Question Set E:

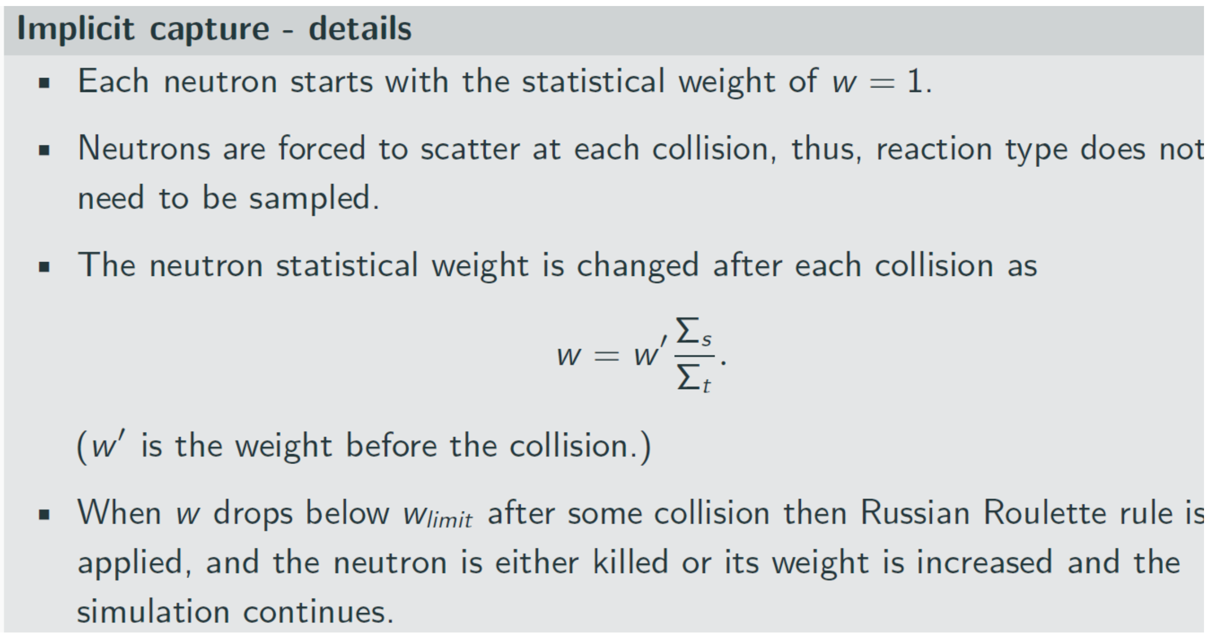
1. Describe the idea of the implicit capture.

   Implicit capture (survival biasing) is designed for shielding simulations. The method can be applied by altering the probability distribution for sampling the reaction type. Meaning that only the scattering reactions is selected to improve the chances that some neutrons will interact with the desired material (as the neutrons travel through an absorbing material).

   As we know, the probability that scattering is selected is $\frac{\Sigma_s}{\Sigma_t}$
   The altered probability for sampling the scattering reaction using the implicit capture method is 1
   There for the correction factor is $\frac{\Sigma_s}{\Sigma_t}/1 = \frac{\Sigma_s}{\Sigma_t}$

   So, every time a new collision is sampled, the neutron rate is multiplied with the correction factor.

   Assuming that the correction factor is 0.5, The weight will decrease twice at every collision point, until the weight of the neutron is below the cut off rate. Then, the Russian roulette rule needs to be applied, meaning the neutron may be killed or may continue to be simulated with increased rate.
   The choice of the weight limit depends on the type of problem. If the problem requires many collisions, the weight limit should be set to a small value.



Implicit capture - details

- Each neutron starts with the statistical weight of $w = 1$.
- Neutrons are forced to scatter at each collision, thus, reaction type does not need to be sampled.
- The neutron statistical weight is changed after each collision as

$$w = w' \frac{\Sigma_s}{\Sigma_t}.$$

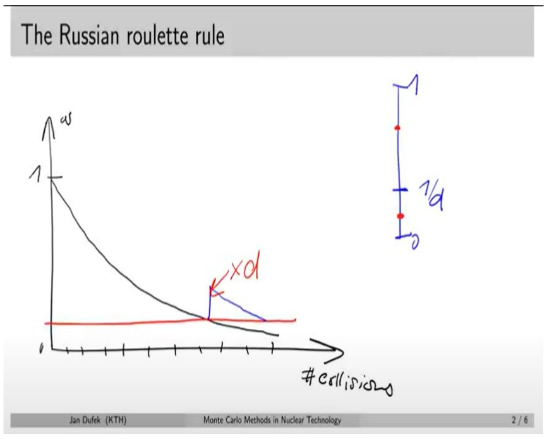
  (w' is the weight before the collision.)
- When w drops below w_{limit} after some collision then Russian Roulette rule is applied, and the neutron is either killed or its weight is increased and the simulation continues.

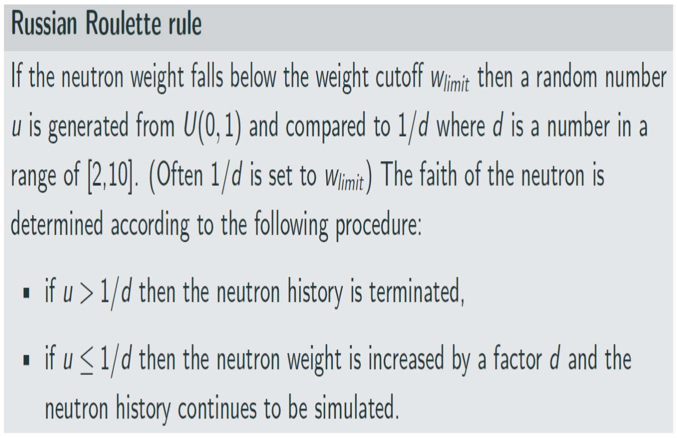
2. Describe the Russian roulette method and its purpose.

Each neutron history starts with a neutron weight which is close to 1. And it decreases with each successive collision.
After the weight drops a lot, the weight will eventually lose its impact on the results. To be smart, a limit to the neutron weight is introduced to decide what to do with the low weighted neutron histories because its simulation takes the same computing cost as the high weighted neutrons. The limit can be high or low, depending on the application.

The Russian roulette rule is used to decide what do with the neutrons that cross the limit, some neutron histories are killed, and others are allowed to survive by increasing the weight.

Russian roulette rule
This is done with random numbers.
Generates numbers btw 0 and 1
It requires that you specify a value 1/d (d= 2 to 10)
If the weight drops below the limit, a random number is generated btw 0 and 1
If the number generated is above the 1/d, then the neutron history is terminated
If however it falls below the 1/d, the statistical weight is increased by the factor of d


The Russian roulette rule
w
1
xd
#collisions
1
1/d
0
Jan Dufek (KTH)
Monte Carlo Methods in Nuclear Technology
2 / 6



Russian Roulette rule

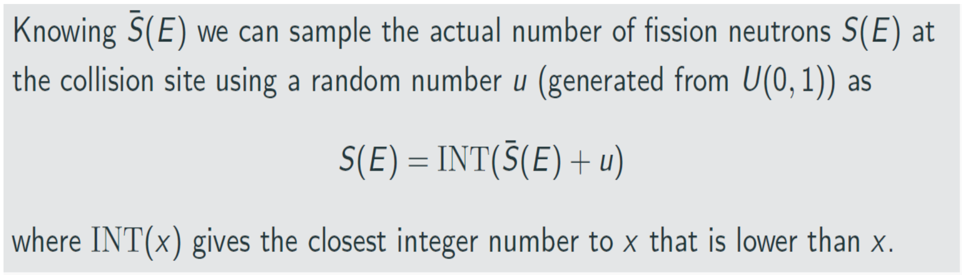
If the neutron weight falls below the weight cutoff w_{limit} then a random number u is generated from $U(0,1)$ and compared to $1/d$ where d is a number in a range of [2,10]. (Often $1/d$ is set to w_{limit}) The faith of the neutron is determined according to the following procedure:

- if $u > 1/d$ then the neutron history is terminated,
- if $u \leq 1/d$ then the neutron weight is increased by a factor d and the neutron history continues to be simulated.

The purpose of the Russian roulette is to avoid the simulation of low statistical weights, which would spend the same computing cost as large statistical weights but have very low impact to the results.

3. How is the fission source for next cycle sampled in non-analog Monte Carlo simulations?

In a fissile system, part of the drop of the neutron weight at every collision account for the fission reaction. Therefore, the implicit capture method simulates the fission reaction implicitly. Because the fission reaction is reflected to every scattering point, it is possible to evaluate the expected number of fission neutrons $\dot{S}(E)$ at every collision point by the collision or track-length estimator. The $\dot{S}(E)$ may be very small. The actual number S of the fission neutron in the collision is then decided based on the value of $\dot{S}(E)$.



Knowing $\bar{S}(E)$ we can sample the actual number of fission neutrons $S(E)$ at the collision site using a random number u (generated from $U(0,1)$) as

$$S(E) = \mathrm{INT}(\bar{S}(E) + u)$$

where $\mathrm{INT}(x)$ gives the closest integer number to x that is lower than x.

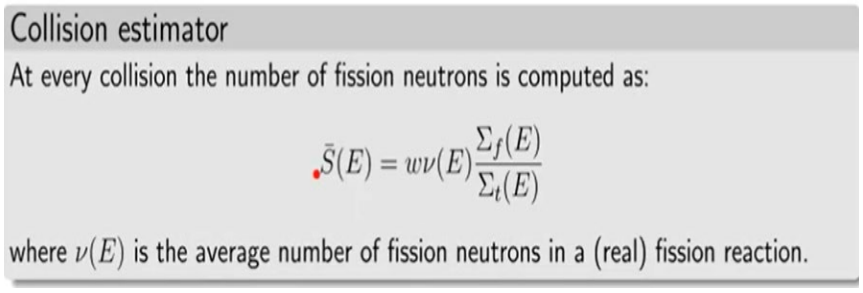
Example: $\dot{S}(E)$ = 0.1 P= 10%, s =1 u(0,1) = 0.5, new $\dot{S}(E)$ Is 0.6 = INT(0.6) Is 0 (that is zero fission neutron)

P = 90 %, s=0

Can only sample 1 fission neutron with this example If u(0,1) is > 0.9

4. Describe how the neutron flux is computed by the collision estimator.

   In order to estimate the expected number of fission neutrons Ṡ(E) at every collision point using the collision estimator. The collision estimator takes the probability that fission reaction occurs in the collision point of the specific nuclide selected $\frac{\Sigma_f(E)}{\Sigma_t(E)}$ and it is multiplied by the average number of fission neutrons v(E) and the statistical weight of the neutron that collide at the point



Collision estimator

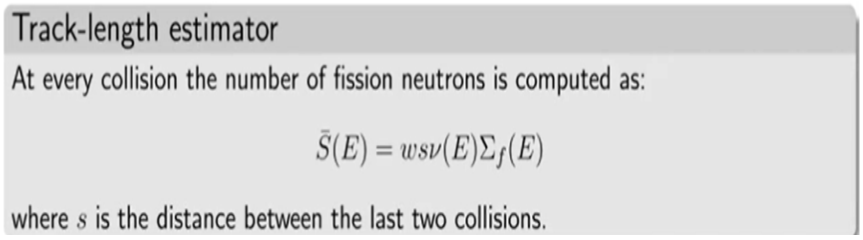
At every collision the number of fission neutrons is computed as:

$$\bar{S}(E) = w\nu(E)\frac{\Sigma_f(E)}{\Sigma_t(E)}$$

where $\nu(E)$ is the average number of fission neutrons in a (real) fission reaction.

5. Describe how the neutron flux is computed by the path-length estimator.

   The path length estimator (track-length estimator), estimates the Ṡ(E) according to the product of four factors. The first is the statistical weight, then the distance between the last two collision (s), the average number of fission neutrons v(E), and the last on is the macroscopic cross-section for fission.



Track-length estimator

At every collision the number of fission neutrons is computed as:

$$\bar{S}(E) = ws\nu(E)\Sigma_f(E)$$

where s is the distance between the last two collisions.

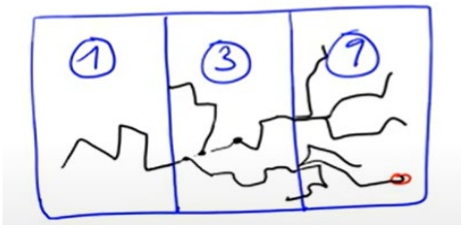
6. Describe the principle of the exponential transformation variance reduction method.

   This method can be applied to simulations where we wish to study a small part of a large system. It is used to improve a precision of a computed response of a small detector. It can be applied in both fixed source and criticality calculations.

   The method biases the transition kernel (probability density function describing the distance between the collisions) in a way that the distances between subsequent collisions get larger in the direction towards the region of interest, and the distances get shortened in the opposite direction.

7. Describe the principle of the geometry splitting variance reduction method.

   The technique is designed to attract neutrons towards a spatial region of interest. The system is partitioned into a number of regions, and each region is assigned an importance. As a neutron moves from a region of low importance to region of high importance (along the boundaries), it is split into a number of neutrons (with the same energy and direction). Although, this must be compensated by decreasing the weight of the neutron. Conversely, when a neutron travels from a high importance region into a low importance region, it is treated through a game of Russian roulette.



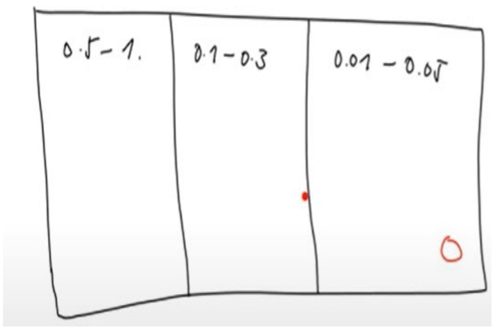
8. Describe the principle of the weight-window variance reduction method.

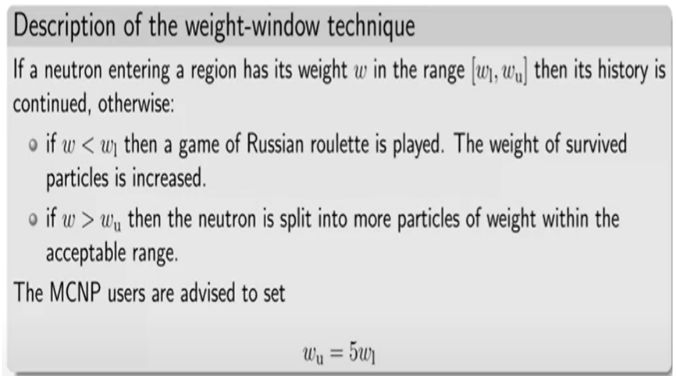
   This technique combines geometry and energy splitting method. For each space-energy region, a weight-window is defined. Each window has user-defined lower and upper-weights, w_l and w_u

   In this method unlike the geometry split, this technique splits the neutrons more in the important segment (not at the boundary).

   So, the idea of the technique is to decide a range of the statistical weights that are allowed in different segments of the system. The segments with high importance, the range of statistical weight that are allowed in them will be very small (eg. 0.01-0.05), while the least important segments will have large statistical weights (eg. O.5-1).

   How the method works
   So, when a fission neutron is born in the most important segment, it is typically assigned a statistical weight of 1, so the neutrons is split into 20 neutrons (since the maximum weight allowed is 0.05). Eventually the rate will decrease as they scatter because of the implicit capture, and when it gets below 0.01 weight the Russian roulette rule will be used to decide its faith. If the neutron is crossing the boundary, also the Russian roulette rule will be used to decide its faith.





Description of the weight-window technique

If a neutron entering a region has its weight w in the range $[w_l, w_u]$ then its history is continued, otherwise:

- if $w < w_l$ then a game of Russian roulette is played. The weight of survived particles is increased.
- if $w > w_u$ then the neutron is split into more particles of weight within the acceptable range.

The MCNP users are advised to set

$$w_u = 5w_l$$

9. Comment on efficiency of parallel fixed-source and criticality Monte Carlo simulations.

   Depending on the type of problem, the efficiency of the calculation may not scale linearly with the number of processors.

   The fixed source calculations can be parallelised very easily, since all neutron histories are independent. The neutron histories are simply divided among the available CPUs where they can be simulated independently, and the results can be collected at the end of the simulation. Although there are some procedures that cannot be parallelised. e.g loading the data library or data transfer.

   The criticality simulations are more difficult to run in the parallel mode due to the dependence between the cycles. The fission source is dependent in each cycle on the simulations in the previous cycle. Can not simply split the neutron histories among the processors, because we do not know where they should start.

10. Describe the master/slave scheme for parallel Monte Carlo criticality simulations.

    A mater process and a number of slave processors, the master is controlling the process and splits the neutron batch size among the slave processors to simulate a cycle in parallel, then the master combines the new source from the parallel simulation and is normalised, and then splits the neutrons again for the next cycle. Until all cycles are complete.

11. What is the purpose of running independent parallel Monte Carlo criticality simulations.

    This approach runs independent criticality calculations in parallel with different seeds in RNGs, and combines results at the end of all simulations (TRIPOLI)

    The benefit is in reducing the random noise.

12. Summarize advantages/disadvantages of the master/slave scheme and the independent parallel simulations.

Master/Slave scheme

- **Advantage:** Fission source can converge faster in wall clock time
- **Disadvantage:** Poor CPU utilization, because the processes need to communicate between the master processor and the slave processors. Also the slave processors might take different computing time to simulate because the no of collisions might be different with different neutron histories, and the master processor has to wait till all slave processors complete the simulation (it is not recommended to use more than 10 processor for a criticality simulation)

Independent parallel

- **Advantage:** No communication between CPUs, so full CPU utilisation is possible
- **Disadvantage:** All parallel simulation must perform a certain number of inactive cycles. The total computational time wasted in inactive cycles may be enormous (the real strength with this approach is when your fission source has already converged at the beginning of the simulation)