usually, more than one VRT is applied at the same time

- Goodness of simulations can be estimated with a Figure of Merit: $FOM=1/(\sigma^2 \cdot t)$

  the larger the better: less time and smaller uncertainty

Non-biased Monte Carlo simulations

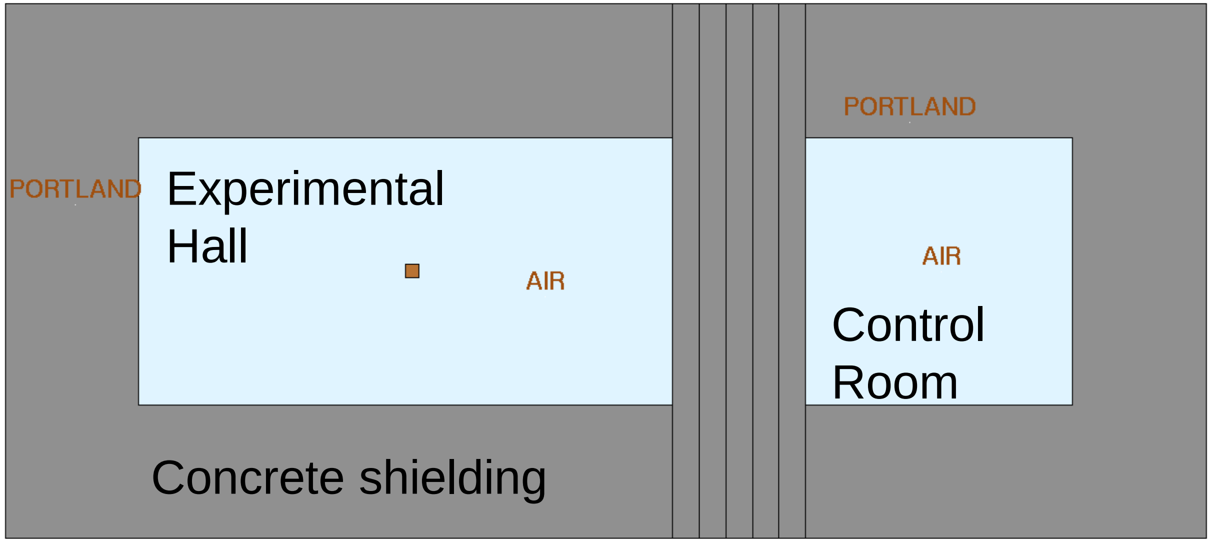
Characteristics

- Samples uniformly from

  the phase-space distribution
- Preserves correlations
- Reproduces fluctuations

Drawbacks

- Converges slowly
- Rare events are… “rare”



Non-biased Monte Carlo simulations

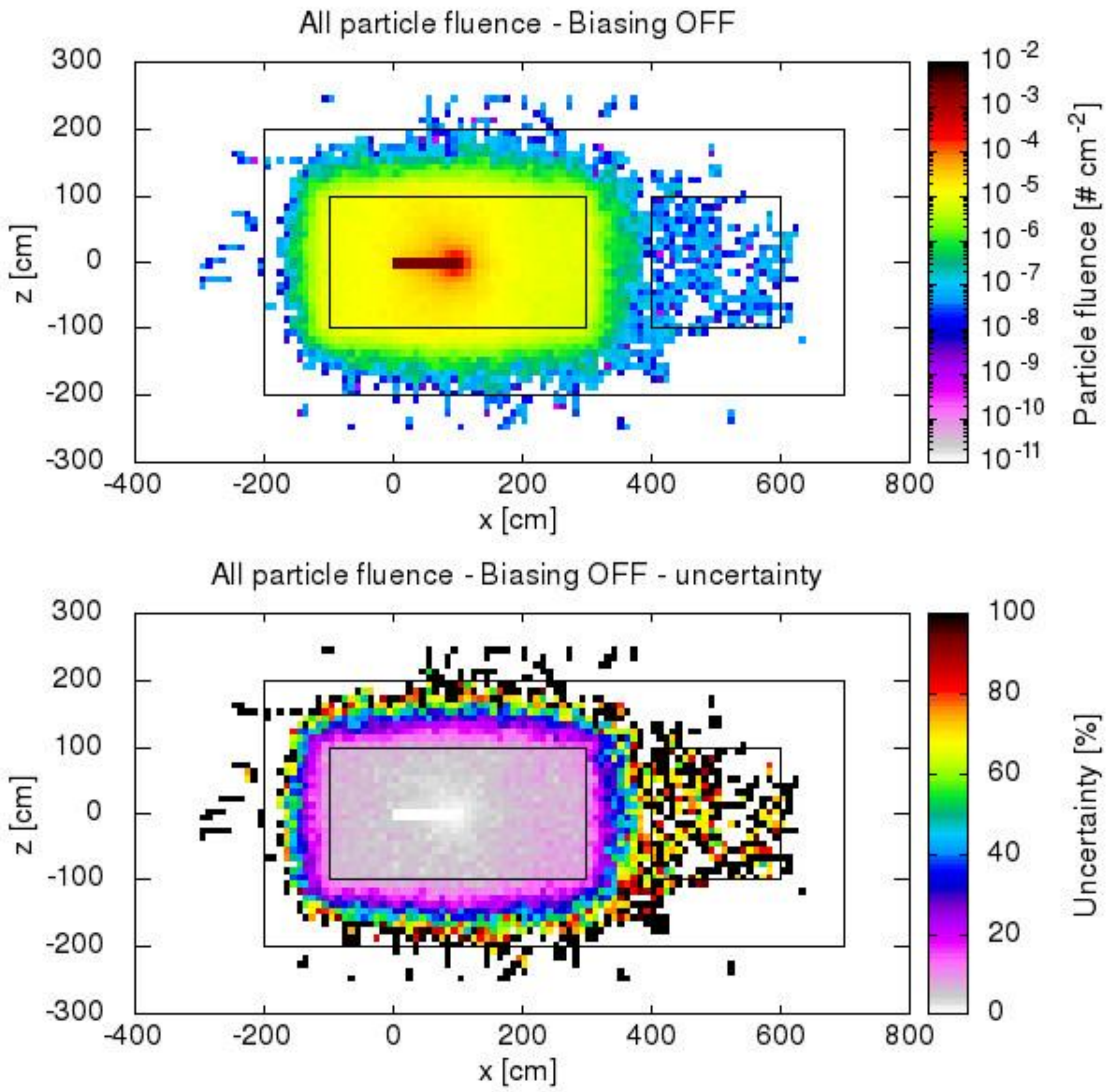
200000 primaries

Characteristics

- Samples uniformly from
  the phase-space distribution
- Preserves correlations
- Reproduces fluctuations

Drawbacks

- Converges slowly
- Rare events are… “rare”



Biased Monte Carlo simulations

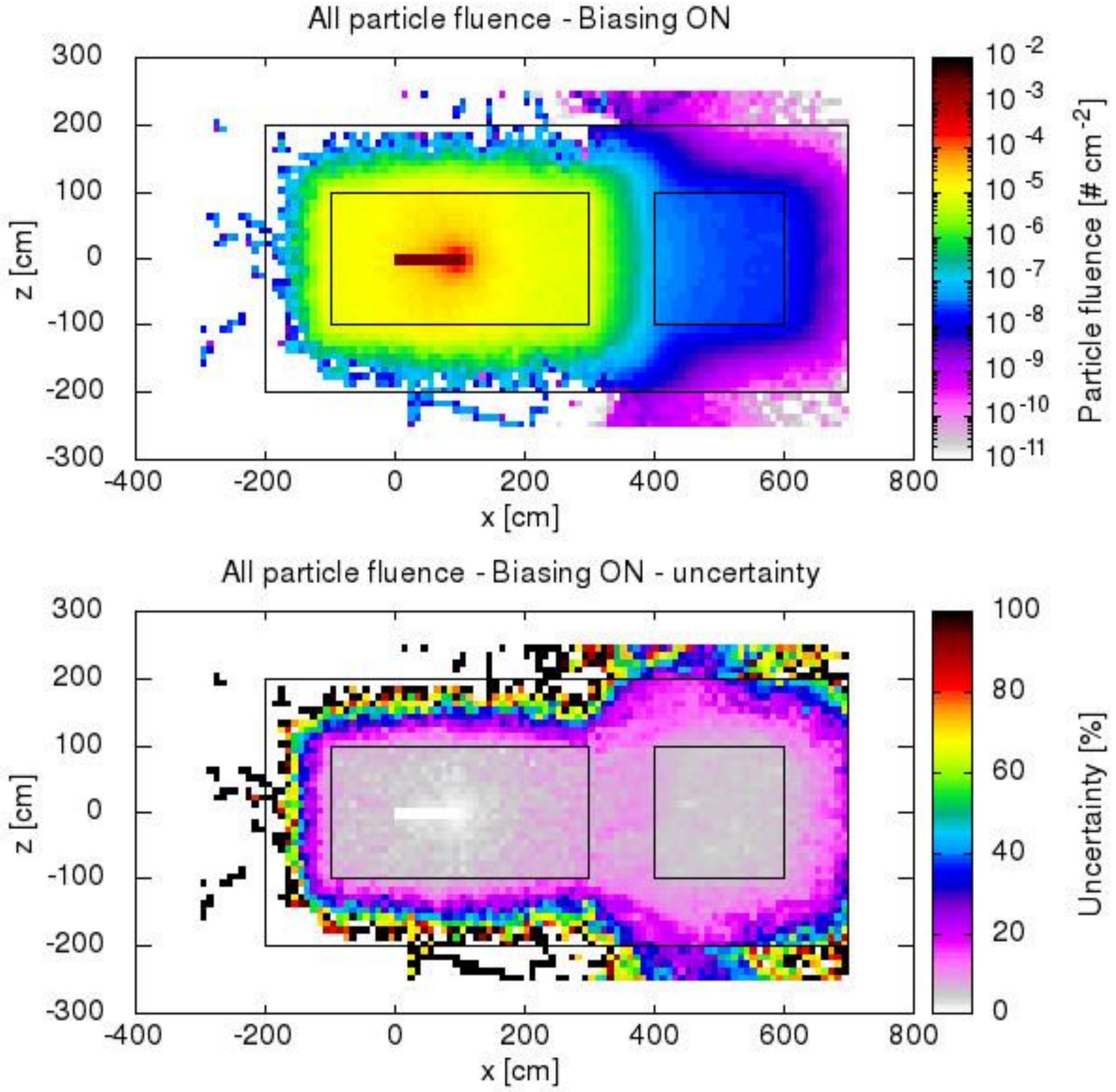
200000 primaries

Characteristics

- Samples from distorted distributions
- Converges "quickly"

Drawbacks

- Cannot reproduce fluctuations and correlations
- Requires active reasoning and experience
- Requires user's time to be implemented



Biasing techniques in FLUKA

- Region Importance Biasing (**BIASING**)
- Mean Free Path Biasing (**LAM-BIAS**)
- *Leading Particle Biasing* (**EMF-BIAS**)
- *Multiplicity Tuning* (**BIASING**)
- *Lifetime* (**LAM-BIAS**)
- *Decay-length Biasing* (**LAM-BIAS**)
- *Weight Windows* (**WW-FACTO**, **WW-THRES**)
- *User defined biasing* (**BIASING** , usimbs.f)

Discussed in the beginners' course

Discussed here

Leading Particle Biasing

Input card: **`EMF-BIAS`**

Leading Particle Biasing

- Input card: **EMF-BIAS**
- Applies only to electromagnetic interactions of electrons, positrons, and photons
- Interaction processes to be affected are selected one by one (see next slide)
- Two electromagnetic particles in the final state, only one is retained
- Applies only within selected regions
- Survival probability p proportional to energy
- Weight is adjusted $w' = w / p$
- Generally used to speed up simulations of electromagnetic showers
- Few surviving low-energy particles might generate strong fluctuations
- Multiple **EMF-BIAS** cards are allowed

Leading Particle Biasing

Input card: **`EMF-BIAS`**

- *Type*
  - LPBEMF *Leading Particle Biasing for ElectroMagnetic interaction in Fluka*
  - LAMBEMF LAMbda Biasing for ElectroMagnetic Fluka
  - LAMBBREM LAMbda Biasing for BREMsstrahlung interactions
  - LBRREMF Lambda Biasing w/ Russian Roulette for ElectroMagnetic Fluka
  - LBRRBREM Lambda Biasing w/ Russian Roulette for BREMsstrahlung interactions

  - Lambda Biasing is not discussed in the course, details are found in the manual

Leading Particle Biasing

Input card: **EMF-BIAS**

- *Type*
  - LPBEMF
  - …
- *Ethr e-e+ - Ethr γ*
  - Threshold below which LPB applies
    - For electrons: E = kinetic energy
    - For positrons: E = total energy (kinetic energy plus rest mass energy)
- *Processes to which LPB applies*
  - Self-explanatory
  - “Old bremss.” is a relic of the past for backward compatibility
- *Reg - to Reg - Step*
  - Standard FLUKA region selection

Leading Particle Biasing

Input card: **EMF-BIAS**

- *Type*
  - LPBEMF
- *Ethr e-e+ - Ethr γ*
  - Threshold below which LPB applies (*)
- *Processes to which LPB applies*
  - Self-explanatory
- *Reg - to Reg - Step*
  - Standard FLUKA region selection

<u>Example explanation</u>:
LPB is applied during *bremsstrahlung* and *pair production* processes, within *every other region (step=2)* between *region=a2* and *region=a8*, to *photons*, *electrons*, and *positrons* below a *20 MeV (*) energy threshold*

EMF-BIAS	Type: LPBEMF ▾	Ethr e-e+: 0.02	Ethr γ: 0.02
Old bremss.: off ▾	Bremsstrahlung: On ▾	Pair Prod.: On ▾	e+ ann @rest: off ▾
Compton: off ▾	Bhabha&Moller: off ▾	Photo-electric: off ▾	e+ ann @flight: off ▾
	Reg: a2 ▾	to Reg: a8 ▾	Step: 2

*Beware of the different thresholds for electrons and positrons (see previous slide)

Multiplicity Tuning

Input card: `BIASING`

Multiplicity Tuning

- Input card: **`BIASING`**
- Large number of secondaries produced in hadronic interactions
- “Leading Particle Biasing for hadrons”, i.e. reduces the number of secondaries
  - Very useful for high-energy heavy ions simulations
- Hadronic secondaries have similar characteristics
- A RR (Russian Roulette) reduction factor is defined
  - E.g. RR=0.5 means that 50% of the secondaries are discarded
- Weight is adjusted
- Applied within selected regions
- Multiple **`BIASING`** cards are allowed

Multiplicity Tuning

Input card: **BIASING**

- *Type*
  - Hadrons&muons
- *Reg - to Reg - Step*
  - Standard FLUKA region selection
- *Imp*
  - Importance of the selected region(s)

Example explanation:
Hadrons and muons secondaries generated within all regions between *region=a2* and *region=a4* have a 50% probability (*RR=0.5*) to survive with doubled particle weight

BIASING Type: **Hadrons & Muons ▾** RR: **0.5** Imp:
Opt: ▾ Reg: **a2 ▾** to Reg: **a4 ▾** Step: **1**

Lifetime Biasing

Input card: **`LAM-BIAS`**

Lifetime Biasing

- Input card: **`LAM-BIAS`**
- Allows to *modify the lifetime of unstable particles by a given factor*
- Weight is adjusted
- It can be applied to specific materials and/or specific particles
- Multiple **`LAM-BIAS`** cards are allowed

Lifetime Biasing

Input card: **LAM-BIAS**

- *Type*
  - <empty>
- ˣ *mean life*
  - Lifetime correction factor (particle rest frame)
- ˣ *λ inelastic*
  - Doesn't apply in this use-case
- *Mat*
  - Material where the correction factor applies
- *Part - to Part - Step*
  - Standard FLUKA particle selection

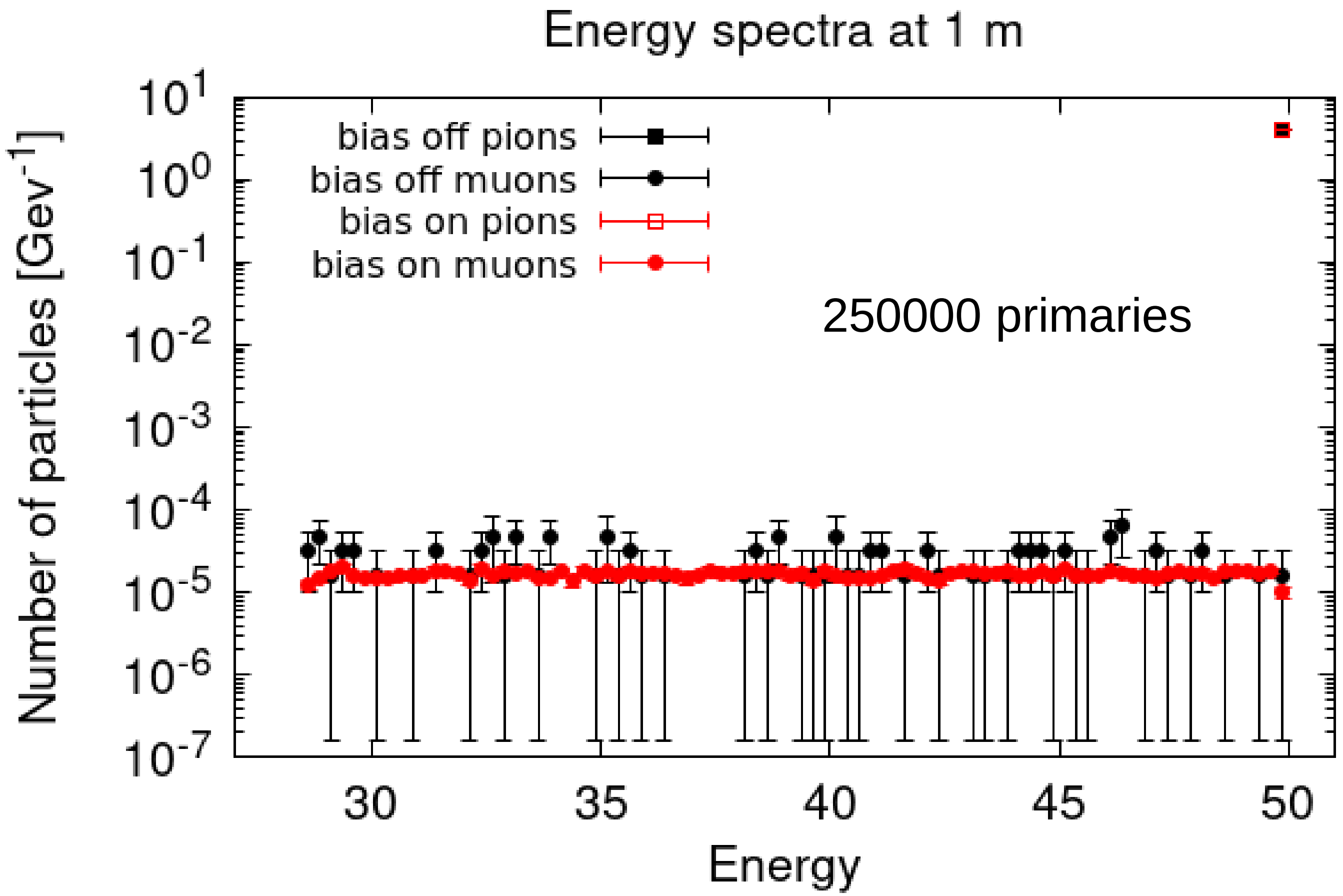
Example explanation:
Lifetime of *positive pions and positive muons*, *in air*, are multiplied by a *correction factor 0.02*

LAM-BIAS	Type: ▾	× mean life: 0.02	× λ inelastic:
Mat: AIR ▾	Part: MUON+ ▾	to Part: PION+ ▾	Step: 3

Lifetime Biasing

Input card: **LAM-BIAS**

- Primaries: 50 GeV pions, π^+
- In vacuum
- Spectra of pions and muons @1m
- Black: no biasing applied
- Red: Lifetime biasing applied

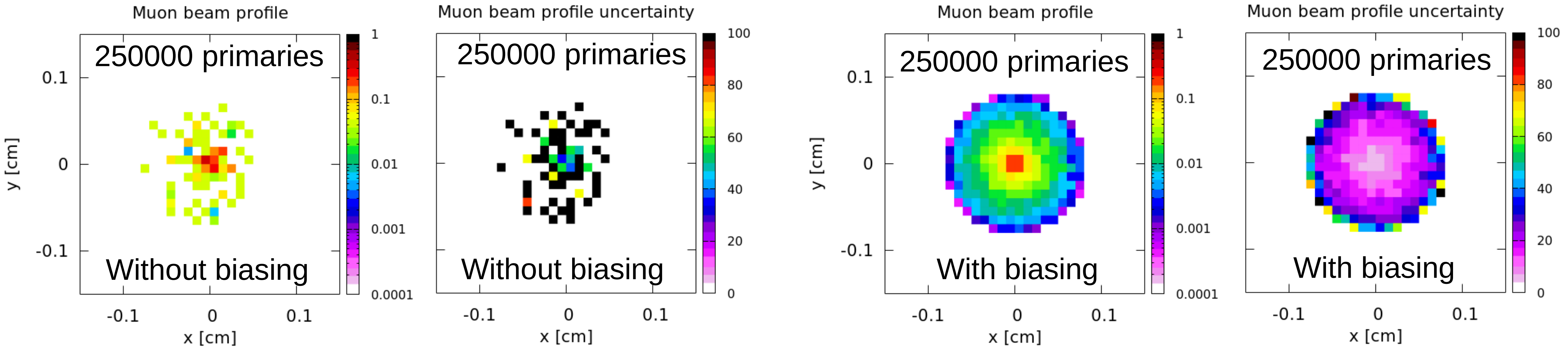


LAM-BIAS Type: ▾ × mean life: **0.01** × λ inelastic:
Mat: **VACUUM** ▾ Part: **PION+** ▾ to Part: ▾ Step:

Lifetime Biasing

Input card: **`LAM-BIAS`**

- Primaries: 50 GeV pions, π^+
- In vacuum
- Transverse distribution of muons @1m



LAM-BIAS Type: ▾ × mean life: **0.01** × λ inelastic:
Mat: **VACUUM** ▾ Part: **PION+** ▾ to Part: ▾ Step:

Decay-length Biasing

Input card: **`LAM-BIAS`**

Decay-length Biasing

- Input card: **`LAM-BIAS`**
- Allows to *set the decay-length of unstable particles*
- Users have to provide the new decay-length in the laboratory in cm
- Weight is adjusted
- It can be applied to specific materials and/or specific particles
- Multiple **`LAM-BIAS`** cards are allowed

Decay-length Biasing

Input card: **LAM-BIAS**

- *Type*
  - GDECAY
- <*λ*>
  - Decay-length in the laboratory frame [cm]
- *x λ inelastic*
  - Doesn't apply in this use-case
- *Mat*
  - Material where the correction factor applies
- *Part - to Part - Step*
  - Standard FLUKA particle selection

Example explanation:
Decay-length of *positive and negative muons*, *in air*, are set to *0.02 cm* in the laboratory frame

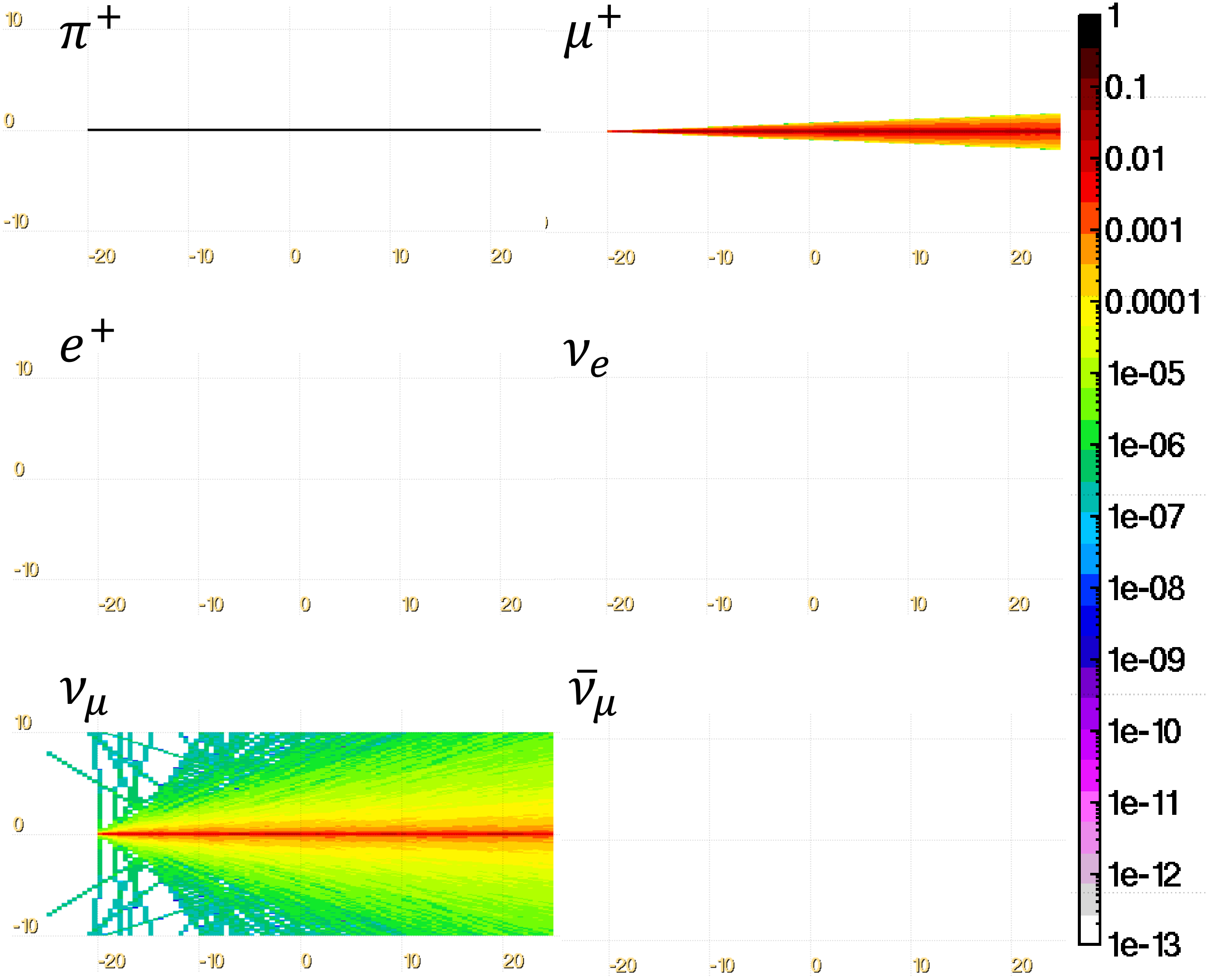
LAM-BIAS Type: **GDECAY ▾** <λ>: **0.02** x λ inelastic:
Mat: **AIR ▾** Part: **MUON+ ▾** to Part: **MUON- ▾** Step:

Decay-length Biasing

Input card: **LAM-BIAS**

- Primaries: 1 GeV pions, π^+
- In vacuum
- Fluences of $\pi^+, \mu^+, e^+, \nu_e, \nu_\mu, \bar{\nu}_\mu$
- BR($\pi^+ \rightarrow \ \mu^+\nu_\mu$) ≈ 0.9999
- BR($\pi^+ \rightarrow \ e^+\nu_e$) ≈ 0.0001
- BR($\mu^+ \rightarrow \ e^+\nu_e\bar{\nu}_\mu$) ≈ 1
- No biasing applied

250000 primaries

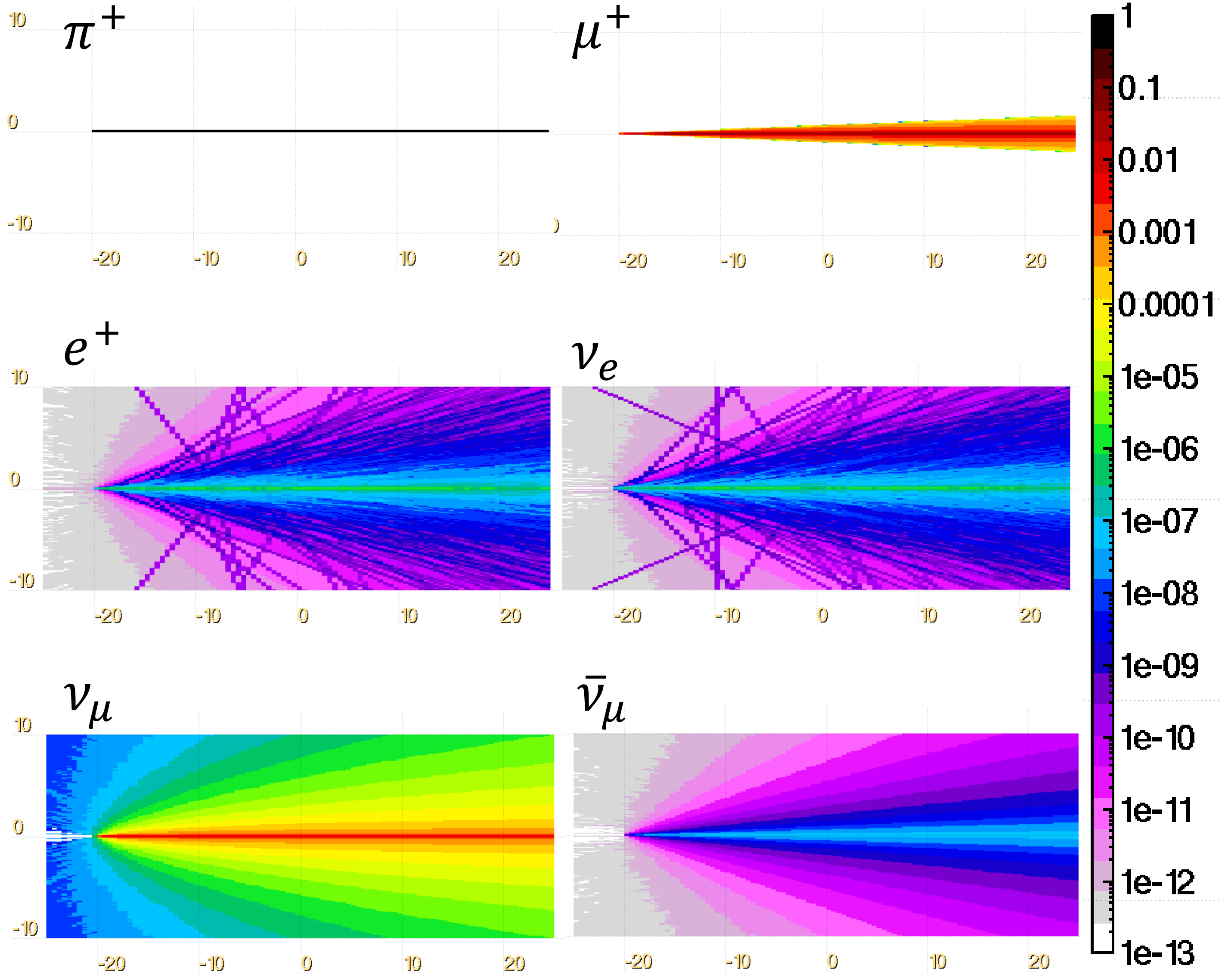


Decay-length Biasing

Input card: **LAM-BIAS**

- Primaries: 1 GeV pions, π^+
- In vacuum
- Fluences of $\pi^+, \mu^+, e^+, \nu_e, \nu_\mu, \bar{\nu}_\mu$
- BR($\pi^+ \rightarrow \mu^+ \nu_\mu$) ≈ 0.9999
- BR($\pi^+ \rightarrow e^+ \nu_e$) ≈ 0.0001
- BR($\mu^+ \rightarrow e^+ \nu_e \bar{\nu}_\mu$) ≈ 1
- Decay-length biasing applied
  - 5 cm for both π^+ and μ^+

250000 primaries



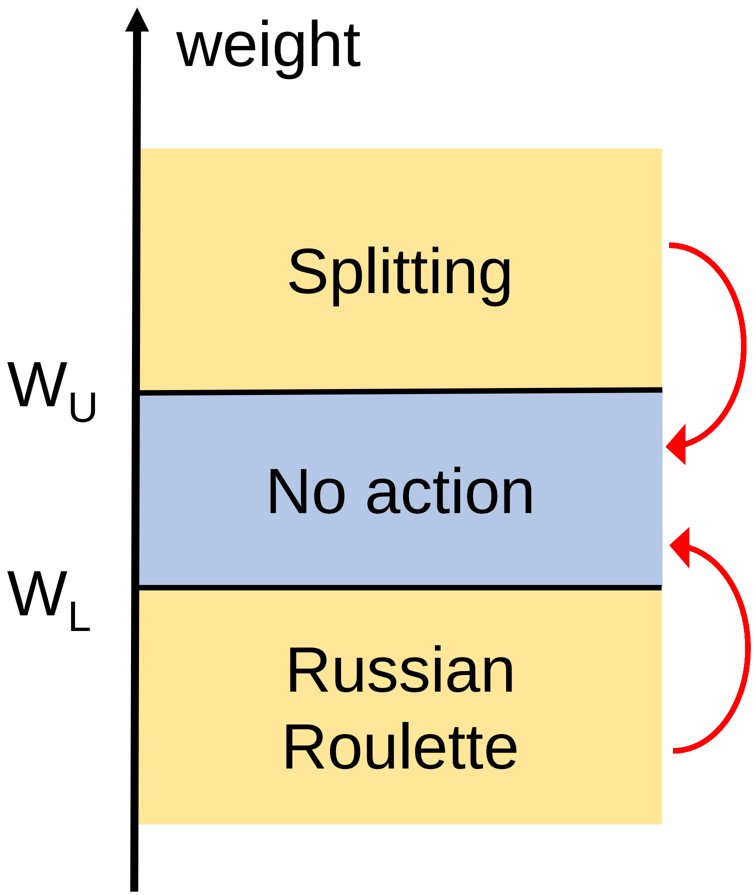
Weight Windows

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

Weight Windows

- Input cards: **`WW-FACTO`**, **`WW-THRES`** **IMPORTANT: both cards needed!**
- Can be used as stand-alone VRT
- Allow to control large weight fluctuations introduced by other biasing techniques
- Combination of RR (Russian Roulette) and Splitting
- Based on the absolute weight value
- Set upper and lower limit on particle weight
  - Killing particle with low weight decreases t and has little effect on σ
  - Splitting a particle with high weight increases t but decreases σ
  - Overall effect: increase of the $FOM=1/(\sigma^2 \cdot t)$
- Typical ratio between upper and lower limit is ~10
- Requires careful design

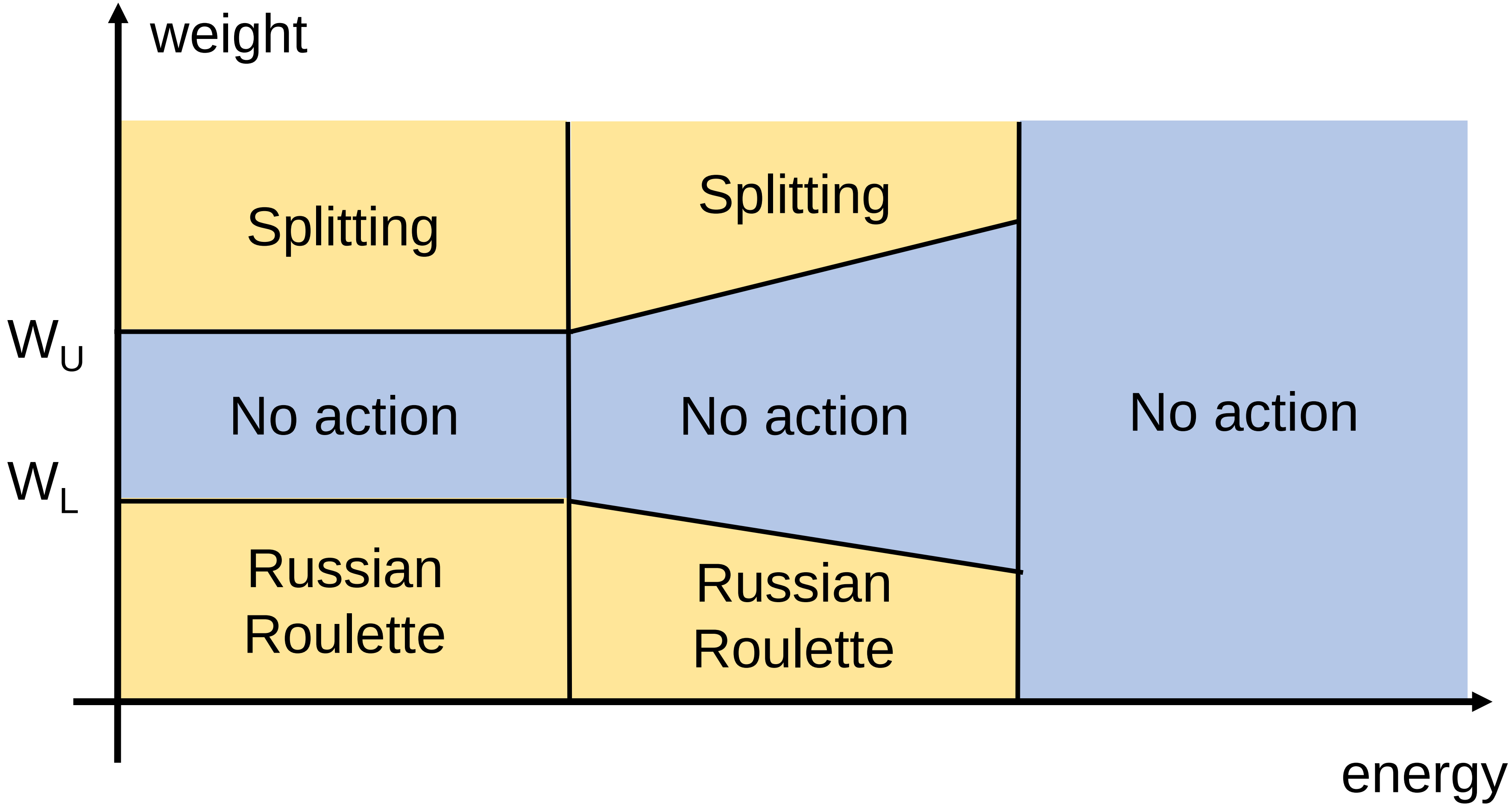
Weight Windows



- Given a particle whose weight is w_i …
- If… $w_i > W_U$
  - The particle is split
- If… $W_U > w_i > W_L$
  - No action is performed
- If… $W_L > w_i$
  - A Russian Roulette is applied
  - If the particle survives, the weight is increased

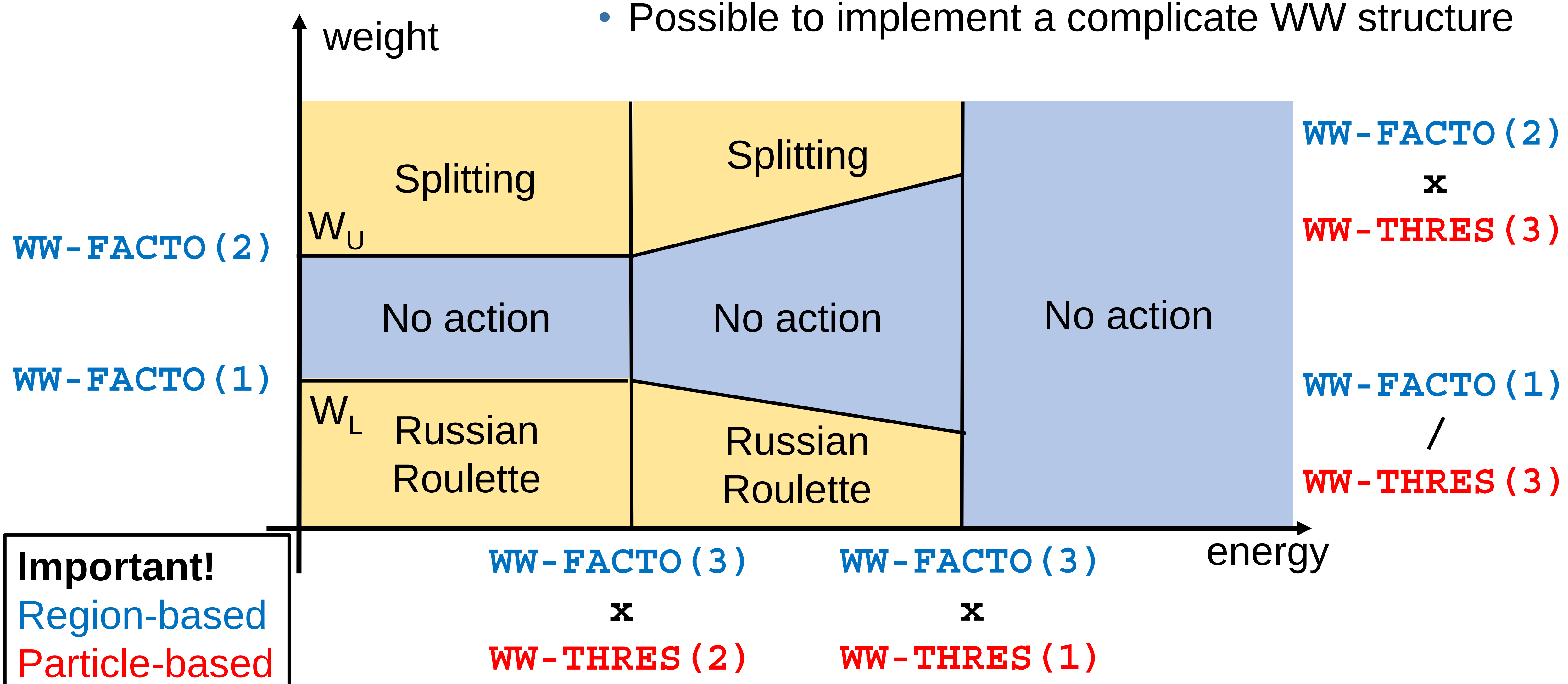
Weight Windows

- Possible to implement a complicate WW structure as function of:
  - Particle energy
  - Particle type
  - Region



Weight Windows

- Possible to implement a complicate WW structure



Important!
Region-based
Particle-based

Weight Windows

Input card: **`WW-THRES`**

- *E upper*
  - Maximum particle energy for which the WW is applied
- *E lower*
  - Minimum particle energy for which the WW is applied
- *amp f*
  - Amplification factor for different WW over different energy ranges
- *Opt*
  - WW applied or not to primaries
- *Part - to Part - Step*
  - Standard FLUKA particle selection

Example explanation:
WW applied to all particles (from *4-HELIUM* to *@LASTPAR*) having energy between *1 KeV* and *2 GeV*; the *amplification factor =1* means that the WW is constant

WW-THRESH E upper: **2** E lower: **1E-6** amp f: **1**
Opt: **PRIMARY ▾** Part: **4-HELIUM ▾** to Part: **@LASTPAR ▾** Step:

Weight Windows

Input card: **`WW-FACTO`**

- *RR*
  - Apply Russian Roulette below this weight
- *Split*
  - Apply splitting above this weight
- *mult f*
  - Mulitplicative factor for the energies defined in **`WW-THRES`**
- *LowE n*
  - Please ignore
- *Reg - to Reg - Step*
  - Standard FLUKA region selection

<u>Example explanation</u>:
WW applied only in region *Shield1*.
RR is applied to all particles having weight lower than *1E-2*.
Splitting is applied to all particles having weight larger than *1/1.5849*.

WW-FACTOR		RR:	1E-2	Split:	=1/1.5849	mult f:	1
LowE n:	▾	Reg:	Shield1 ▾	to Reg:	▾	Step:	

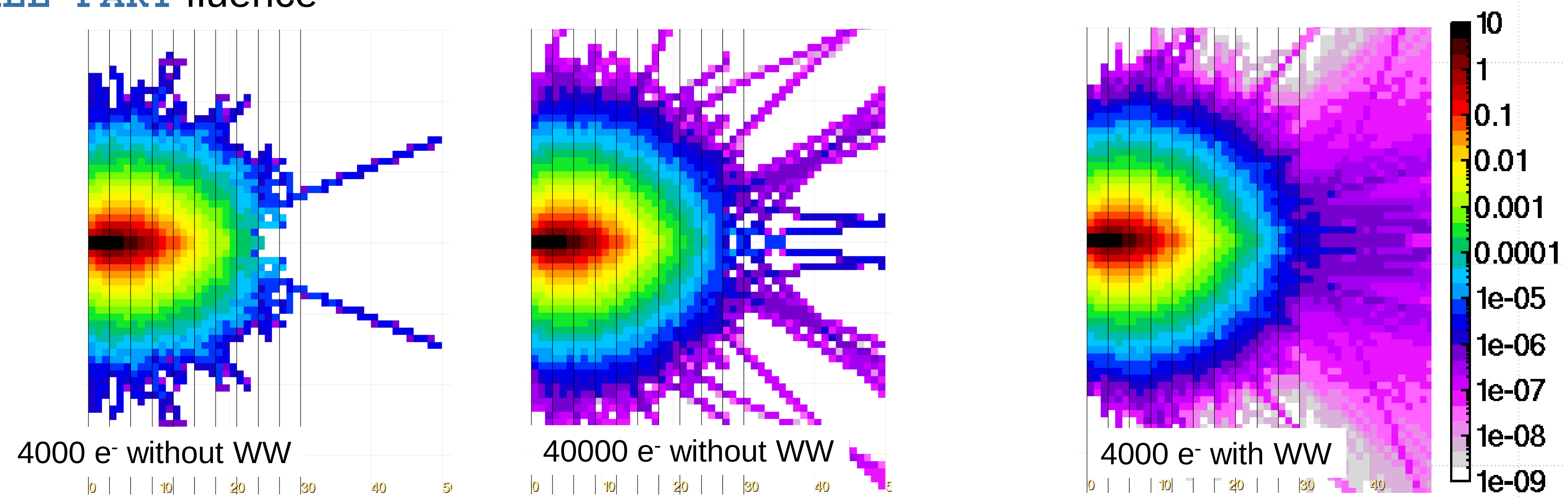
Weight Windows

Input card: **`WW-THRES`**, **`WW-FACTO`**

Primaries: 500 MeV electrons

From vacuum to lead target 10 slab, 3 cm each

`ALL-PART` fluence



4000 e^- without WW

40000 e^- without WW

4000 e^- with WW

USIMBS: USer defined IMportance BiaSing

Input card: `BIASING`

USIMBS

- Input card: **`BIASING`**
- Allows to…
  - *…code very complicated user-defined biasing*
- Improper implementation can lead to divergent simulation time
- Requires extremely careful tuning
- Multiple **`BIASING`** cards are allowed
- It also works with pointwise neutron cross sections

USIMBS

Input card: **`BIASING`**

- *Opt*
  - USER (to select biasing activated via the **`usimbs.f`** user routine)
- *Type*
  - Select to which type of particles the used defined biasing has to apply
  - "All particles", "Hadrons & Muons", "e-e+, γ", "Low neutrons"
- *Imp*
  - To activate/deactivate USIMBS calls in the selected regions
  - Imp = 1 deactivates USIMBS calls ; *Imp ≠ 1 activates USIMBS calls*
- *Reg - to Reg - Step*
  - Standard FLUKA region selection
- *RR*
  - To apply a Russian Roulette to the secondary generation (only for "Hadrons & Muons")

USIMBS

- Example 1

  BIASING | Type: All particles ▾ | RR: | Imp: 2.0
  Opt: USER ▾ | Reg: TARGET ▾ | to Reg: ▾ | Step:

  - *usimbs* biasing applied to *all particles* in region *TARGET*

- Example 2

  BIASING | Type: Hadrons & Muons ▾ | RR: 0.5 | Imp: 2.0
  Opt: USER ▾ | Reg: TARGET ▾ | to Reg: ▾ | Step:

  - *usimbs* biasing applied to *"Hadrons & Muons"* in region *TARGET*, *50% Russian Roulette* applied to *secondaries from "Hadrons & Muons"*

- Example 3

  BIASING | Type: All particles ▾ | RR: 0.5 | Imp: 2.0
  Opt: USER ▾ | Reg: TARGET ▾ | to Reg: ▾ | Step:

  - *usimbs* biasing applied to *all particles* in region *TARGET*, *50% Russian Roulette* applied to *secondaries from "Hadrons & Muons"*

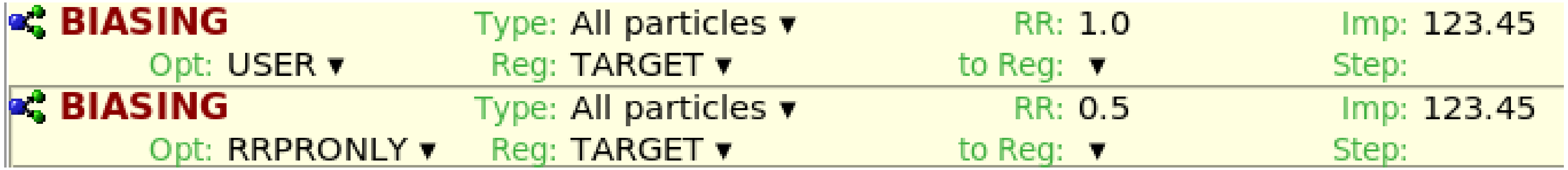
- Example 4

  BIASING | Type: e-e+,γ ▾ | RR: 0.5 | Imp: 2.0
  Opt: USER ▾ | Reg: TARGET ▾ | to Reg: ▾ | Step:

  - *usimbs* biasing applied to "*e-e+, γ*" in region *TARGET*, *NO Russian Roulette* is applied

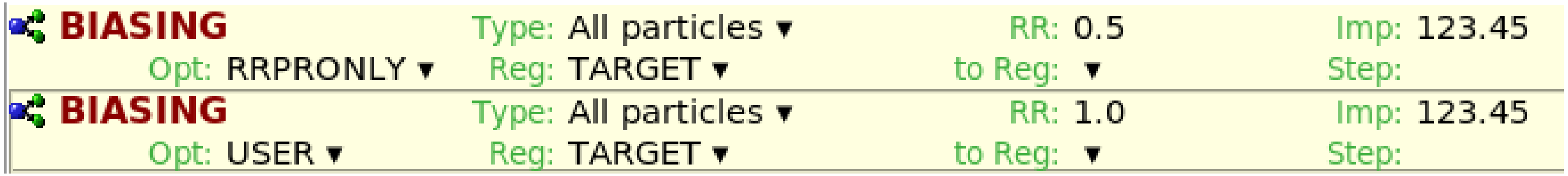
USIMBS

- Example 5



  - *usimbs* biasing applied to *all particles* in region *TARGET*
  - *50% Russian Roulette* applied to *secondaries from "Hadrons & Muons" primary interactions*
  - **`RRPRONLY`**: Russian Roulette to PRimary interactions ONLY

- Example 6



  - *usimbs* biasing applied to *all particles* in region *TARGET*
  - *NO Russian Roulette* applied to any secondary
  - The second card supersede the first card

USIMBS

- Example 7

BIASING	Type: All particles ▾	RR:	Imp: 123.45
Opt: USER ▾	Reg: REG_1 ▾	to Reg: ▾	Step:
BIASING	Type: All particles ▾	RR:	Imp: 543.12
Opt: ▾	Reg: REG_2 ▾	to Reg: ▾	Step:
BIASING	Type: All particles ▾	RR:	Imp: 23.451
Opt: ▾	Reg: REG_3 ▾	to Reg: ▾	Step:

  - Misleading use of biasing
  - One card with *Opt USER* activated *usimbs* biasing for all regions having *Imp ≠ 1*
  - *usimbs* biasing is applied to *all particles* in *REG_1*, *REG_2*, & *REG_3*

- Example 8

BIASING	Type: All particles ▾	RR:	Imp: 123.45
Opt: USER ▾	Reg: REG_1 ▾	to Reg: @LASTREG ▾	Step:
BIASING	Type: All particles ▾	RR:	Imp: 1.0
Opt: ▾	Reg: REG_2 ▾	to Reg: ▾	Step:

  - Possible clever of biasing: avoid unnecessary calls to the user routine
  - First, *usimbs* biasing is applied to *all particles* in *all regions*
  - Then, *usimbs* biasing is deactivated in *REG_2*

USIMBS – `usimbs.f`

- **`usimbs.f`** contains a subroutine and a entry which are called at different times during transport

- **`SUBROUTINE USIMBS`**

- **`ENTRY USIMST`**

```
*=== Usimbs ===========================================================
      SUBROUTINE USIMBS ( MREG, NEWREG, FIMP )
      INCLUDE 'dblprc.inc'
      INCLUDE 'dimpar.inc'
      INCLUDE 'iounit.inc'
*----------------------------------------------------------------------
*     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.inc'
      FIMP   = ONEONE
      RETURN
*
*======================================================================
*     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 – `usimbs.f`

- **`ENTRY USIMST`**
- Called when the step is calculated
- It halves any step > 1 cm
- Then, **`SUBROUTINE USIMBS`** is called
- Relevant for neutral particles:

  the end of the step is

  either an interaction point

  or a boundary crossing
- The step length **`STEP`** is not available

  in the subroutine

```
*=== Usimbs ===========================================================
      SUBROUTINE USIMBS ( MREG, NEWREG, FIMP )
      INCLUDE 'dblprc.inc'
      INCLUDE 'dimpar.inc'
      INCLUDE 'iounit.inc'
*----------------------------------------------------------------------
*     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.inc'
      FIMP   = ONEONE
      RETURN
*
*======================================================================
*     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 – `usimbs.f`

- **SUBROUTINE USIMBS**
- Called when the step is performed
- Variable **FIMP** sets the ratio to apply

  Russian roulette and splitting
- Any combination of criteria can be

  used to calculate **FIMP**

```
*=== Usimbs ===========================================================
      SUBROUTINE USIMBS ( MREG, NEWREG, FIMP )
      INCLUDE 'dblprc.inc'
      INCLUDE 'dimpar.inc'
      INCLUDE 'iounit.inc'
*----------------------------------------------------------------------
*     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.inc'
      FIMP   = ONEONE
      RETURN
*
*======================================================================
*     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 – `usimbs.f` implementation example

```
FIMP = ONEONE
if ( JTRACK .eq. 8 ) then
   if ( XTRACK(0).gt.1234.d0 .and. XTRACK(0).lt.4321.d0 ) then
      if ( YTRACK(0).gt.0.d0 .and. YTRACK(0).lt.21.d0 ) then
         FIMP = ...
      endif
   endif
endif
```

Implementation based on
particle type (**JTRACK=8** means neutrons)
& particle position

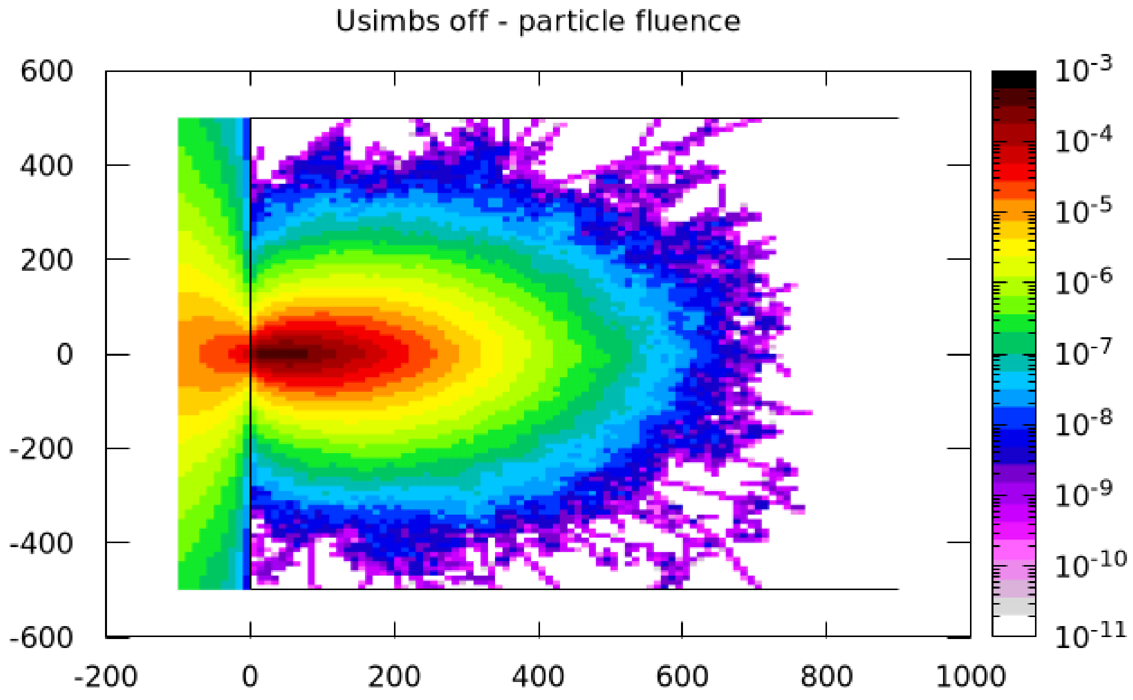
USIMBS general comments

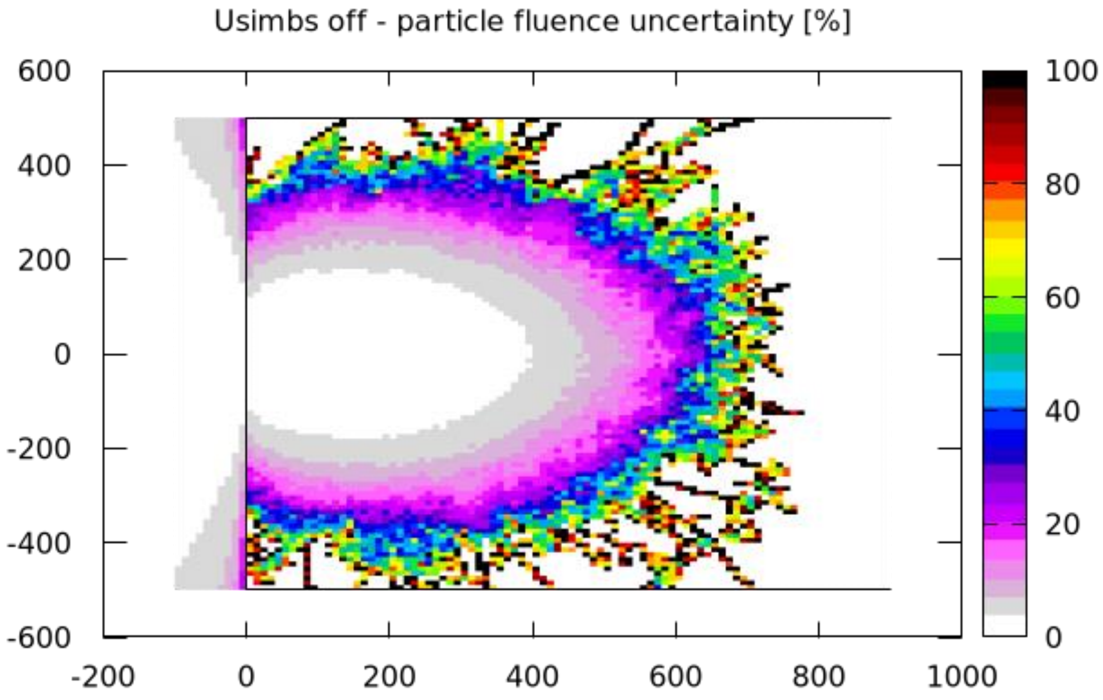
- `FIMP` increase should not be excessive
- The increase is called at each step
- The stack can be filled quickly
- Excessive biasing can lead to a divergent CPU time
- `FIMP = 1.2` is already quite large
- Good strategy: compensate for the particle attenuation
- Important feature: directional biasing can be implemented
- Neutrons and photons biasing could be sufficient:
  - Charged particles step length can be “complicated”: multiple scattering, energy loss, δ-ray
  - Neutral particles are “drivers” for charged particles

USIMBS

- Example of directional biasing
- Target: concrete cylinder, 5m radius, 10 m depth
- Beam: 1 GeV neutrons
- Without biasing

250000 primaries

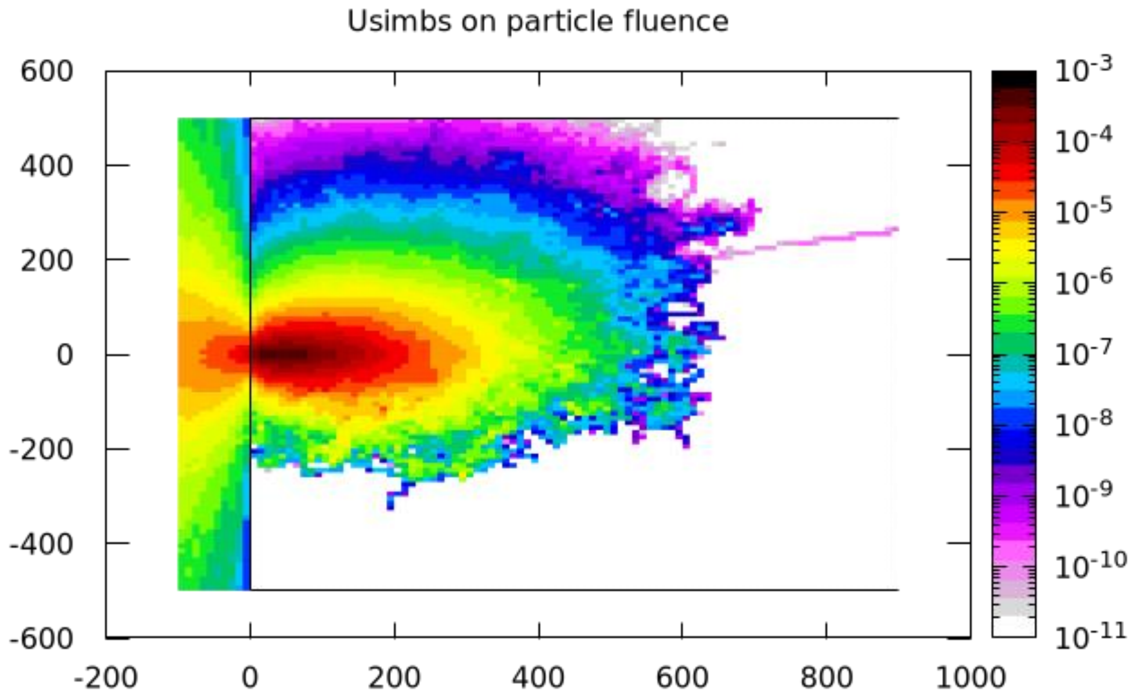


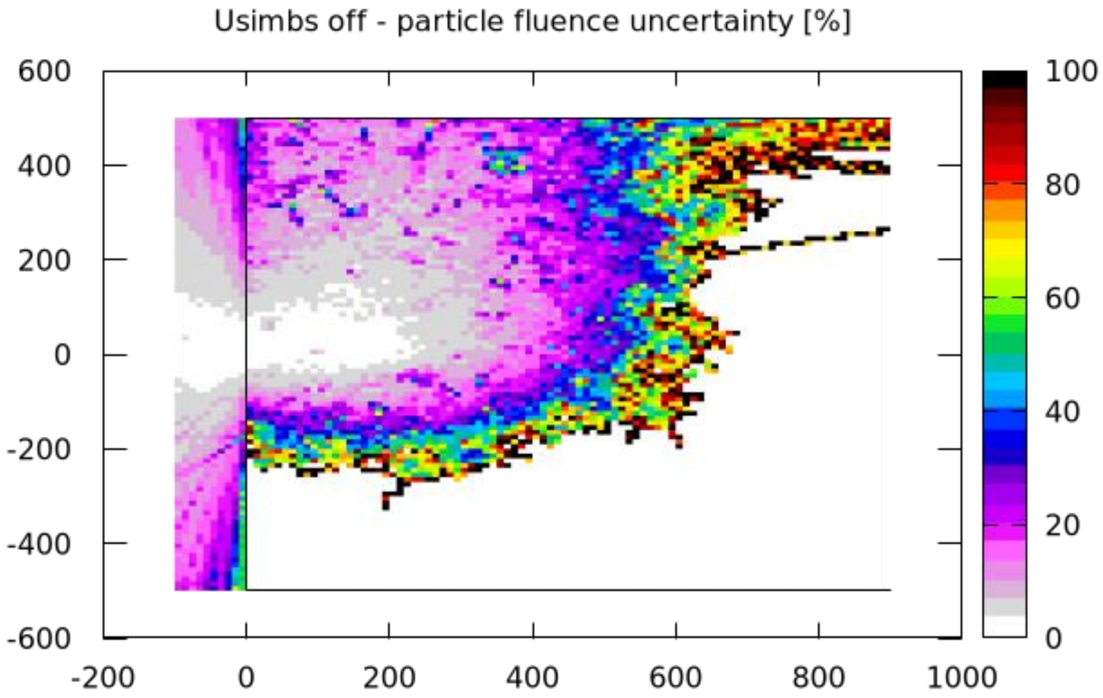


USIMBS

- Example of directional biasing
- Target: concrete cylinder, 5m radius, 10 m depth
- Beam: 1 GeV neutrons
- With directional biasing toward (1000.d0, 0.d0, 200.d0)

25000 primaries





Summary of the input cards for biasing

Summary of the input cards for biasing

- `BIASING`
  - Region Importance biasing (Surface Splitting and Russian Roulette)
  - Secondaries multiplicity tuning in hadronic interactions
  - User defined importance biasing, USIBMS
- `EMF-BIAS`
  - Leading Particle Biasing for electron, positron, and photon interactions
- `LAM-BIAS`
  - Mean free path biasing (interaction length)
  - Lifetime & Decay-length biasing
- `WW-FACTO` , `WW-THRES`
  - Weight windows biasing

FLUKA