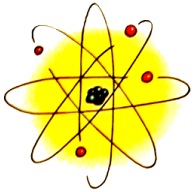




Joint ICTP-IAEA Workshop on Monte Carlo Radiation Transport and Associated Data Needs for Medical Applications
28 October – 8 November 2024
ICTP, Trieste, Italy

Lecture 26

egs++ applications: egs_chamber

Frédéric Tessier

NRC Metrology
National Research Council Canada

Government of Canada Gouvernement du Canada

Carleton University

Université de Montréal

nova scotia health

All egs++ applications require an `.egsinp` file

All the egs++ applications distributed with EGSnrc rely on input blocks saved in a text file ending in `.egsinp`

The input file must reside inside the same directory as the application source code, e.g., inside the `$EGS_HOME/egs_chamber` directory for the `egs_chamber` application.

Standard egs++ applications require:

1. Geometry definition
2. Media definition
3. Source definition
4. Monte Carlo transport parameters
5. Run control input
6. Random number generator seeds
7. **Application-specific input**

1. Geometry definition input

Usually, several geometries are defined and are then combined with composite geometry objects to build the final, more complex geometry. The typical layout is:

```
:start geometry definition:

    :start geometry:
       name = foo
       (...)
    :stop geometry:
```

1. Geometry definition input

Usually, several geometries are defined and are then combined with composite geometry objects to build the final, more complex geometry. The typical layout is:

```
:start geometry definition:

    :start geometry:
       name = foo
       (...)
    :stop geometry:

    :start geometry:
       name = bar
       (...)
    :stop geometry:
```

1. Geometry definition input

Usually, several geometries are defined and are then combined with composite geometry objects to build the final, more complex geometry. The typical layout is:

```
:start geometry definition:

    :start geometry:
       name = foo
       (...)
    :stop geometry:

    :start geometry:
       name = bar
       (...)
    :stop geometry:

    simulation geometry = foo   # or bar

:stop geometry definition:
```

1. Geometry definition input

Usually, several geometries are defined and are then combined with composite geometry objects to build the final, more complex geometry. The typical layout is:

```
:start geometry definition:

    :start geometry:
       name = foo
    :stop geometry:

    :start geometry:
       name = bar
    :stop geometry:

    simulation geometry = foo   # or bar

:stop geometry definition:
```

The `simulation geometry` key specifies the geometry to load in `egs_view`, but the scoring input might override this for the actual calculation geometry.

2. Media definition input

All media names in the input file, as well as the energy limits, must be defined in the media definition input block. The layout is, for example:

```
:start media definition:

    ae = 0.521
    ap = 0.010
    ue = 50.511
    up = 50
```

2. Media definition input

All media names in the input file, as well as the energy limits, must be defined in the media definition input block. The layout is, for example:

```
:start media definition:

    ae = 0.521
    ap = 0.010
    ue = 50.511
    up = 50

    :start water:
       name = water_liquid
    :stop water:
```

2. Media definition input

All media names in the input file, as well as the energy limits, must be defined in the media definition input block. The layout is, for example:

```
:start media definition:

    ae = 0.521
    ap = 0.010
    ue = 50.511
    up = 50

    :start water:
       name = water_liquid
    :stop water:

    :start my_water:
        elements        = H, O
        number of atoms = 2, 1
        rho             = 0.998
    :stop my_water:

:stop media definition:
```

2. Media definition input

All media names in the input file, as well as the energy limits, must be defined in the media definition input block. The layout is, for example:

```
:start media definition:

    ae = 0.521
    ap = 0.010
    ue = 50.511
    up = 50

    :start water:
       name = water_liquid
    :stop water:

    :start my_water:
        elements        = H, O
        mass fractions  = 0.111894, 0.888106
        rho             = 0.998
    :stop my_water:

:stop media definition:
```

3. Source definition input

Applications get particles from the `getNextParticle()` method of the source object.
If particles are created outside of the geometry, they take a single long step along their initial direction until they hit the geometry.

```
:start source definition:

    :start source:
        name = foo
        (...)
    :stop source:
```

3. Source definition input

Applications get particles from the `getNextParticle()` method of the source object.
If particles are created outside of the geometry, they take a single long step along their initial direction until they hit the geometry.

```
:start source definition:

    :start source:
        name = foo
        (...)
    :stop source:

    :start source:
        name = bar
        (...)
    :stop source:
```

3. Source definition input

Applications get particles from the `getNextParticle()` method of the source object.
If particles are created outside of the geometry, they take a single long step along their initial direction until they hit the geometry.

```
:start source definition:

    :start source:
        name = foo
        (...)
    :stop source:

    :start source:
        name = bar
        (...)
    :stop source:

    simulation source = foo   # or bar

:stop source definition:
```

3. Source definition input

Applications get particles from the `getNextParticle()` method of the source object.
If particles are created outside of the geometry, they take a single long step along their initial direction until they hit the geometry.

```
:start source definition:

    :start source:
        name = foo
    :stop source:

    :start source:
        name = bar
    :stop source:

    simulation source = foo   # or bar

:stop source definition:
```

Particles might miss the geometry! Make sure that your source and geometry are defined so that particles are inside the geometry or aimed towards it.

4. Monte Carlo transport parameters

Monte Carlo transport parameter inputs are common to all EGSnrc applications. Default values are set to provide accurate simulation of coupled electron-photon transport. For example:

```
:start MC transport parameter:
    Global ECUT                     = 0.521   # electron cutoff (MeV)
    Global PCUT                     = 0.010   # photon cutoff (MeV)
    Spin effects                    = On      # [On], Off
    Brems cross sections            = NRC     # [BH], NIST, NRC
    Bound Compton scattering        = On      # [On], Off, norej
    Rayleigh scattering             = On      # [On], Off, custom
    Atomic relaxations              = On      # [On], Off
    Brems angular sampling          = KM      # Simple, [KM]
    Pair angular sampling           = KM      # Off, [Simple], KM
    Photoelectron angular sampling  = On      # [On], Off
    Electron Impact Ionization      = Off     # On, [Off], ...
    Photon cross sections           = xcom    # [xcom], epdl, si
:stop MC transport parameter:
```

5. Run control input

Simulations are split into **chunks** (just one chunk in serial execution) and chunks are further divided in **batches** to help in displaying progress and saving intermediate results.

The simulation is controlled by a **run control object** (RCO), which:

- reads the number of histories requested
- reports the progress of the simulation after each batch
- defines the type of simulation (first, restart, combine or analyze)
- terminates the simulation if the sought accuracy is attained
- terminates the simulation if the maximum alloted CPU time is reached.

```
:start run control:
    ncase                  = 1e9   # number of histories to run
    geometry error limit = 1e2   # allow a few geometry errors
    statistical accuracy sought  = 1   # in percents (%)
    calculation            = first # [first], restart, combine, analyze
    nbatch                 = 10    # number of batches [10]
    nchunk                 = 10    # number of chunks  [10]
:stop run control:
```

6. Random number generator seeds

Statistically independent simulation runs require independent random number generator seeds. In egs++ applications the seeds are set via a **rng definition** input block:

```
:start rng definition:
    initial seeds = 91 2556    # any two integers less than 30000
:stop rng definition:
```

In **parallel runs,** the application object takes care of incrementing the seed so that each job in the parallel run is statistically independent.

EGSnrc bundles a few egs++ applications

The EGSnrc distribution contains some ready-made egs++ applications geared towards specific radiation transport scenarios. These applications are derived from either `EGS_SimpleApplication` or `EGS_AdvancedApplication` and are normally installed in corresponding directories under `$EGS_HOME/`.

- `tutor2pp`, `tutor7pp`: tutorial egs++ applications
- `cavity`: ion chamber dose calculations
- **egs_chamber**: efficient in-phantom ion chamber calculations
- `egs_fac`: free-air chamber correction factors calculations
- `egs_cbct`: cone-beam CT scatter correction calculations
- (`egs_brachy`: brachytherapy calculations; not in distribution yet)
- (for this course: `myapp`, `tutor_4pp`, `tutor_6pp`)

You steer `egs_chamber` with scoring options

The `scoring options` input block in the `.egsinp` file determines the calculation scenario. It must contain at least one `calculation geometry` input block.

Internally, EGSnrc tracks energy deposition. To calculate **dose**, the deposited energy is divided by the **cavity mass**, which you **must supply manually**.

```
### simplest scoring options input block

:start scoring options:

    :start calculation geometry:
        geometry name  = my_world  # OVERRIDES the "simulation geometry"
        cavity regions = 2 5 76    # list of cavity region indices
        cavity mass    = 42.31     # in grams (just a scaling factor)
    :stop calculation geometry:

:stop scoring options:
```

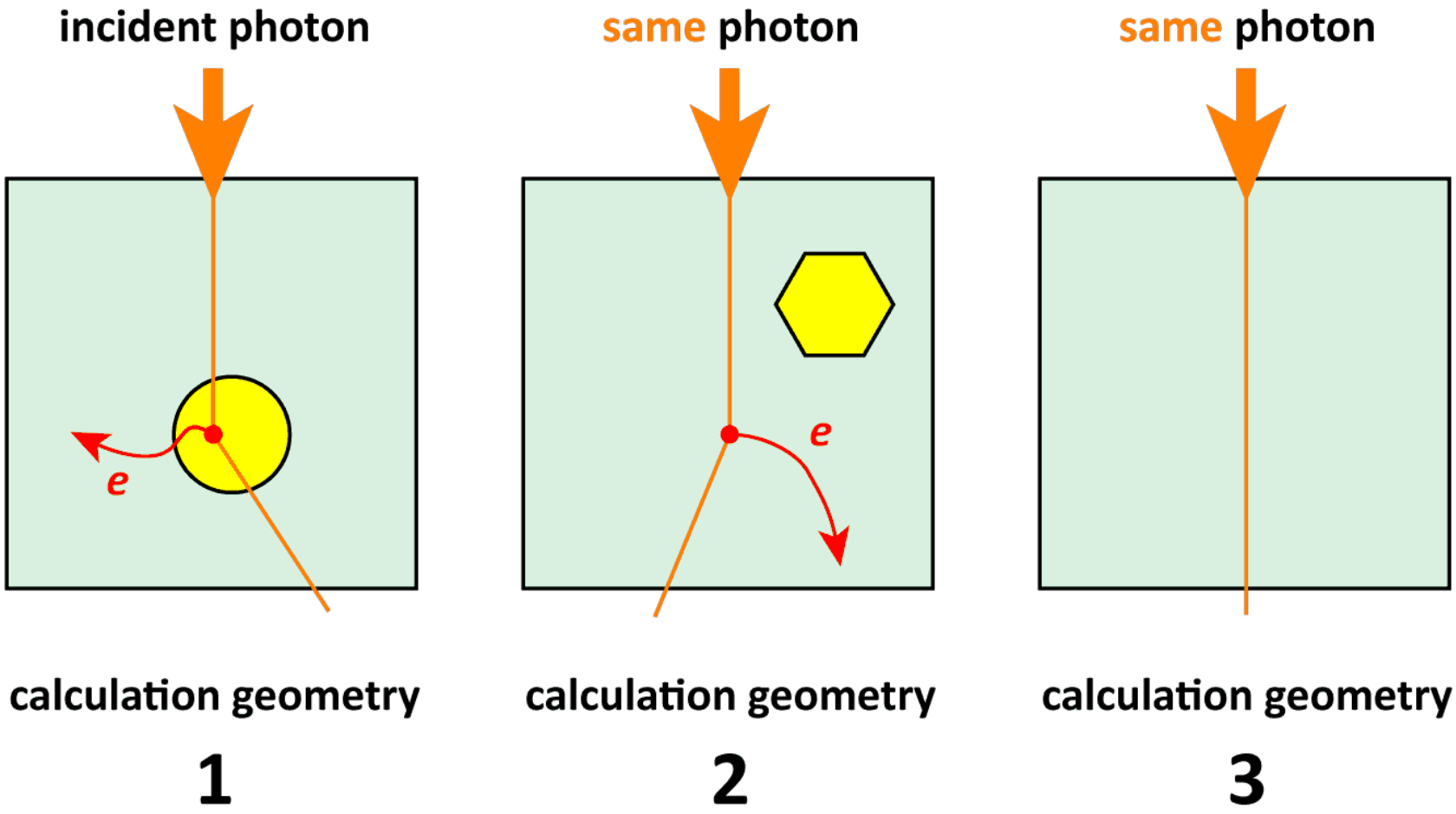
You can define multiple calculation geometries

If there is more than one calculation geometry, then egs_chamber transports every incident particle through each calculation geometry in turn, independently.

```
:start scoring options:

    :start calculation geometry:
        geometry name  = my_world
        cavity regions = 2 5 76
        cavity mass    = 42.31
    :stop calculation geometry:

    :start calculation geometry:
        geometry name  = another_world
        cavity regions = 0
        cavity mass    = 8.43
    :stop calculation geometry:

:stop scoring options:
```

You can define multiple calculation geometries

If there is more than one calculation geometry, then `egs_chamber` transports every incident particle through each calculation geometry in turn, independently.



egs_chamber **outputs the dose to the cavity**

The dose is given in Gy (J/kg), per incident particle, along with its relative statistical uncertainty, near the bottom of the output:

If there are more than one calculation geometry, then one dose value is output **for each calculation geometry:**

```
Finished simulation

Total cpu time for this run:            63.31 (sec.) 0.0176(hours)
Histories per hour:                     5.68631e+07
Number of random numbers used:          1639787994
Number of electron CH steps:            8.67789e+07
Number of all electron steps:           9.5435e+07


 last case = 1000000 fluence = 1e+06

Geometry                              Cavity dose
-----------------------------------------------
my_world                    1.0719e-15 +/- 0.511  %
another_world               1.2435e-15 +/- 0.312  %
```

Apply transformations to source particles!

You can apply a rigid body transformation to **incident source particles** before they are transported in a calculation geometry (i.e., for a rotation matrix $\mathbf{R}$ and translation $\boldsymbol{t}$, the original source particle position $\boldsymbol{x}$ becomes $\boldsymbol{x}' = \mathbf{R}\boldsymbol{x} + \boldsymbol{t}$).

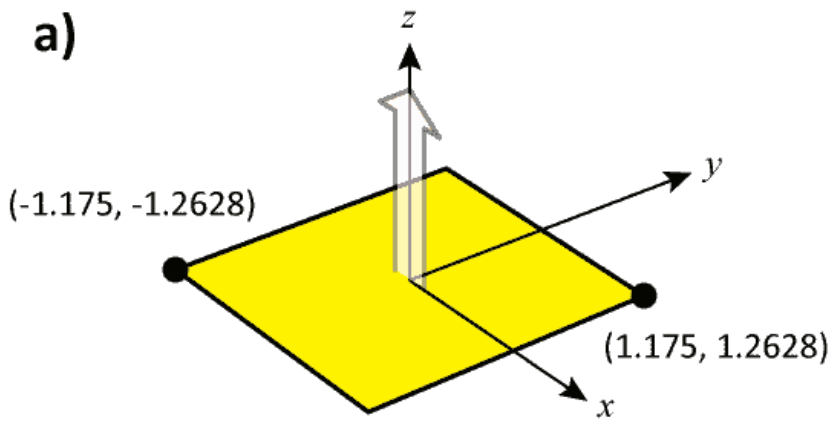
This is much more efficient than using a transformed geometry because the transformation is applied only once at the outset to the source particle coordinates, rather than at *every single step* through the geometry.

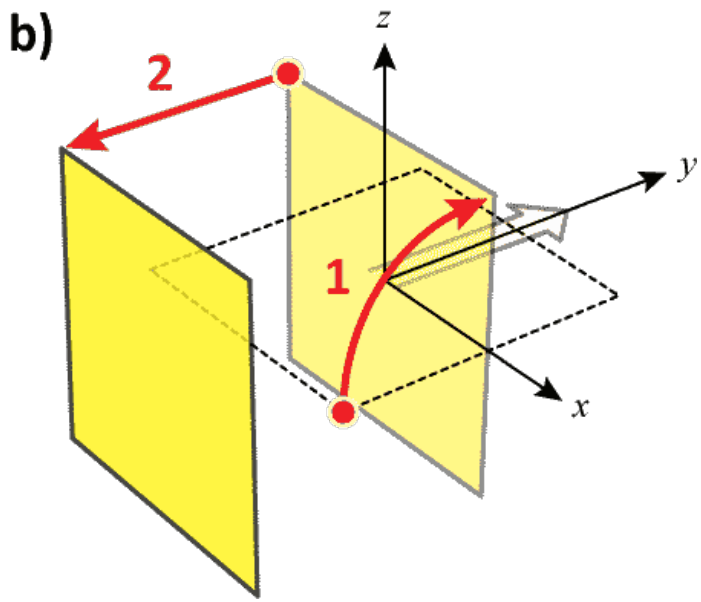
```
:start calculation geometry:

     geometry name  = my_world
     cavity regions = 2 5 76
     cavity mass    = 42.31

    :start transformation:      # transform source: rotate, then translate
       rotation    = 1.57 0 0   # radians, around x, y, z axes
       translation = 0   -2 0   # translation along x, y, z
    :stop transformation:

:stop calculation geometry:
```

Apply transformations to source particles!

You can apply a rigid body transformation to **incident source particles** before they are transported in a calculation geometry (i.e., for a rotation matrix $\mathbf{R}$ and translation $\boldsymbol{t}$, the original source particle position $\boldsymbol{x}$ becomes $\boldsymbol{x}' = \mathbf{R}\boldsymbol{x} + \boldsymbol{t}$).

This is much more efficient than using a transformed geometry because the transformation is applied only once at the outset to the source particle coordinates, rather than at *every single step* through the geometry.





The egs_chamber application improves efficiency

The egs_chamber application features true variance reduction techniques for efficient in-phantom ion chamber simulations and calculation of perturbation factors.

These will probably be implemented as scoring objects in the future.

1. **ECUT** on a region-by-region basis
2. **Radiative splitting**
3. **Photon splitting**
4. **Russian roulette** based on range
5. **XCSE:** photon cross-section enhancement
6. **IPSS:** intermediate phase-space storage
7. **Correlated sampling**

J Wulff, K Zink, I Kawrakow. **Med. Phys. 35,** 4, 1328–1336 (2008).

The egs_chamber application improves efficiency

The egs_chamber application features true variance reduction techniques for efficient in-phantom ion chamber simulations and calculation of perturbation factors.

These will probably be implemented as scoring objects in the future.

1. **ECUT** on a region-by-region basis
2. **Radiative splitting**
3. **Photon splitting**
4. **Russian roulette** based on range
5. **XCSE:** photon cross-section enhancement
6. **IPSS:** intermediate phase-space storage
7. **Correlated sampling**

J Wulff, K Zink, I Kawrakow. **Med. Phys. 35,** 4, 1328–1336 (2008).

1. ECUT on a region-by-region basis

It is possible to attribute a larger value of ECUT to a group of regions in the geometry: electrons with total energy $E < E_{\rm cut}$ deposit their energy locally and are discarded.

This is an **approximate variation reduction technique** but it can be useful, e.g., to investigate the contribution of electrons from different regions of the geometry.

```
:start calculation geometry:

    geometry name  = my_world
    cavity regions = 2 5 76
    cavity mass    = 42.31

    ECUT regions   = 7 14 31 56    # list of regions numbers
    ECUT           = 2  1  1 0.7   # value of ECUT (MeV)

:stop calculation geometry:
```

2. Radiative splitting (bremsstrahlung splitting)

Radiative splitting has now been implemented as a scoring object.

Essentially the same as Uniform Bremsstrahlung Splitting (UBS) in BEAMnrc.

Implemented by setting `the_egsvr->nbr_split` to the supplied splitting number n_{split}.

Each radiative event generates n_{split} photons of statistical weight $1/n_{\text{split}}$.

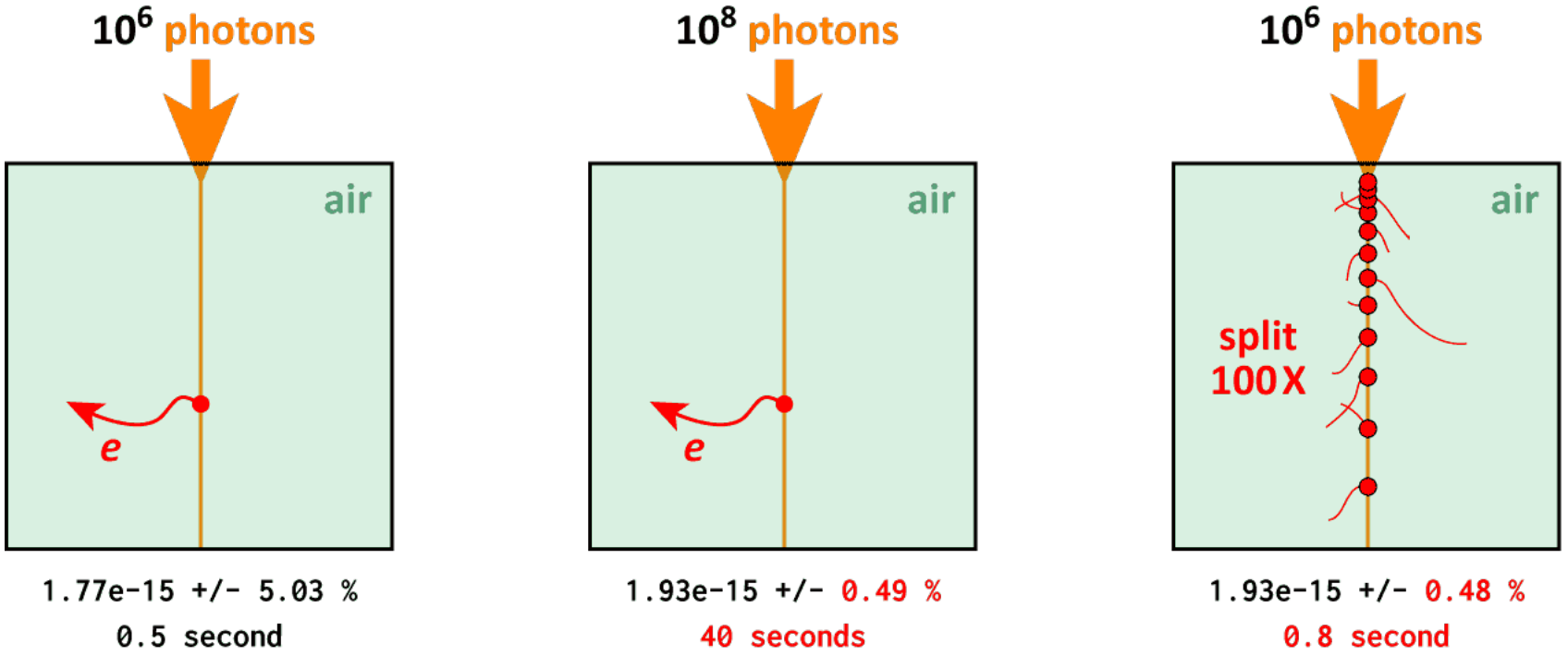
Defined in the top-level `variance reduction` input block.

```
:start variance reduction:
    radiative splitting = 100   # splitting number n_split
:stop variance reduction:
```

3. Photon splitting

All photons are split into n_{split} photons bearing a statistical weight of $1/n_{\mathrm{split}}$, and their interaction sites are distributed uniformly, *in probability space*.

```
:start variance reduction:
    photon splitting = 100   # splitting number n_split
:stop variance reduction:
```



The egs_chamber application improves efficiency

The egs_chamber application features true variance reduction techniques for efficient in-phantom ion chamber simulations and calculation of perturbation factors.

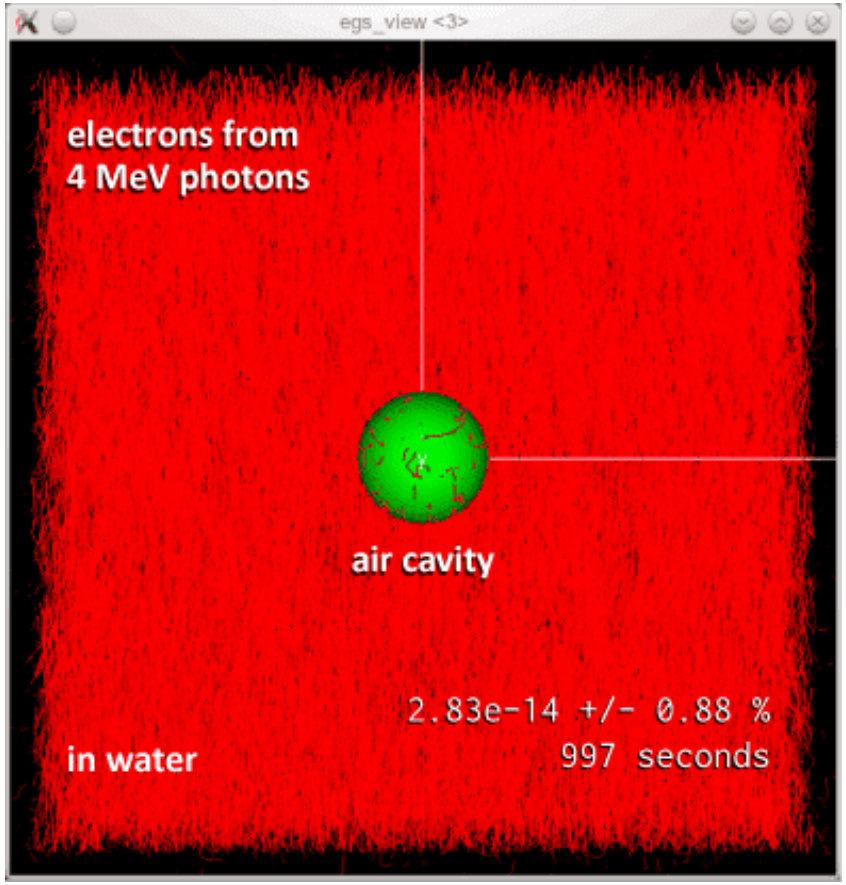
These will probably be implemented as scoring objects in the future.

1. **ECUT** on a region-by-region basis
2. **Radiative splitting**
3. **Photon splitting**
4. **Russian roulette** based on range
5. **XCSE:** photon cross-section enhancement
6. **IPSS:** intermediate phase-space storage
7. **Correlated sampling**

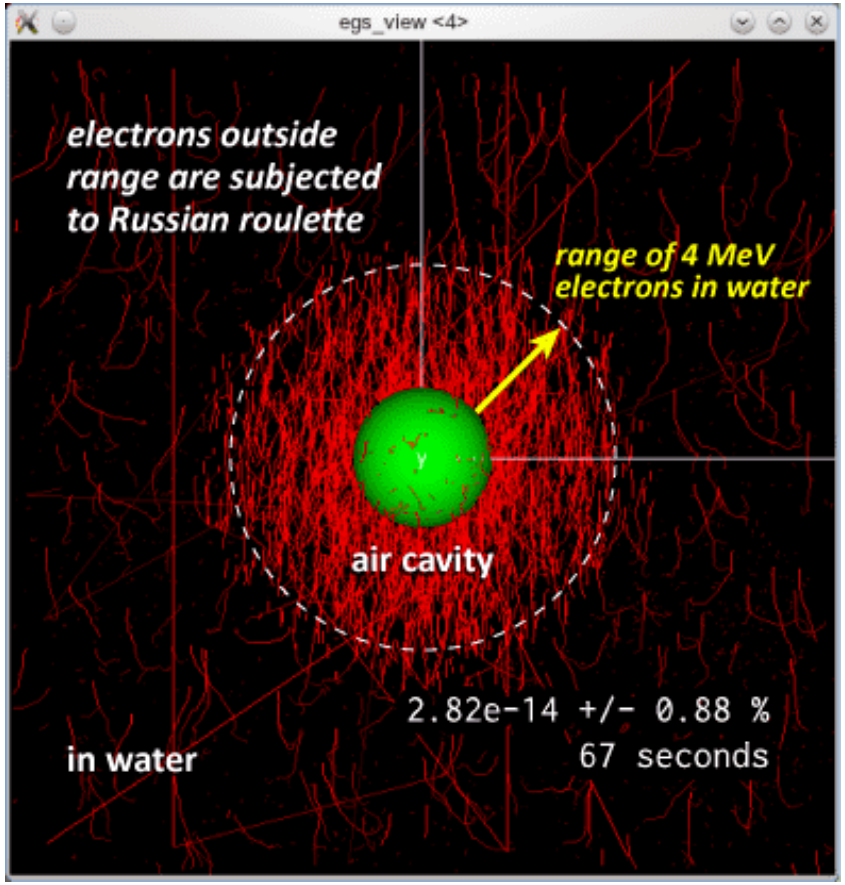
J Wulff, K Zink, I Kawrakow. **Med. Phys. 35,** 4, 1328–1336 (2008).

4. Russian roulette removes useless electrons

no variance reduction



with Russian roulette




```
rejection                = 100
Esave                    = 0.521
cavity geometry          = my_cav
rejection range medium   = water
```

Efficiency can be increased by a factor of **15** with Russian roulette!

4. Russian roulette removes useless electrons

A charged particle that is **outside** of a user-defined `cavity geometry` is subjected to Russian roulette with survival probability $1/n_{\mathrm{r}}$ when either:

- its range is not sufficient to leave the current region; or
- it is further away from the cavity than its range in the `rejection range medium`.

If the charged particle is **inside** the `cavity geometry` and its range is not sufficient to leave the cavity, its energy is deposited immediately if its energy is below E_{save}.

```
:start variance reduction:

    :start range rejection:
        rejection       = 100         # n_r: survival probability = 1/n_r
        Esave           = 0.521       # in-cavity roulette threshold (MeV)
        cavity geometry = my_cav      # name of a user-defined geometry
        rejection range medium = air  # medium with smallest stopping power
    :stop range rejection:

:stop variance reduction:
```

4. Russian roulette removes useless electrons

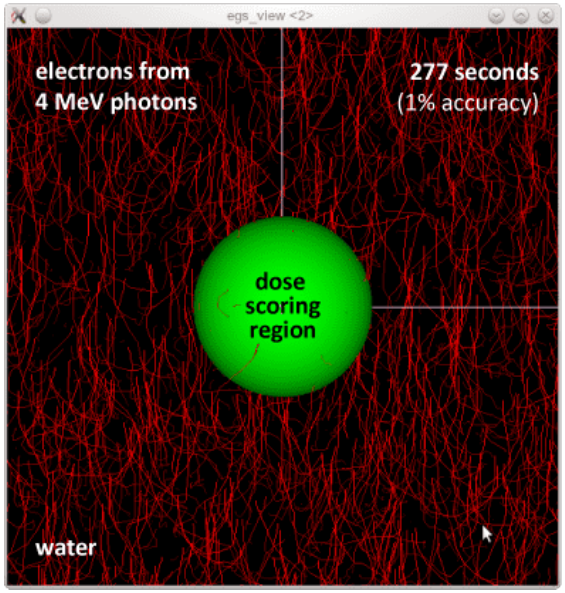
Three important caveats:

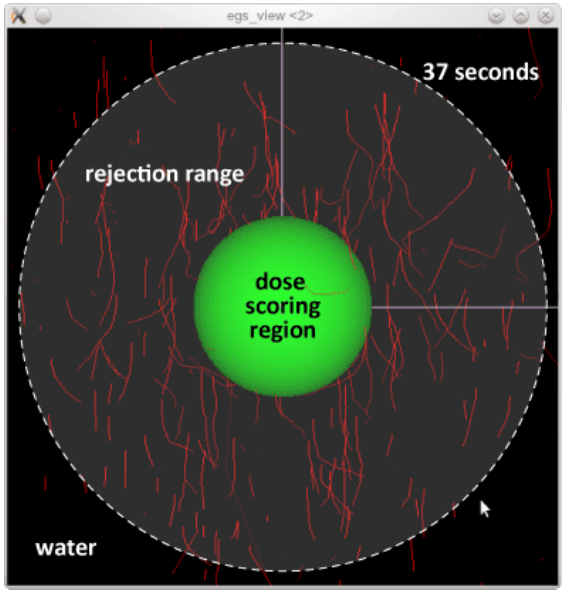
1. It is worth **optimizing** the rejection number factor for each simulation context.
2. in `egs_chamber`, you should specify a `cavity geometry` for each one of your geometries, inside the `calculation geometry` input block (you must nevertheless provide a *dummy* cavity geometry input in the variance reduction block).

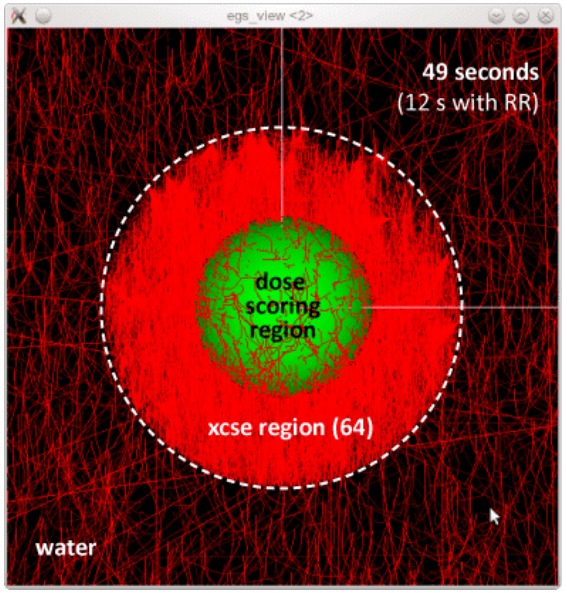
```
:start scoring options:
    :start calculation geometry:
        geometry name   = my_world
        cavity regions  = 2 5 76
        cavity mass     = 42.31
        cavity geometry = cav_world  # rr cavity for THIS geometry
    :stop calculation geometry:
:stop scoring options:
```

3. If you set `rejection = 1`, then particles are range-rejected without playing Russian roulette: if the particle cannot leave the current region and its energy is below $E_{\rm save}$, then its energy is deposited immediately. This becomes an **approximate** efficiency improvement technique.

5. XCSE generates more useful electrons







Photon splitting is useful in photon beams if dose depositions is of interest **everywhere**. But the typical situation in dosimetry is: **small ion chamber** in a **large phantom**, for example a $1\,\mathrm{cm}^3$ air cavity in a $30 \times 30 \times 30\,\mathrm{cm}^3$ water phantom. Splitting photons everywhere would in that case become very inefficient.

XCSE offers a region-based virtual increase of the photon cross-section by a user-defined factor to enhance secondary electron production in that region (adjusting particle weights accordingly).

5. XCSE generates more useful electrons

First **modify your geometry:** add regions surrounding the scoring regions, in which you will turn on XCSE. Then adjust your calculation geometry and variance reduction input blocks as follows:

```
:start scoring options:

    :start calculation geometry:
        # ...
        enhance regions =  3  4  5 ...    # list of regions to enhance
        enhancement     = 64 64 64 ...    # the enhancement factors
    :stop calculation geometry:

:stop scoring options:

:start variance reduction:

    cs enhancement = 1                    # to turn on xcse

:stop variance reduction:
```

5. XCSE generates more useful electrons

Four important caveats:

1. It is worth **optimizing** the XCSE factor for each simulation context (the optimal value is often in the range 32–128).

2. Rule of thumb for calculations in a water phantom: add a **1 cm water region** in each direction surrounding the detector, in which XCSE is turned on.

3. It is rarely useful to use different XCSE factors concurrently, but if so, then any factor must be divisible by all smaller factors (e.g., 2, 4, 8, 16, 32).

4. The **rejection** number **must be divisible by, and larger than, ALL of XCSE factors.** Just stick to powers of 2, and ensure that $n_{\mathrm{r}} > \max\{n_{\mathrm{xcse}}\}$.

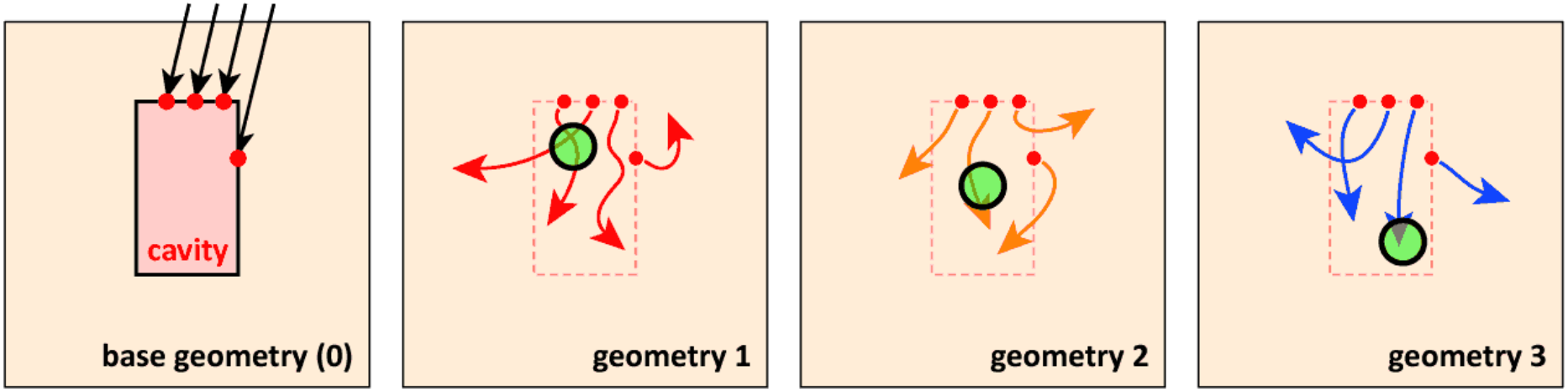
6. IPSS: hold on to your dear particles!

Many simulations involve local changes between the different calculations geometries, i.e., most of the geometry remaining identical between them: it is a waste of time to recalculate particle trajectories in the portions of the geometry that don't change.

IPSS uses the first calculation geometry as a **temporary phase space:**

- the particle is stored on-the-fly as it enters the "cavity" region of the first geometry
- the particle is transported from there successively in all other calculation geometries

IPSS is turned on with `TmpPhsp = 1` in the variance reduction input block.



6. IPSS: hold on to your dear particles!

```
:start scoring options:

    :start calculation geometry:
        geometry name   = phsp
        cavity regions  = 1       # "save" particles that reach the cavity
    :stop calculation geometry:

    :start calculation geometry:
        geometry name   = geom_1
        cavity regions  = 234
    :stop calculation geometry:

    :start calculation geometry:
        geometry name   = geom_2
        cavity regions  = 234
    :stop calculation geometry:

:start scoring options:
```

7. Correlated sampling

Perturbation factors are calculated as dose ratios, e.g., $P_{\mathrm{cel}} = D_{\mathrm{nocel}}/D_{\mathrm{cel}}$.

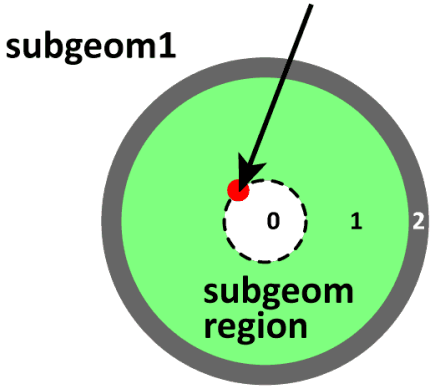
Usually only very small portion of the geometry changes: **take advantage of correlations** to reduce the uncertainty for dose ratios:

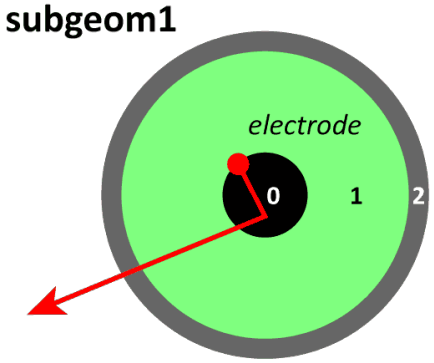
$$c = \frac{x}{y}; \qquad \left(\frac{s_{\overline{c}}}{\overline{c}}\right)^2 = \left(\frac{s_{\overline{x}}}{\overline{x}}\right)^2 + \left(\frac{s_{\overline{y}}}{\overline{y}}\right)^2 - \frac{2\,\mathrm{cov}(x,y)}{(N-1)\,(\overline{x}\,\overline{y})}$$

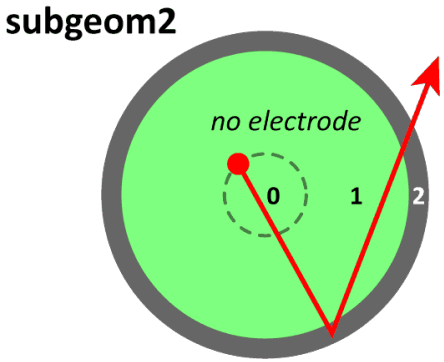
- dose for one history is scored in one geometry and copied to other geometry until particle reaches pre-defined regions.
- phase-space is stored (as in IPSS) and reused in sub-geometries, so as to *maximize* correlations.

You must turn on intermediate phase-space scoring via `TmpPhsp = 1` to use correlated sampling.

7. Correlated sampling








```
:start scoring options:

    :start calculation geometry:
        geometry name   = subgeom1
        cavity regions  = 1
        sub geometries  = subgeom1 subgeom2
        subgeom regions = 0  # regions that differ
     :stop calculation geometry:

    :start calculation geometry:
        geometry name   = subgeom1
        cavity regions  = 1
    :stop calculation geometry:

    :start calculation geometry:
        geometry name   = subgeom2
        cavity regions  = 0 1
    :stop calculation geometry:

    correlated geometries = subgeom1 subgeom2

:stop scoring options:
```