# Chapter 4  Variance Reduction Techniques

## 4.1 PDF-modifying Variance Reduction

In the next few sections, we will look at **heuristic** variance reduction techniques. Of course, it is not really the techniques themselves which are heuristic, but rather the explanations that we will offer to justify them (at this time). We will look at the most popular variance reduction techniques:

- Weighting in lieu of absorption
- Splitting
- Forced collisions
- Russian roulette
- Exponential transform
- Source biasing (in position, energy, and direction).

Before we start in with this list, we will take a look at the mathematical **rules** by which we can legitimately deviate from the purely analog random walk that we looked at previously and apply them to event-based problems we already know how to use.

### *Mathematical basis of cheating*

Okay. Maybe it isn't really cheating; it just **feels** like cheating when we "fool Mother Nature" by choosing from a probability distribution that WE want to use rather than using the one that the physical data indicate.

The first extra mathematical trick that we need to introduce is the idea of a particle **weight**. From our particle transport point of view this is expressed by the idea that a particle we are tracking is not necessarily <u>**a**</u> particle -- it might be half of a particle or two particles, etc. The "number of particles" that the object we are tracking represents (at any given time) is represented by the numerical value of the weight. And our basic tool for carrying out our "cheating" is the adjustment of this weight to keep the method unbiased.

So, if the probability distribution dictated by the physics is $\pi \;\; x$ and we want to use a second distribution $\pi \;\; x$, we can do it if we use a weight correction, $w_{corr}$ (that multiplies the previous weight of the particle to get the new weight ) **AFTER** the desired distribution has been used to pick the value $x$. The weight correction is chosen to maintain the overall "expectation" for the choice:

True probability of choosing within dx about x = ("Used" probability of choosing within dx about x) x (Weight correction)

$$\pi(x)dx = \tilde{\pi}(x)dx \cdot w_{corr} \tag{4-1}$$

Therefore,

$$w_{corr} = \frac{\pi(\hat{x}_i)}{\tilde{\pi}(\hat{x}_i)} \tag{4-2}$$

or, in the immortal words of D. Bentzinger, “Shoulda over did.” (I have inserted the caret and the subscript into the previous equation to emphasize that the weight correction is determined from the ratio of the two PDF’s AT THE VALUE CHOSEN using the one in the denominator.)

One of the typical restrictions specified for the “new” distribution is that all possible x’s in the original distribution are still possible in the new distribution. (In mathematical terms, the domain of the original distribution must be a subset of the domain of the new distribution.) If this is not obeyed, then there are x’s that are supposed to be possible which will never actually be chosen. It turns out that you CAN design your Monte Carlo process so that possible choices are made impossible; but the answer will be biased UNLESS YOU ARE CERTAIN that the choices that are eliminated have NO POSSIBILITY of contributing to the answer. We will see an example or two of this.

---

**Example**: One of the simplest uses of this procedure occurs if you are programming an application that is supposed to make a choice of x in the domain (a,b) using some complicated distribution $\pi(x)$ and YOU DON’T WANT TO DO IT. (Maybe because it is hard to sample from, maybe because you think it might reduce the variance, or maybe just because you don’t think it is worth the trouble.)

For example, it is entirely unbiased (i.e., okay) for you to choose an x in the domain uniformly and the adjust the contribution weight by multiplying the previous weight by the correction.

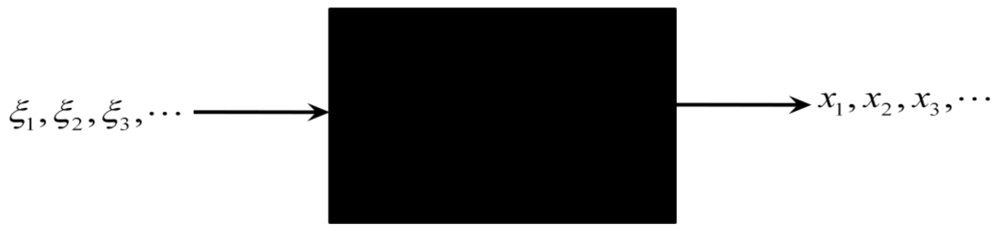
Unfortunately, although it is unbiased to do this, it is also usually unwise. The reason for this is that, if you choose from the “correct” distribution, each of the chosen values of x will have a weight of 1; if you use the flat distribution, the weights will vary according to $\pi(x)$, which usually means an increase in variance. Therefore, all other things being equal, using this method will generally **increase** the variation in the process and therefore, the variance of the result. (On the other hand, if I were programming a “quick and dirty” MC solution to some problem and this particular distribution were my only obstacle, I would not hesitate to do it this way to save developer time by sacrificing computer time.)

---

Now that we know HOW to implement a non-physical distribution, let’s examine the issue of WHY?

## *Why do it?*

We do it to make our Monte Carlo programs better. “Better” means “lower variance.” Lower variance, in view of our “black box” view of Monte Carlo:



means “have the x’s come out as nearly identical as possible.”

## *Focus on weight corrections*

The key idea for designing variance reduction methods this way—through changing PDFs with their attendant weight corrections—is that it is possible to work the problem backwards: Decide the desired weight corrections to reduce variance and then choose the PDFs that coincide with the weight corrections. This is best shown with an example.

---

**Example**: For the simple two choice game represented by the table below, choose th optimum PDFS to use to minimize variance (in fact, reduce it to zero).

| Choice i | Prob, $\pi_i$ | $x_i$ |
|---|---|---|
| 1 | 0.8 | 1 |
| 2 | 0.2 | 6 |

If this is implemented without variance reduction, we would expect a value of:

$$\bar{x} = (0.8)(1) + (0.2)(6) = 2$$

with resulting variance:

$$\sigma^2 = (0.8)(1-2)^2 + (0.2)(6-2)^2 = 4$$

If, however, e make choice #1 with probability 0.4 and choice #2 with probability 0.6, the weight corrections result in:

| Choice i | Prob, $\pi_i$ | $x_i$ | $\tilde{\pi}_i$ | $w_{corr}$ | $\tilde{x}_i$ |
|---|---|---|---|---|---|
| 1 | 0.8 | 1 | 0.4 | 2 | 2 |
| 2 | 0.2 | 6 | 0.6 | 1/3 | 2 |

The result of applying these alternate probabilities is that both choices result in a score of 2; therefore, since every sample gives the correct answer, the variance is zero.

## 4.2 Figure of Merit (FOM)

Although up to now in the course I have referred to improvements to the Monte Carlo "black box" as "variance reduction" methods (in keeping with the statistical formulas that we developed at the beginning of the course), in practical situations reducing variance is not our primary reason for adding them to our algorithms: We are actually interested in increasing the efficiency of our Monte Carlo methods by getting more accuracy per unit computer time rather than accuracy per particle history (i.e., which sample variance measures and which subsequently drives the standard deviation of the mean). Thus, the metric of efficiency called the figure-of-merit or FOM includes computer time as well.

In order to have the metric go UP as improvements are made to a Monte Carlo process, the figure of merit is defined as:

$$FOM = \frac{1}{S_{\hat{x}}^2 T} \tag{4-3}$$

where:

$$\begin{aligned} S_{\hat{x}}^2 &\equiv \text{Estimate of variance, and} \\ T &\equiv \text{Computer time} \end{aligned}$$

Recalling our previous result:

$$S_{\hat{x}}^2 = \frac{S^2}{N} \tag{4-4}$$

where:

$$\begin{aligned} S^2 &\equiv \text{Estimate of variance of the process (which} \\ &\quad\ \text{which will not change during the Monte Carlo} \\ &\quad\ \text{run)} \\ N &\equiv \text{Number of histories run} \end{aligned} \tag{4-5}$$

then we should expect that (by multiplying both sides by $N$):

$$S^2 \sim \text{constant} \cong N S_{\hat{x}}^2 \tag{4-6}$$

Under the assumption that each history will take, on average, the same amount of computer time, we have the result that:

$$NS_{\hat{x}}^2 \sim TS_{\hat{x}}^2 \sim \text{constant}$$
$$FOM \equiv \frac{1}{TS_{\hat{x}}^2} \sim \text{constant} \tag{4-7}$$

That is, the FOM should be relatively constant as a function of *N* during the running of a Monte Carlo calculation.

The FOM has several uses as a measure of efficiency:

1. When you try a new variance reduction technique, the ratio of FOMs before and after gives you the factor of improvement.

2. When a new version of a Monte Carlo code is released, the ratio of the FOMs for identical sample problems gives you the factor of improvement.

3. When you change computers, the ratio of FOMs for identical sample problems gives you the factor of improvement.
4. When the FOM is not a constant as a function of *N* (i.e., as a single Monte Carlo calculation proceeds through its histories)—especially for a deep penetration problem—that result is not statistically stable; that is, no matter how many histories have been run, the important particles are showing up infrequently and have not yet been sampled enough.

   *[NOTE: Obviously, this brings into question every Monte Carlo transport calculation! What if you stop the calculation before some "important particle path" has been sampled even the first time? That is why you have to understand your problem, to recognize problems that might be susceptible to this difficulty.]*

## 4.3 Absorption weighting

Now that we have the math and the justification behind us, let us avoid re-inventing the wheel by looking at the non-physical distributions that have been proven over the ages to be good to use for the various decisions that we have to make.
For each of them, we will follow the same pattern:

- General description ("heuristics") of what the idea is.
- Recollection (from Lesson 6) which of the transport decisions is being adjusted and what the physical distribution is. Note that the decision steps were:

  1. Particle initial position
  2. Particle initial direction
  3. Particle initial energy

4. Distance to next collision.
5. Type of collision
6. Outcome of a scattering event

- Mathematical layout of the non-physical distribution that we will use.
- The resulting weight correction that we will have to apply to keep the solution un-biased.

***NOTE:*** *Although I will mention it again later, this "heuristic" approach has a much different feel than the more mathematical approach that we will see later. The heuristic approach starts with a (hopefully) improved DISTRIBUTION to use and the weight corrections are calculated as if they were the "penalty" that must be applied to keep the game fair. In the mathematical approach, as we will see, we flip this around by figuring out WEIGHT CORRECTIONS we want and then computing the proper distributions to use to deliver these (hopefully) superior weight distributions.*

## *Weighting in lieu of absorption*

This is the most commonly used variance reduction technique. In fact, in many transport codes, this option cannot be turned off.

***General description:*** The basis idea is to PREVENT particles from absorbing. Then particles will live longer and have more of a chance to score.

***Which of the transport decisions is being adjusted:*** #5. Type of collision.

***Mathematical layout:***
Let us assume the two possible outcomes are scattering and absorption. The 6atural ("shoulda") PDF's are:

- Scattering with probability $\pi_1 = \frac{\Sigma_s}{\Sigma_t}$
- Absorbing with probability $\pi_2 = \frac{\Sigma_a}{\Sigma_t}$

Our decision is to pick scattering with a 100% probability. ("did")

***Resulting weight correction:***

$$w_{corr} = \frac{\pi_s}{\tilde{\pi}_s} = \frac{\frac{\Sigma_s}{\Sigma_t}}{1} = \frac{\Sigma_s}{\Sigma_t} \tag{4-8}$$

Note that this is an example of what I mentioned before: we have made a normally POSSIBLE choice (absorption) IMPOSSIBLE.  This is allowed ONLY because we are absolutedly certain that absorbed particles have NO possibility of contributing to the answer.
It is interesting to note also that, although we are guaranteed to have a lower variance using this procedure, we are ALSO guaranteed to have higher computer run times, since each history will be longer.  It may or may not be true that the variance reduction is WORTH the extra time.  This is one of the issues we will address in the in-class exercise.

Another way of looking at this technique is to imagine that we have divided the particle into two parts: the fraction that scatters and the fraction that absorbs.  We continue to follow only the first (scattering) part and let the fraction that absorbs die.  This makes it a special case of the idea of "splitting" which is covered next.

## 4.4 Splitting and Russian Roulette

### *Splitting*

"Splitting" is a technique in which a particle being tracked is divided into multiple pieces that each are individually followed.  The mathematical aspects of splitting are the basis for other variance reduction techniques.

There are two common situations in particle transport that involve splitting:

1. The actual physics sometimes involves particle splitting, (e.g., (n,2n)), when TWO particles come out of a collision.  In general, one of them is followed as a continuation of the current particle history, and then, after the original particle history is over, we come back and "pick up" the second particle from the original collision site and follow IT to conclusion.  ("Banking")

   *[Note that this does not really fit the pattern of MODIFYING the probability distribution.  Instead, it is based on AVOIDING a discrete decision by following ALL of the options. Since a 7robabilistic decision is avoided, the decision's contribution to the variance is avoided.]*

2. As a Monte Carlo technique to increase efficiency, a particle can be ARTIFICIALLY split into two or more "pieces" when an "important" region is entered in order to provide a better sampling of the region. There is no physical particle division; this is just a variance reduction technique.

The resulting weights of the emitted particles is different in these two situations.  For a physical splitting situation, the post-collision weights of the multiple particles is driven by their relative yields from the physical reaction.  For the example of the (n,2n) reaction, since each of the two particles is emitted 100% of the time, the actual doubling of the number of neutrons is simulated by the fact that EACH of the two particles keeps the original particle weight. On the other hand, a particle that is only emitted 40% of the time should be assigned a weight of 0.40 times the original particle weight.

An example will make this idea clearer for the general mathematical case (and give you an idea of how the banking is actually performed in the code). Assume that a history (in progress) with weight of 0.5 faces a discrete decision with three possible outcomes:

- State 1 with a 60% probability;
- State 2 with a 30% probability; and
- State 3 with a 10% probability.

The idea of splitting is that, instead of choosing between the three possible resulting states, we follow each of them in turn. In this particular case, we continue the history by following the history into State 1 with a weight of 0.30 (=60% of original weight of 0.5) and “bank” (i.e., save the complete location, new direction, and new energy in a special array of “states” to be continued later) one history in State 2 with weight 0.15 (=30% of 0.5) and one history in State 3 with weight 0.05. We would proceed to the end of the first history beginning in State 1.

BUT THEN, instead of beginning the next history, we would check the bank to see if is empty.

If not, we RETURN to the state represented in the next entry of the bank and follow this particle to the end of its history. Then, we go back and check the bank again, continuing in this way until the bank is empty.

Of course, in the process of following any one of these three, we might face another discrete decision (or maybe the same one again!) which will cause more histories to be added to the bank. As you can imagine, if the code branches too many times, it might spend all of its time following bank histories and never finish the original history!

> *[NOTE: This is exactly what sometimes happens with fission. It is possible to treat the fission reaction as a “multiple upscatter” event, using the fission bank; MCNP will actually do this if you have fissionable material in a source-driven shielding problem. However, if you make the mistake of modeling a SUPERCRITICAL configuration, the resulting neutron bank will GROW with time, and the calculation will encounter a single source neutron whose life never ends, so the code will get “stuck”! That is one of the reasons that we developed the special techniques for handling k-effective problems that we discussed in a previous section.]*

## ***Russian roulette***

Like splitting (and in many ways its OPPOSITE) Russian Roulette is not so much a variance reduction technique as a mathematical tool that is needed for implementing variance reduction techniques.

In contrast to splitting, which serves to CREATE several particles out of one particle, Russian Roulette COMBINES several particles into one particle. Of course, since we only follow a single particle at a time, the mathematics takes the form of having a particle DIE with a probability of $1-p$ (typically 90-99%). To keep the method unbiased, if a particle survives, its weight is increased by a factor of *1/p*.

The need for this tool is obvious when used in combination with absorption weighting and forced collision (to be discussed later)—the latter two methods eliminate BOTH ways of ending a history. Without the (artificial) death condition added by Russian Roulette, the first history would never end!

In practice, Russian Roulette is performed whenever a particle's weight falls below a lower weight cutoff, $w_{cut}$. So, although this technique actually INCREASE the problem variance, it increases the efficiency of a Monte Carlo process by saving the computer time that would otherwise be spent (wasted?) following low weight particles.

## 4.5 Forced collisions

***General description:*** The basis idea of the forced collisions technique is to FORCE a collision in a particular section of the path of a particle. We will use it in a particular way, by forcing the collision in the range of the path that is INSIDE the problem. Like the absorption weighting technique, this is expected to cause particles to live longer and have more of a chance to score.

***Which of the transport decisions is being adjusted:*** #4. Distance to next collision.

***Mathematical layout:***
In general, this is another splitting technique: the particle is divided between the part that DOES collide in the desired region and the part the DOES NOT collide in the desired region. Therefore the actual mathematics depends on WHERE the particle is being forced to collide. In the version we will look at, we assume that the Monte Carlo calculates the distance to the boundary (in the direction that the particle is travelling) to be $\tau_0$, and the numbers look like this

Actual probability distribution:

- Probability of escaping: $\pi_{escape} = e^{-\tau_0}$
- Non-escape with probability: $\pi_{non-escape} = 1 - e^{-\tau_0}$

Our decision is to pick non-escape with probability 1.00.

***Resulting weight correction:***

$$w_{corr} = \frac{\pi_{non-escape}}{\tilde{\pi}_{non-escape}} = \frac{1-e^{-\tau_0}}{1} = 1-e^{-\tau_0} \tag{4-9}$$

Now, of course, we have to remember the splitting "roots" of this procedure, and recognize that the "other " part of the particle DOES escape. Therefore, we should contribute:

$$we^{-\tau_0}$$

to the leakage of the closest boundary, where w is the weight BEFORE the correction.

Again, it is possible that the longer computer run time will hurt more than the lower variance will help. In fact, most of the literature indicates that this is USUALLY that case, so that "non-escape" forced collision is seldom used.

> *[NOTE: The "DXTRAN sphere" method is the name of a MCNP technique that falls into the general category of "forced outcome methods." This will be covered in a later section.]*

## 4.6 Exponential transform

***General description:*** The basis idea of the exponential transform technique is to make it EASIER for a particle to travel in a desired direction by THINNING the material in the desired direction to make it easier to cover ground. At the same time, the material is made THICKER in directions away from the desired direction. It is important to remember that this method does not make it more LIKELY that desired directions will be chosen, just easier to travel in desired directions.

***Which of the transport decisions is being adjusted:*** #4. Distance to next collision.

***Mathematical layout:***
The basic idea is to change the total cross section and make it dependent on the direction travelled. Denoting the cross section used with an asterisk, we have:

$$\Sigma_t^*(\hat{\Omega}) = \Sigma_t(1 - p\hat{\Omega}\cdot\hat{\Omega}_d) \tag{4-10}$$

where ***p*** is a general parameter chosen by the user (or programmer) with 0<p<1 and $\hat{\Omega}_d$ is the desired direction. Note that the adjusted cross section is lowest if the direction of travel is the SAME as the desired direction, being reduce by the factor (1-p). Conversely, the adjusted cross section is greatest if the direction of travel is OPPOSITE to the desired direction, being increased by the factor of (1+p). Because of the limitations on p (0<p<1) these minimum and maximum values range from 0 to twice the true cross section.

As we have seen in our in-class exercise, Decision 4 really involves two steps: (1) choosing the number of mean free paths travelled to the next collision and (2) translating this into a new position using the total cross sections of regions that are enterred by the particle. The exponential transform just involves modifying the cross sections used in the second step.

There are actually two cases that have to be considered: The particle reaches the outer boundary or it collides inside the problem geometry.

Actual probability distribution:

- Probability of escaping: $\pi_{escape} = e^{-\tau_0}$
- Probability of next collision at distance *s*: $\pi(s) = \Sigma_t e^{-\Sigma_t s}$

For each of these two conditions, the probability distributions actually used are:

- Probability of escaping: $\tilde{\pi}_{escape} = e^{-\tau_0 (1 - p\hat{\Omega}\cdot\hat{\Omega}_d)}$

Probability of next collision at distance s: $\tilde{\pi}(s) = \Sigma_t (1 - p\hat{\Omega}\cdot\hat{\Omega}_d) e^{-\Sigma_t (1 - p\hat{\Omega}\cdot\hat{\Omega}_d) s}$

***Resulting weight correction:***
The weight correction depends on which of the two events actually occurred:

- If the particle escaped:

$$w_{corr} = \frac{\pi_{escape}}{\tilde{\pi}_{escape}} = \frac{e^{-\tau_0}}{e^{-\tau_0 (1 - p\hat{\Omega}\cdot\hat{\Omega}_d)}} = e^{-\tau_0 p\hat{\Omega}\cdot\hat{\Omega}_d} \tag{4-11}$$

- If the particle collided at distance *s*:

$$w_{corr} = \frac{\pi \;\; s}{\tilde{\pi} \;\; s} = \frac{\Sigma_t e^{-\Sigma_t s}}{\Sigma_t \;\; 1 - p\hat{\Omega}\cdot\hat{\Omega}_d \;\; e^{-\Sigma_t \;\; 1 - p\hat{\Omega}\cdot\hat{\Omega}_d \;\; s}} = \frac{e^{-\Sigma_t p\hat{\Omega}\cdot\hat{\Omega}_d s}}{1 - p\hat{\Omega}\cdot\hat{\Omega}_d} \tag{4-12}$$

Notice that it is the second weight correction above that keeps us from using p=1 (i.e., turning the cross section into a void in the direction of travel) because the weight correction would approach infinity as the direction of travel approaches the desired direction.

## 4.7 DXTRAN sphere

The DXTRAN sphere method is the name of the MCNP technique that is related to the category of "forced collisions" studied in a previous section, but has a few twists in implementation.

The basic idea is that of a splitting followed by a forced transport (but not necessarily a collision). That is, an initial release of a particle and on each subsequent scattering of the particle, we divide the particle into two pieces:

- One piece continues on its normal path with NO CHANGE in particle weight. No weight correction is the first "twist".
- A second particle is released with the cone of directions that point at a user-specified sphere (somewhere in the problem geometry) and the particle is forced to reach the point

of first intersection with the sphere. (At that point of intersection, the particle is still in flight—no collision is forced. This is the second "twist".)

This second particle (referred to as the "DXTRAN particle") is assigned a weight equal to the original particle weight times the directional probability (i.e., the fractional solid angle of the sphere times the PDF for the direction chosen) times the probability of surviving the trip (i.e., e to the power of the number of mean free paths to the DXTRAN sphere intersection point.)

- The bias introduced by not reducing the initial particle weight is statistically compensated for b y adding a condition that if the non-DXTRAN particle crosses the DXTRAN sphere, it is killed.

The usefulness of the DXTRAN approach is the power it gives the user to draw particles toward important regions of the problem. The efficient use of the method is very much the "art" of properly placing them where "successful" particles will go.

I have most often seen DXTRAN spheres placed such that they surround a detector. But I have also seen it used to draw particles into an important streaming duct in a problem that might otherwise end up undersampled by MCNP.

The main problem with this technique is the added computational burden imposed by the need to track all of the split particles. MCNP deals with this by eliminating most of the DXTRAN spheres by immediately playing Russian Roulette; because the weight of the particle has been reduced so much (i.e., by the fractional solid angle of the sphere times the exponential attenuation factor of getting the particle to the sphere).

## 4.8 Source biasing

***General description:*** We have saved until last the most general of the variance reduction techniques. The basic idea is simple, but very foggy: Instead of using the true distribution, use some other distribution, i.e., that you have some reason to believe is better. (What does "better" mean? See "Why Do It?" above.)

***Which of the transport decisions is being adjusted:*** #1-#3. Initial source position, energy, and direction.

***Mathematical layout and weight correction:***
In the basic layout of the idea, no guidance is actually given about distributions to use.

Therefore, all we have is the basic theory laid out above in the "Mathematical basis of cheating" section:

"So, if the probability distribution dictated by the physics is $\pi\ x$ and we want to use a second distribution $\tilde{\pi}\ x$, we can do it if we use a weight correction, $w_{corr}$ (that multiplies the previous weight of the particle to get the new weight ) **AFTER** the desired distribution has been used to pick the value $x$. The weight correction is chosen to maintain the overall "expectation" for the choice:

$$w_{corr} = \frac{\pi(\hat{x}_i)}{\tilde{\pi}(\hat{x}_i)} \tag{4-13}$$

***Choosing the modified distribution to use***
Since the basic layout gives you no guidance in choosing the actual distributions to use, I feel like I need to give you some basic advice on the subject. In general, you want to modify the natural distributions in order to favor choices that are more IMPORTANT. What is "importance"? To us the answer is simple:

IMPORTANCE = EXPECTED CONTRIBUTION

Therefore, our job is to modify the distributions to favor following the particles that are the expected to contribute the most to the "score" that you care the most about. We will learn later that, if you know the imporance of each of your possible choices, $I(\hat{x})$, the optimum choice of your alternate distribution is given by:

$$\tilde{\pi}(\hat{x}) \sim \pi(\hat{x})I(\hat{x}) \tag{4-14}$$

The successful approaches I have seen to picking alternate distributions fall into the following three categories: (1) Heuristic (i.e., seat of the pants) choices, (2) Experimental choices, and (3) Adjoint-flux-based choices. Let's look at each of these briefly.

*Category 1: Heuristic*
This is the technique that is best suited for this lession (see title of lesson). Modify the natural distribution to favor the choice of particles that you think MUST BE more important.

---

**Example:** Source particle location (Decision #1) If you are interested in determining the right leakage, pick source locations preferentially to the right.

---

**Example:** Source direction (Decision #2) If you are interested in determining the left leakage, pick source directions preferentially heading to the left.

---

**Example:** Source energy (Decision #3) If you are interested in deep penetration, pick source energies where total cross section is low.

---

The problem, of course, is that you are left to your own intuition about HOW MUCH to favor the more important particles. Therefore, this approach tends to be trial and error.

*Category 2: Experimental*
This technique is based on you running a few (hopefully) short "test runs" to get an idea of the

relative importance of various initial source choices. The test cases correspond to restricting the choices of one of the variables to a sub-domain, running a short problem, and interpreting the resulting answer as the importance of the sub-domain.

---

**Example:** Assume you have a 1D slab shielding problem in which particles are born uniformly in the range 0<x<10 and you want to bias the choice of initial position to improve your leakage statistics on some distant surface of the geometry. You run a two short test problems: One in which particles are born uniformly in 0<x<5 and a second one in which they are born uniformly in 5<x<10. If the answers for you two problems are 0.001 and 0.01, respectively, how should you optimally bias the source choice?

**Answer:** The optimum choice would be to choose the position uniformly in 0<x<5 with probability 1/11 and uniformly in 5<x<10 with probability 10/11.

---

**Example:** Same situation as Example 4, except you want to bias the source starting energy group (2 group problem). Assume the natural source distribution is 70% in group 1 and 30% in group 2. You run two short test problems: One in which the source particles are all born in group 1, and a second one in which they are all born in group 2. If the answers for your two problems are 0.01 and 0.02, how should you optimally bias the source choice?

**Answer:** The optimum choice would be group 1 with probability 7/13 (=0.7*0.1/(0.7*0.1+0.3*0.2) ) and group 2 with probability 6/13.

---

*Category 3: Adjoint flux based*
As we will study later in the course, we can actually write and solve an equation for the importance function for source particle (and scattered particle) distributions. The equation turns out to be the Adjoint Boltzmann Equation. If a solution of this equation can be obtained (or, more often, approximated), then the optimum source biasing distributions can be deduced from it. We will postpone this important topic until we have built our mathematical background a bit more.

## 4.9 Cell weighting

A second category of variance reduction techniques (which has come to be the most popular approach with the advent of mesh-based tallies and importance maps) is to <u>control</u> the particle population through judicious application of splitting and Russian Roulette <u>between decisions</u> instead of modifying the decision PDFs.

In our discussion of this family of techniques, I want to make sure you understand the underlying basis of the method: Reduction of <u>future</u> variance contribution of the particle being tracked by splitting particles when they get into high variance-producing regions and, for relative efficiency, reducing the particle population when it travels into low variance regions.

Notice my use of the phrase "high variance regions" rather than "high importance regions", the way it is usually phrased. The principal difference between the two phrasings is usually small because high importance regions typically imply high variance (especially for low-probability binomial processes, where the importance equals the variance).

In such a low-probability binomial distribution, the expected value is given by:

$$\bar{x} = p(1) + (1-p)0 = p \tag{4-15}$$

and the variance by:

$$\begin{aligned} \sigma^2 &= p(1-\bar{x})^2 + (1-p)\bar{x}^2 \\ &= \bar{x}(1-\bar{x})^2 + (1-\bar{x})\bar{x}^2 \\ &= \bar{x}(1-2\bar{x}+\bar{x}^2) + (\bar{x}^2 - \bar{x}^3) \\ &= (\bar{x} - 2\bar{x}^2 + \bar{x}^3) + (\bar{x}^2 - \bar{x}^3) \\ &= \bar{x} - \bar{x}^2 = p - p^2 \cong p \end{aligned} \tag{4-16}$$

So, the expected value (also known as the importance) and the variance come together as p decreases to zero.

In such a method, most of the variance comes from the large contribution by the very rare scoring particles. The population control methods base their success on having the scoring spread out over more scoring particles. (Even if every rare success—scoring 1—can be replaced by two successes scoring ½ each would cut the variance by a factor of 4.) For our situations, this often leads to a strategy to try to get as many particles as possible to "score".

The first of the population-control methods is cell weighting. In this method each material cell is assigned a numerical importance, $I_{cell}$ , by the user. Subsequently, when:

- a particle crosses into a region of <u>higher</u> importance, the particle is split into $I_{new}/I_{old}$ pieces. Each new particle has a weight correction of $I_{old}/I_{new}$ .

  *[NOTE: How do you split a particle into $I_{\text{new}}/I_{\text{old}}$ pieces when this ratio is not an integer? There are various ways this could be done, but the most straightforward one is the one used by MCNP:*

  - *The fractional part is used as the probability that the next larger integer is used as the number of particles after the split; or*

- *No matter how many particles result, each has a weight of* $I_{old}/I_{new}$ *times the particle weight before the collision.*

*This technique does not conserve the original particle weight (except on average), but minimizes the variation in weights.]*

- a particle crosses into a region of <u>lower</u> importance, the particle is undergoes a Russian Roulett game with probability of survival of $I_{new}/I_{old}$ . If the particle survives, the resulting weight correction is $I_{old}/I_{new}$ .

  *[NOTE: If a cell weight of 0 is assigned to a cell, then these rules would have a particle undergo Russian Roulette with <u>no chance</u> of survival—which means the particle would be killed. Such "zero importance" regions are used to mark the edge of the problem in MCNP, i.e., the "rest of the universe" cell from which re-entry into the volume of interest is impossible.*

  *When no cell-weighting importance information is available, all cells which are part of the desired problem geometry are assigned a weight of 1.00.]*

The principal shortcomings of the cell weight technique (as implemented by MCNP) are:

1. The splitting and Russian Roulette are applied indiscriminately to every particle crossing material (or grid) boundaries, whether or not a particle "needs"" it or not. The importances are based on expected particles; it has no way to customize the treatment to the individual particle weight encountered in the actual problem.

2. There is no way to put in importances by energy region, although particle importance is sometimes strongly energy dependent.

Both of these shortcomings are addressed in the weight windows method discussed next.

## 4.10 Weight windows

The second of the population control methods was designed to make up for both shortcomings of the cell weight method. The principal differences are:

1. The values of interest are inverted: Expected particle weight instead of importance (which tend to be inverses of each other).

   In the cell weighting method the cell (or grid) importances will be highest at the detector and will decrease as you go away from it; this is consistent with its interpretation of the expected contribution of a particle released in the region. Since the rules of use only use ratios of region importances, the absolute values 16ren't' important, although it is traditional to assign an importance of one the source regions.

   The weight windows, as its name implies, uses the expected particle weight as its region-based value; again, the source regions is assigned a value equal (or close) to one. From then, the weight varies inversely to the importance values, with the expectation being that a particle's weight will decrease as the particle moves from source to detector. (In fact, MCNP gives you the option of just putting in region importances and having the code invert the values to get region-average weights.)

2. In addition to the expected weight per particle, the method requires the user to input and upper and lower limit on the region weights (or, equivalently, to provide factors to multiply and divide the average region weight to get the upper and lower limits, respectively).

3. These upper, average, and lower cell weights (i.e., allowed weight "windows") are used to keep the particle weights within the windows provided, using the same rules as before:

    1. If the weight of a particle weight entering a cell region (either spatially through a boundary or from one energy range to another by scattering inside a cell) is below the lower weight allowed by the window, Russian Roulette is played aimed at bringing the weight value up to the average window value if the particle survives the game, using:

    $$p = \Pr\{\text{particle survives}\} = \frac{\text{current particle weight}}{\text{average particle weight}}$$

    with the resulting weight correction of:

    $$w_{curr} = \frac{\text{average particle weight}}{\text{current particle weight}} = \frac{1}{p}$$

    if the particle survives.

    2. If the particle weight enters a weight region with a weight that is above the upper limit of the weight window, the particle is split into a number of pieces which will bring each pieces weight back up to the average, i.e.,

    $$n_{split} = \frac{\text{current particle weight}}{\text{average particle weight}}$$

    *[NOTE: If the ratio is not an integer, the same non-integer technique described in a previous section can be used.*

    The weight correction for each off the split particles is (formally) given by:

    $$w_{curr} = \frac{\text{average particle weight}}{\text{current particle weight}}$$

    Of course, applying this "correction" is just equivalent to just setting each split particle's weight to the window average weight.

4. Recent changes to MCNP allow the weight windows (but NOT cell importances) to be entered for each “mesh cell” in a regular grid that overlays your geometry.

   *[NOTE: This simple addition has revolutionized the use of “hybrid” calculations using both Monte Carlo and discrete ordinates, the latter of which requires a regular grid.]*

5. Unlike the cell importances, the weight-windows values for each region can also be energy-dependent, with weights provided (or automatically calculated) inside energy regions specified by the user.
6. MCNP includes an option called a weight-windows generator that tells MCNP to collect information during a run that can be turned into appropriate weight windows for use in a subsequent run.

## Exercises

4-1. Rerun problem 2-9, using the following (unnormalized) PDFs to select x (did) [NOTE: “Shoulda” is an upside-down “V” from 0 to 2]:

a. $\pi(x) = x,\ 0<x<2$

b. $\pi(x) = x^2,\ 0<x<2$

c. $\pi(x) = x^3,\ 0<x<2$

4-2. Incorporate absorption weighting into the 2-group code used in Chapter 3. Compare results to an unbiased run, including FOM comparison.

4-3. Incorporate forced collision (in the form of not allowing particles to escape) into the 2-group code used in Chapter 3. Compare results to an unbiased run, including FOM comparison.

4-4. Incorporate energy group source biasing into the 2-group code used in Chapter 3. Use the right leakage in group 2 as the tally to optimize. Compare results to an unbiased run, including FOM comparison.

# Chapter 4

4-1. a. $0.18 \pm \frac{0.247}{\sqrt{N}}$

b. $0.18 \pm \frac{0.221}{\sqrt{N}}$

c. $0.18 \pm \frac{0.218}{\sqrt{N}}$