NE 155, S15
April 20/27, 2016

In the last several classes we covered a bunch of basics for MC transport: sampling, scoring, and statistics. At the beginning, we talked a little bit about analog vs. non-analog MC.

(recap: What we have talked about so far is *Analog* Monte Carlo:

- Natural laws are **preserved**
- The game is the "analog" of the physical problem of interest (the history of each particle is simulated exactly)
- No alteration of PDFs
- At collision, particle is killed if absorption
- Particle is born with weight 1
- weight unchanged throughout history
- Score when tallying events is 1

We often, instead, want to do *Non-Analog* Monte Carlo:

- To reduce computation time, the strict analog simulation of particles is abandoned
- Variance Reduction techniques: Absorption suppression, Russian Roulette (history termination), Splitting (history propagation), Forced collisions, Source biasing, Hybrid methods
- Alter PDFs to favor events of interest
- Particle can have different birth weight
- Weight is altered if biased PDF is used
- Particle survives "absorption" and weight is changed
- Splitting and RR can change weight
- Score current weight when tallying

)

The next thing to think about is how to measure success. How do we know if a calculation is "better"?

We use the figure of merit

$$FOM = \frac{1}{R^2 t} ,$$

where R is the relative error and t is the particle tracking time.
What we really want is to reduce both of these.
Why are they related to one another this way? Recall that $S_x \propto \sqrt{\frac{1}{N}}$.
It's clear that without variance reduction techniques to reduce error by a factor of two you need to increase particle count (and hence time) by a factor of four.
With VR we hope to beat out the direct relationship of between R and t by reducing one or both faster than with analog MC alone. FOM measures if we're really winning.

The idea of VR is to track particles that will contribute meaningfully to the desired results and to avoid tracking those that will not while maintaining a fair game.

Categories of VR methods
There are a huge number of VR methods out there. Some are simple, some are complicated. Some are easy to use, others are not.
Misuse of many of these methods can lead to incorrect answers without clear warning that the answers are incorrect.
Here is a list of common types of VR, but we'll go into detail of a very few. There are four main categories (I'm using the MCNP manual as a reference here):

1. Truncation methods: cut off the parts of phase space you don't think you need
    - geometry truncation
    - energy cutoff
2. Population Control methods: directly control the number and weight of particles
    - splitting
    - roulette
    - executed a variety of ways: geometry-based splitting/roulette, energy-based splitting/roulette, weight windows, weight cutoff
3. Modified Sampling methods: play games with the representation of physics to try to get better particle numbers and weights (alter underlying physical reality without biasing results)
    - exponential transform

- survival biasing (implicit capture)
- forced collisions
- source biasing
- neutron-induced photon production biasing

4. Partially Deterministic methods [Hazard!]: black magic; circumvent the normal random-walk process by using deterministic-like techniques. These methods can be combined to give results that look good and are completely wrong.
   - point detectors
   - DXTRAN
   - correlated sampling

Survival Biasing

This is a very common technique. In analog MC, what happens when a particle undergoes a collision?

$$\text{tally } w_i \; ; \qquad w_{i+1} = 0$$

If this particle's history is terminated, is it available to contribute to the answer and provide more data?

To keep it around, we'll just change the particle's weight in a way that preserves physics rather than terminate it.

What is the probability of an absorption during a collision in terms of macroscopic cross sections?

$$P_{abs} = \frac{\Sigma_a}{\Sigma_t}$$

We can use this to change particle weight and keep particles around for longer

$$\text{tally } w_i * \frac{\Sigma_a}{\Sigma_t} \; ; \qquad w_{i+1} = w_i * (1 - \frac{\Sigma_a}{\Sigma_t})$$

The reduction in particle weight at each collision compensates, statistically, for the nonphysical scattering. This maintains a fair game and provides more information per history, though at the cost of each collision being worth less.

Target Weight Map

We can use the notion of weight to conduct VR.

What if we have a weight that is really *low*? : wastes time

What if we have a weight that is really *high*? : increases variance (relative error)
Ideally, we'd like to keep $w_{\min} \leq w \leq w_{\max}$.

When looking for a specific answer, some regions may be more important to the answer than others. We can make a map expressing the relative importances for a given problem (as we've alluded to). As particles move, they will traverse from regions of one importance to another. We can use how they change importance values to change their weight and the number of particles that we're tracking. We will talk about how you use these maps, and then about some ways to make them.

We can use the idea of importance to create the target weights where we'd like particles to exist and associated bounding values. We usually make a map of w_{nom} and set $w_{\min}$ and $w_{\max}$ from there.

Splitting
If particles have too high a weight or are moving into a more important region, we can split them into more particles with lower weights. We preserve the total weight to maintain a fair game:

---

if $w_i > w_{\max}$:

- Split Ratio (SR) = $\frac{w_i}{w_{nom}}$
- get ξ, a random number on [0, 1)
- if $\xi \geq$ (SR - int(SR)):
  - create int(SR) new particles
- else:
  - create int(SR) + 1 new particles
- For all new particles, $w_{i+1} = \frac{w_i}{\#\text{new particles}}$

You're preserving the total weight on average and converting 1 particle with too high a weight to multiple particles with a more useful weight and/or tracking more particles in an important part of the problem. (draw picture of one large particle crossing into another region and becoming two or several smaller particles where size represents weight)

Rouletting
Conversely, if a particle has too low of a weight or is moving into a less important region, we can

either increase its weight or kill it. Again, we preserve the total weight.

---

if $w_i < w_{\min}$:

- Roulette Ratio (RR) = $\frac{w_i}{w_{nom}}$
- get ξ, a random number on [0, 1)
- if $\xi \geq$ RR:
  - kill particle; $w_{i+1} = 0$
- else:
  - $w_{i+1} = w_{nom}$

On average we're killing enough particles to make up for increasing the weight of some particles. We are taking low weight particles that we don't want to waste our time tracking and converting them to useful particles and/or tracking fewer particles in less important parts of the problem. (draw picture of several small particles crossing into another region and becoming one larger particle)

Note that for both of these ideas, w_{nom}, $w_{\min}$, and $w_{\max}$ are what guides when we check to splitting and rouletting as well as the outcome. Therefore, the key is to get weight sets that work well for our problem. What can you imagine the weight sets being dependent on?

Making an Importance/Weight Map

How do we assign importances to a problem? We can do it by hand. E.g. you can assign each geometry component an importance based on what you think should be happening. All splitting and rouletting is based on the importance ratios between geometric cells. Guidance to do this is to keep adjacent cells within a factor of a few of one another and to keep cells only about two mean free paths thick. It is one thing to think about doing that in a 1D slab (and it still may be too hard to do—think about doing that accurately for a six-order-of-magnitude fall off over 30 cm of lead), but choosing these properly in 3D? Not so easy...

We would prefer to have an automated method that does not rely on guessing. The Consistent Adjoint Driven Importance Sampling (CADIS) and Forward Weighted (FW)-CADIS methods do just that. They were developed by John Wagner, Ali Haghighat, and Douglas Peplow at Penn State and ORNL. We'll go through these methods, highlighting conceptual ideas. In particular, we note the application of the adjoint.

Quick Aside: the forward flux describes the particle distribution as a result of system conditions. The adjoint flux describes how important the particle distribution is to the answer in question. We are sort of going to breeze by this, but hopefully this is enough:

$$\begin{aligned} R &= \int dV \int dE\, f(\vec{r}, E)\phi(\vec{r}, E) \quad \text{and} \\ &= \int dV \int dE\, \phi^\dagger(\vec{r}, E) q(\vec{r}, E) \end{aligned}$$

where f is the response function of interest and we've left out angle for simplicity.
We got here by setting $q^\dagger = f$ and using the adjoint identity.
The physical interpretation of the adjoint flux is the influence or importance of a particle in phase space to what we used for the adjoint source: it maps source particles directly into the response. Thus, if we set the response that we are solving for to be the adjoint source, the adjoint flux is the importance map corresponding to that response.

CADIS

These notes are derived from the SCALE manual descriptions.

Note that if we had an accurate representation of $\phi^\dagger$, we would then just have the answer. Solving for $\phi^\dagger$ is just as hard as solving for ϕ.
The idea motivating CADIS is to use an approximate version of $\phi^\dagger$ that was obtained quickly with a deterministic solver to create VR parameters for MC.

We use the adjoint flux to bias the source distribution, set target weights, and choose particle birth weights:

$$\begin{aligned} \hat{q}(\vec{r}, E) &= \frac{1}{R} q(\vec{r}, E)\phi^\dagger(\vec{r}, E) \\ w_{nom} &= \frac{R}{\phi^\dagger(\vec{r}, E)} \\ w_0 \equiv \frac{q(\vec{r}, E)}{\hat{q}(\vec{r}, E)} &= \frac{R}{\phi^\dagger(\vec{r}, E)}\,, \end{aligned}$$

where w_0 is birth weight. Note that we need to set the birth weight this way such that a fair game is preserved given the biased source sampling.
Further, notice that the birth weight exactly matches the target weight. This means time is not wasted splitting/rouletting immediately upon birth.
The consistency among these items is also the source of the method's name.

Multiple tallies

What if we want multiple tallies? For these problems, the user must accept a total simulation time that is controlled by the tally with the slowest convergence and simulation results where the tallies have a wide range of relative uncertainties.
The obvious way around this problem is to create a separate problem for each tally and use CADIS to optimize each. Each simulation can then be run until the tally reaches the level of acceptable uncertainty. For more than a few tallies, this approach becomes complicated and time-consuming for the user.
For mesh tallies, this approach is not reasonable.

Another approach to treat several tallies, if they are in close proximity to each other, or a mesh tally covering a small portion of the physical problem, is to use the CADIS methodology with the adjoint source near the middle of the tallies to be optimized.
Since particles in the forward Monte Carlo simulation are optimized to reach the location of the adjoint source, all the tallies surrounding that adjoint source should converge quickly.
The drawback to this approach is the difficult question of "how close". If the tallies are too far apart, certain energies or regions that are needed for one tally may be of low importance for getting particles to the central adjoint source. This may under-predict the flux or dose at some of the tally sites.

For several tallies that are far from each other, multiple adjoint sources could be used. In the forward Monte Carlo, particles would be drawn to one of those adjoint sources.
The difficulty with this approach is that typically the tally that is closest to the true physical source will converge faster than the other tallies.
Finding the correct relative source strengths so that all of the tallies converged to the same relative uncertainty in one simulation becomes an iterative process for the user.

FW-CADIS
If we want the answer everywhere or in many locations (like in shielding or some reactor and security calculations), we need a different approach: forward weighting.
There are two ways to think about why this works:

1) In order to converge several tallies to the same relative uncertainty in the one simulation, the adjoint source corresponding to each of those tallies needs to be weighted inversely by the expected tally value.
With mesh tallies, instead of using a uniform adjoint source strength over the entire mesh tally volume, each voxel of the adjoint source should be weighted inversely by the expected forward tally value for that voxel.
Areas of low flux or low dose rate would have more adjoint source strength than areas of high flux

or high dose rate.

First, a forward, coarse deterministic calculation is done to estimate the expected tally results.
A total adjoint source is constructed, where the adjoint source corresponding to each tally is weighted inversely by those forward tally estimates.

2) (from Wagner) To get uniformly-low statistical uncertainty, it has been proposed that the distribution of MC particles should be uniform throughout the system.
Although this is not a 'physical' response, it does intuitively represent a desirable objective for obtaining uniform uncertainty.
Using adjoint transport theory, we can define the adjoint source such that the result is a an importance function targeting uniform particle distribution.

Either way you want to think about it, we do the same thing.
We do a course deterministic calculation to get the forward flux and use it in constructing the adjoint source:

$$\begin{aligned}
\text{energy- and space-dependent flux, } \phi(\vec{r},E): &\quad q^\dagger(\vec{r},E) = \frac{1}{\phi(\vec{r},E)} \\
\text{space-dependent total flux, } \int dE\ \phi(\vec{r},E): &\quad q^\dagger(\vec{r},E) = \frac{1}{\int dE\ \phi(\vec{r},E)} \\
\text{space-dependent total dose rate, } \int dE\ \phi(\vec{r},E)\Sigma_d(\vec{r},E): &\quad q^\dagger(\vec{r},E) = \frac{\Sigma_d(\vec{r},E)}{\int dE\ \phi(\vec{r},E)\Sigma_d(\vec{r},E)}
\end{aligned}$$

You can see the inverse weighting discussed in explanation (1), or you can think of how this impacts the approximate response (in this case defined as $\langle \phi, q^\dagger \rangle$) for explanation (2):

$$R(\vec{r},E) = \int dV \int dE\ \phi(\vec{r},E)\frac{1}{\phi(\vec{r},E)} \approx 1\ ,$$

where the approximately equals is because the ϕ values are rough guesses and can come from different calculations.

Again, if we had the global solution, wouldn't we be finished? The key to FW-CADIS is to remember that both the forward and adjoint calculations are coarse and hence very fast. It therefore takes a small amount of time to create a very effective set of values used for source biasing, birth weight setting, and weight window creation - where the weight windows are used for splitting and rouletting.

Mention that here are some common **general purpose MC codes in nuclear:**

- **MCNP**: developed at LANL, distributed via RSICC, http://rsicc.ornl.gov
- **Geant4**: developed by a large collaboration in the HEP community, http://geant4.web.cern.ch/geant4/
- **EGSnrc**: developed at NRC (Canada), http://www.irs.inms.nrc.ca/EGSnrc/EGSnrc.html
- **SERPENT**: Developed by Dr. Jaakko Leppanen, VTT, Finland, http://montecarlo.vtt.fi/
- **Shift**: developed at ORNL, distributed via RSICC, http://rsicc.ornl.gov