LA-UR-05-4983

Title: FUNDAMENTALS OF MONTE CARLO
PARTICLE TRANSPORT

Author(s): FORREST B. BROWN

Submitted to: Lecture notes for Monte Carlo course

Los Alamos
NATIONAL LABORATORY

Fundamentals of Monte Carlo Particle Transport

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Abstract

Fundamentals of Monte Carlo Particle Transport

Solving particle transport problems with the Monte Carlo method is simple - just simulate the particle behavior. The devil is in the details, however. This course provides a balanced approach to the theory and practice of Monte Carlo simulation codes, with lectures on transport, random number generation, random sampling, computational geometry, collision physics, tallies, statistics, eigenvalue calculations, variance reduction, and parallel algorithms. This is not a course in how to use MCNP or any other code, but rather provides in-depth coverage of the fundamental methods used in all modern Monte Carlo particle transport codes. The course content is suitable for beginners and code users, and includes much advanced material of interest to code developers.   (10 lectures, 2 hrs each)

The instructor is Forrest B. Brown from the X-5 Monte Carlo team. He has 25 years experience in developing production Monte Carlo codes at DOE laboratories and over 200 technical publications on Monte Carlo methods and high-performance computing. He is the author of the RACER code used by the DOE Naval Reactors labs for reactor design, developed a modern parallel version of VIM at ANL, and is a lead developer for MCNP5, MCNP6, and other Monte Carlo codes at LANL.

Topics

1. **Introduction**
   - **Monte Carlo & the Transport Equation**
   - **Monte Carlo & Simulation**

2. **Random Number Generation**
3. **Random Sampling**

4. **Computational Geometry**
5. **Collision Physics**
6. **Tallies & Statistics**

7. **Eigenvalue Calculations – Part I**
8. **Eigenvalue Calculations – Part II**
9. **Variance Reduction**

10. **Parallel Monte Carlo**

11. **References**

Introduction

- **Von Neumann invented scientific computing in the 1940s**
  - **Stored programs, "software"**
  - **Algorithms & flowcharts**
  - **Assisted with hardware design as well**
  - **"Ordinary" computers today are called "Von Neumann machines"**

- **Von Neumann invented Monte Carlo methods for particle transport in the 1940s   (with Ulam, Fermi, Metropolis, & others at LANL)**
  - **Highly accurate – no essential approximations**
  - **Expensive – typically the "method of last resort"**
  - **Monte Carlo codes for particle transport have been proven to work effectively on all types of computer architectures:**

      SIMD, MIMD, vector, parallel, supercomputers,
   workstations, PCs, Linux clusters, clusters of anything,…

Introduction

- **Two basic ways to approach the use of Monte Carlo methods for solving the transport equation:**
  - **Mathematical technique for numerical integration**
  - **Computer simulation of a physical process**

    ⇒ **Each is "correct"**

  - Mathematical approach is useful for:

    Importance sampling, convergence, variance reduction, random sampling techniques, eigenvalue calculation schemes, ...

  - Simulation approach is useful for:

    collision physics, tracking, tallying, ...

- **Monte Carlo methods solve integral problems, so consider the <u>integral</u> form of the Boltzmann equation**

- **Most theory on Monte Carlo deals with fixed-source problems. Eigenvalue problems are needed for criticality and reactor physics calculations.**

Introduction

Simple Monte Carlo Example

Evaluate $\quad G = \int_0^1 g(x)dx, \quad$ with $\quad g(x) = \sqrt{1-x^2}$

g(x)
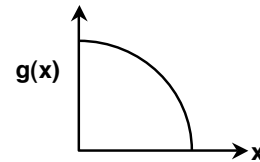
- **Mathematical approach:**

  For $k = 1, ..., N$: $\quad$ choose $\quad \hat{x}_k$ randomly in (0,1)

  $$G = (1-0)\cdot[\text{average value of } g(x)] \approx \tfrac{1}{N}\cdot\sum_{k=1}^{N} g(\hat{x}_k) = \tfrac{1}{N}\cdot\sum_{k=1}^{N} \sqrt{1-x_k^2}$$

- **Simulation approach:**

  "darts game"

  For $k = 1, ..., N$: $\quad$ choose $\quad \hat{x}_k, \hat{y}_k$ randomly in (0,1),

  $\quad$ if $\quad \hat{x}_k^2 + \hat{y}_k^2 \leq 1$, tally a "hit"

  + miss

  y

  + hit
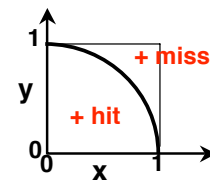
  $$G = [\text{area under curve}] \quad \approx \quad (1\cdot1)\cdot\frac{\text{number of hits}}{N}$$

Monte Carlo is often the method-of-choice for applications with integration over many dimensions

Examples: high-energy physics, particle transport, financial analysis, risk analysis, process engineering, ...

Evaluate

$$G = \int_{a_1}^{b_1}\int_{a_2}^{b_2} \cdots \int_{a_M}^{b_M} g(r_1, r_2, ..., r_M)\, dr_1 dr_2 ... dr_M$$

where $r_1, r_2, ..., r_M$ are all independent variables

For k = 1, ..., N:

For m = 1, ..., M: choose $R_m^{(k)}$ randomly in (a_m, b_m)

$$G \sim (b_1 - a_1)\cdot ... \cdot(b_M - a_M) \cdot \frac{1}{N}\sum_{k=1}^{N} g\left(R_1^{(k)}, R_2^{(k)}, ..., R_M^{(k)}\right)$$

1 -7

Introduction – Probability Density Functions

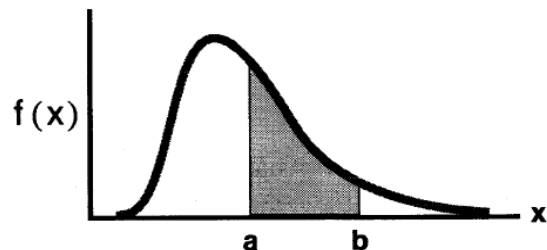- **Continuous Probability Density**

  f(x) = probability density function (PDF)

  $f(x) \geq 0$

  Probability$\{a \leq x \leq b\} = \int_a^b f(x)dx$
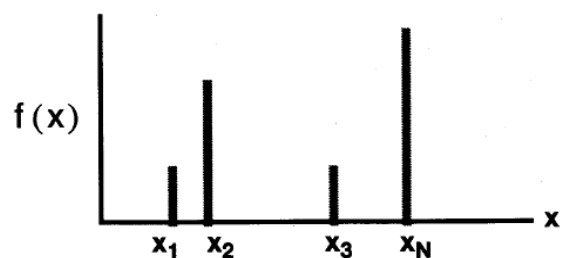
  Normalization: $\int_{-\infty}^{\infty} f(x)dx = 1$

- **Discrete Probability Density**

  $\{f_k\}, \ k = 1,...,N,$ where $f_k = f(x_k)$

  $f_k \geq 0$

  Probability$\{x = x_k\} = f_k$

  Normalization: $\sum_{k=1}^{N} f_k = 1$

1 -8

- **Mean, Average, Expected Value**

$$\bar{x} = \mu = <x> = E[x]$$

$$\mu = \int_{-\infty}^{+\infty} x\,f(x)\,dx \quad \text{[continuous]} \qquad \mu = \sum_{k=1}^{N} x_k f_k \quad \text{[discrete]}$$

- **Variance**

$$\text{var}(x) = \overline{(x-\mu)^2} = \sigma^2 = <(x-\mu)^2> = E\left[(x-\mu)^2\right]$$

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)\,dx \qquad \sigma^2 = \sum_{k=1}^{N} (x_k-\mu)^2 f_k$$

- **Standard Deviation**

$$\sigma = \sqrt{\sigma^2}$$

- **Functions of a Random Variable**

Consider $g(x)$, where x is a random variable with density $f(x)$

$$E[g(x)] = \int_{-\infty}^{+\infty} g(x)\,f(x)\,dx \qquad E[g(x)] = \sum_{k=1}^{N} g_k f_k$$

The key to Monte Carlo methods is the notion of *random sampling*.

- The problem can be stated this way:

  Given a probability density, f(x), produce a sequence of $\hat{X}$'s.
  The $\hat{X}$'s should be distributed in the same manner as f(x).

f (x)

x →
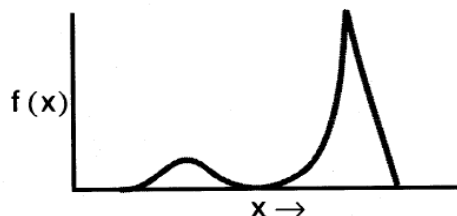
- The use of random sampling distinguishes Monte Carlo from other methods

- When Monte Carlo is used to solve the integral Boltzmann transport equation:
  - Random sampling models the outcome of physical events (e.g., neutron collisions, fission process, sources, …..)
  - Computational geometry models the arrangement of materials

Monte Carlo & Transport Equation

Boltzmann transport equation — time-independent, linear

$$\Psi(\mathbf{r}, \mathbf{v}) = \int \left[\int \Psi(\mathbf{r}', \mathbf{v}')\, C(\mathbf{v}' \to \mathbf{v}, \mathbf{r}')\, d\mathbf{v}' + Q(\mathbf{r}', \mathbf{v}) \right] T(\mathbf{r}' \to \mathbf{r}, \mathbf{v})\, d\mathbf{r}'$$

where

- $\Psi(\mathbf{r}, \mathbf{v})$ =   particle collision density

- $Q(\mathbf{r}', \mathbf{v})$ =   source term

- $C(\mathbf{v}' \to \mathbf{v}, \mathbf{r}')$ =   collision kernel,   change **velocity**   at fixed position

- $T(\mathbf{r}' \to \mathbf{r}, \mathbf{v})$ =   transport kernel,   change **position**   at fixed velocity

- **Angular Flux** $\psi(\mathbf{r}, \mathbf{v}) = \dfrac{\Psi(\mathbf{r}, \mathbf{v})}{\Sigma(\mathbf{r}, |\mathbf{v}|)}$

- **Scalar Flux** $\Phi(\mathbf{r}, |\mathbf{v}|) = \displaystyle\int_{\vec{\Omega}} \dfrac{\Psi(\mathbf{r}, \mathbf{v})}{\Sigma(\mathbf{r}, |\mathbf{v}|)}\, d\vec{\Omega},$ $\mathbf{v} = |\mathbf{v}|\vec{\Omega}$

Source term for the Boltzmann equation:

$$Q(\mathbf{r}, \mathbf{v}) = \begin{cases} S(\mathbf{r}, \mathbf{v}) & \Leftarrow \text{Fixed Source} \\[2em] S(\mathbf{r}, \mathbf{v}) + \int \Psi(\mathbf{r}, \mathbf{v'}) F(\mathbf{v'} \to \mathbf{v}, \mathbf{r}) d\mathbf{v'} & \Leftarrow \text{Fixed Source + Fission} \\[2em] \frac{1}{K} \int \Psi(\mathbf{r}, \mathbf{v'}) F(\mathbf{v'} \to \mathbf{v}, \mathbf{r}) d\mathbf{v'} & \Leftarrow \text{Eigenvalue} \end{cases}$$

where

- $S(\mathbf{r}, \mathbf{v})$ =   fixed source

- $F(\mathbf{v'} \to \mathbf{v}, \mathbf{r})$ =   creation operator (due to fission), particle at $(\mathbf{r}, \mathbf{v'})$ creates particle at $(\mathbf{r}, \mathbf{v})$

- K =   eigenvalue

$$\Psi(r,v) = \int \left[\int \Psi(r',v') \cdot C(v' \to v,r') dv' + Q(r',v) \right] \cdot T(r' \to r,v) dr'$$

- **Assumptions**
  - **Static, homogeneous medium**
  - **Time-independent**
  - **Markovian – next event depends only on current (r,v,E), not on previous events**
  - **Particles do not interact with each other**
  - **Neglect relativistic effects**
  - **No long-range forces (particles fly in straight lines between events)**
  - **Material properties are not affected by particle reactions**
  - **Etc., etc.**

  ⇒ Can use the superposition principle

Basis for the Monte Carlo Solution Method

Let $\quad p = (\vec{r}, \vec{v}) \qquad$ and $\qquad R(p' \to p) = C(v' \to v, r') \cdot T(r' \to r, v)$

Expand Ψ into components having $0, 1, 2, ..., k$ collisions

$$\Psi(p) = \sum_{k=0}^{\infty} \Psi_k(p), \qquad \text{with} \quad \Psi_0(p) = \int Q(r', v) T(r' \to r, v) dr'$$

By definition,

$$\Psi_k(p) = \int \Psi_{k-1}(p') \cdot R(p' \to p) dp'$$

Note that collision k depends only on the results of collision $k-1$,
and not on any prior collisions $k-2, k-3, ...$

Histories

- **After repeated substitution for Ψ_k**

$$\Psi_k(p) = \int \Psi_{k-1}(p') \cdot R(p' \to p) dp'$$
$$= \int ... \int \Psi_0(p_0) \cdot R(p_0 \to p_1) \cdot R(p_1 \to p_2) ... R(p_{k-1} \to p) dp_0 ... dp_{k-1}$$

- **A "history" is a sequence of states $\quad (p_0, p_1, p_2, p_3, ...)$**
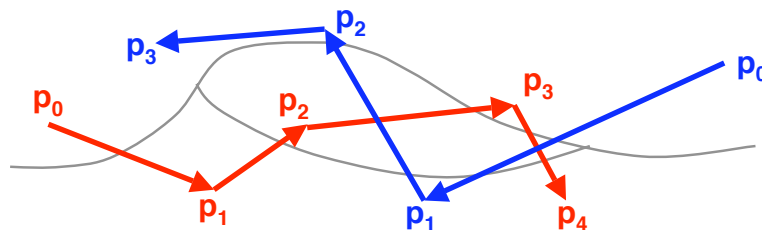
History 1
History 2

- **For estimates in a given region, <u>tally</u> the occurrences for <u>each collision of each "history"</u> within a region**

$$\Psi_k(p) = \int ... \int \Psi_0(p_0) \cdot R(p_0 \rightarrow p_1) \cdot R(p_1 \rightarrow p_2)...R(p_{k-1} \rightarrow p)dp_0...dp_{k-1}$$

Monte Carlo approach:

- **Generate a sequence of states (p_0, p_1, p_2, p_3, …..)   [i.e., a history] by:**
  - **Randomly sample from PDF for source:** $\Psi_0(p_0)$
  - **Randomly sample from PDF for k^{th} transition:** $R(p_{k-1} \rightarrow p_k)$

- **Generate estimates of results by averaging over states for M histories:**

$$A = \int A(p) \cdot \Psi(p)dp \approx \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\sum_{k=1}^{\infty} A(p_{k,m}) \right)$$

Monte Carlo
&
Simulation

"Simulation is better than reality"
Richard W. Hamming, 1991

Simulation approach to particle transport:

Faithfully simulate the history of a single particle from birth to death.

- **Random-walk for a single particle**
  - **Model collisions using physics equations & cross-section data**
  - **Model free-flight between collisions using computational geometry**
  - **Tally the occurrences of events in each region**
  - **Save any secondary particles, analyze them later**

Track through geometry,	Collision physics analysis,	Secondary Particles
- select collision site randomly	- Select new E,Ω randomly	
- tallies	- tallies	
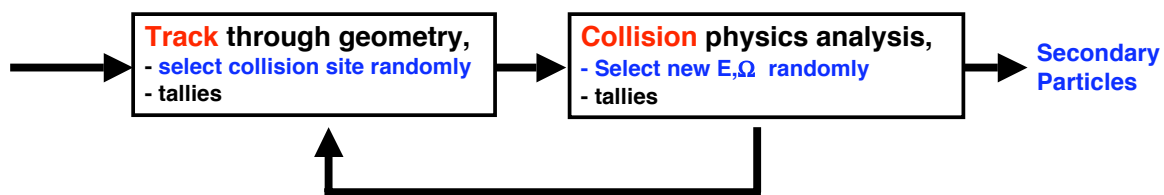
- **A "history" is the simulation of the original particle & all of its progeny**

Source
- select r,E,Ω → Random Walk → Random Walk → Random Walk / Random Walk / Random Walk → Random Walk
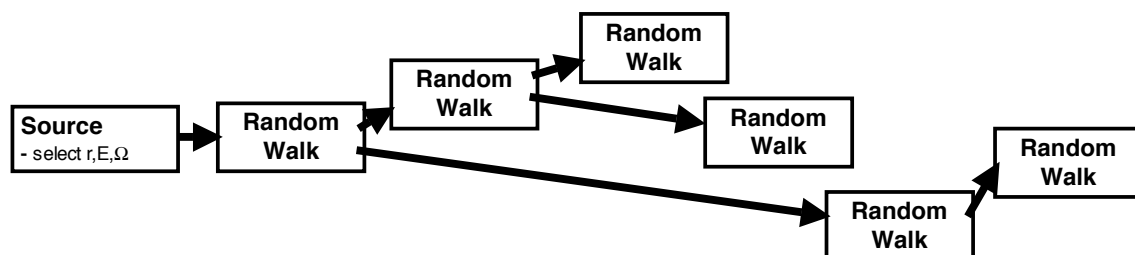
- **Repeat for many histories, accumulating tallies**

- **Fundamental rule: Think like a particle !**

Source

- Random sampling

  E, Ω — analytic, discrete, or piecewise-tabulated PDF's

- Computational geometry

  r — sample from region in 3-D space, or from discrete PDF

Tracking

- Random sampling

  $d_{collide}$ — distance to collision, from mfp & exponential PDF

- Computational geometry

  d_{geom} — distance-to-boundary, ray-tracing, next-region, ...

Collisions

- Random sampling

  E', Ω' — analytic, discrete, or piecewise-tabulated PDF's

- Physics

  $\Sigma, f(\mu)$ — cross-section data, angular PDF's, kinematics, ...

Tallies

- Statistics

Variance Reduction

- Random sampling

Single particle

- random-walk for particle history
- simulate events, from birth to death
- tally events of interest

> Select **source** r, Ω, E *randomly*
>
> **Track** through geometry, select collision site r *randomly*
>
> **Collision** physics analysis, select new Ω, E *randomly*
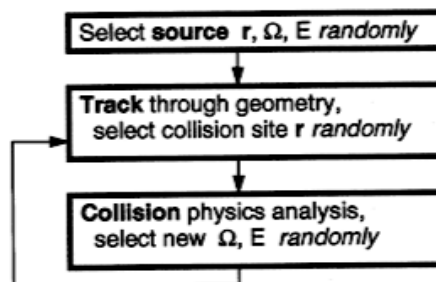
Batch of histories ("generation")

- random-walk for many particle histories
- tally the aggregate behavior

Overall

- timesteps
  - — geometry changes
  - — material changes
    - fuel depletion
    - burnable absorbers
    - control rods

> ✴ Loop over *timesteps*
> .   ✴ Loop over *batches*
> .   .   ✴ Loop over *histories*
> .   .   .   → *random walk*
> .   .   ...
> .   .   → update Keff & reaction rates
> .   ...
> .   → compute statistics
> .   → update number densities, ...
> ...
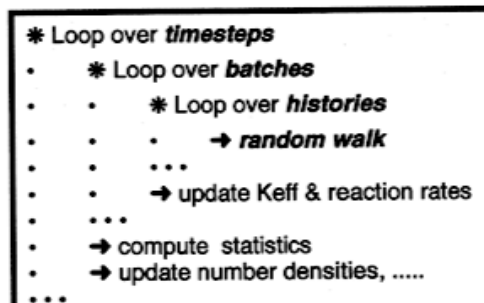
Random Number Generation

"Randomness is a negative property; it is the absence of any pattern."
Richard W. Hamming, 1991

"…random numbers should not be generated by a method chosen at random."
Donald Knuth, 1981

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

2 -1

Random Number Generators

Random Number Generators (RNGs)

- **Numbers are not random; a sequence of numbers can be.**
- **Truly random sequences are generally not desired on a computer.**
- **Pseudo-random sequences:**
    - **Repeatable (deterministic)**
    - **Pass statistical tests for randomness**
- **RNG**
    - **Function which generates a sequence of numbers which appear to have been randomly sampled from a uniform distribution on (0,1)**
    - **Probability density function for f(x)**
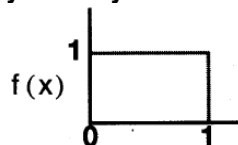
    - **Typical usage in codes: r = ranf()**
    - **Also called "pseudo-random number generators"**
- **All other random sampling is performed using this basic RNG**

2 -2

Most production-level Monte Carlo codes for particle transport use <u>linear congruential random number generators</u>:

$$S_{i+1} = S_i \cdot g + c \quad \text{mod } 2^m$$

S_i = seed, g = multiplier, c = adder, 2^m = modulus

- Robust, over 40 years of heavy-duty use

- Simple, fast

- Theory is well-understood (e.g., DE Knuth, Vol. 2, 177 pages)

- Not the "best" generators, but good enough – RN's are used in unpredictable ways during particle simulation

- To achieve reproducibility of Monte Carlo calculations, despite vectorization or varying numbers of parallel processors, there must be a fast, direct method for **skipping ahead** (or back) in the random sequence

• due to Lehmer, 1949
• most common method, excellent (when not abused)

$s_0 \leftarrow$ initial value

• Method:

$r_k \leftarrow s_k / p$

$s_{k+1} \leftarrow [g \cdot s_k + c] \text{ mod } p$

where

s_k, g, c, p = integers, r_k = real

s_k = seed
g = generator, or multiplier
c = increment
p = modulus
r_k = psuedo-random number, $0 \le r_k \le 1$

• "mod p" $\Rightarrow$ "remainder after division by p", absolutely no roundoff is permitted
• Multiplicative: c = 0
• Mixed: c > 0

Example #1: $s_{k+1} \leftarrow [g \cdot s_k + c] \bmod p$

with $g = 47$, $c = 1$, $s_0 = 1$, $p = 100$

```
s( 0)  =   1
s( 1)  =  (47x1  + 1) mod 100  =     48 mod 100  =   48
s( 2)  =  (47x48 + 1) mod 100  =   2257 mod 100  =   57
s( 3)  =  (47x57 + 1) mod 100  =   2680 mod 100  =   80
s( 4)  =  (47x80 + 1) mod 100  =   3761 mod 100  =   61
s( 5)  =  (47x61 + 1) mod 100  =   2868 mod 100  =   68
s( 6)  =  (47x68 + 1) mod 100  =   3197 mod 100  =   97
s( 7)  =  (47x97 + 1) mod 100  =   4560 mod 100  =   60
s( 8)  =  (47x60 + 1) mod 100  =   2821 mod 100  =   21
s( 9)  =  (47x21 + 1) mod 100  =    988 mod 100  =   88
s(10)  =  (47x88 + 1) mod 100  =   4137 mod 100  =   37
s(11)  =  (47x37 + 1) mod 100  =   1740 mod 100  =   40
s(12)  =  (47x40 + 1) mod 100  =   1881 mod 100  =   81
s(13)  =  (47x81 + 1) mod 100  =   3808 mod 100  =    8
s(14)  =  (47x8  + 1) mod 100  =    377 mod 100  =   77
s(15)  =  (47x77 + 1) mod 100  =   3620 mod 100  =   20
s(16)  =  (47x20 + 1) mod 100  =    941 mod 100  =   41
s(17)  =  (47x41 + 1) mod 100  =   1928 mod 100  =   28
s(18)  =  (47x28 + 1) mod 100  =   1317 mod 100  =   17
s(19)  =  (47x17 + 1) mod 100  =    800 mod 100  =    0
s(20)  =  (47x0  + 1) mod 100  =      1 mod 100  =    1
s(21)  =  (47x1  + 1) mod 100  =     48 mod 100  =   48
s(22)  =  (47x48 + 1) mod 100  =   2257 mod 100  =   57
  etc.
```

Example #2: $s_{k+1} \leftarrow [g \cdot s_k + c] \bmod p$

with $g = 5$, $c = 1$, $s_0 = 1$, $p = 100$

```
s( 0)  =   1
s( 1)  =  (5x1  + 1) mod 100  =      6 mod 100  =    6
s( 2)  =  (5x6  + 1) mod 100  =     31 mod 100  =   31
s( 3)  =  (5x31 + 1) mod 100  =    156 mod 100  =   56
s( 4)  =  (5x56 + 1) mod 100  =    281 mod 100  =   81
s( 5)  =  (5x81 + 1) mod 100  =    406 mod 100  =    6
s( 6)  =  (5x6  + 1) mod 100  =     31 mod 100  =   31
  etc.
```

Example #3: $s_{k+1} \leftarrow [g \cdot s_k + c] \bmod p$

with $g = 5$, $c = 0$, $s_0 = 1$, $p = 100$

```
s( 0)  =   1
s( 1)  =  (5x1 ) mod 100  =      5 mod 100  =    5
s( 2)  =  (5x5 ) mod 100  =     25 mod 100  =   25
s( 3)  =  (5x25) mod 100  =    125 mod 100  =   25
s( 4)  =  (5x25) mod 100  =    125 mod 100  =   25
  etc.
```

Selecting Parameters for Linear Congruential RNGs

$$s_{k+1} \leftarrow [\, g \cdot s_k + c \,] \bmod p$$

- Modulus (p):

  — **choose $p = 2^N$**
  — simplifies "mod p"— discard all but the N least significant bits
  — simplifies division by p— shift the "point" left by N bits
  — N should be as large as possible, N > 35 is best.
  — Usually, choose N to be number of bits in largest positive integer.

- Generator (g), Initial Seed (s_0), & Increment (c) :

  — choose g & c to maximize the period
  — large g is best to reduce serial correlation
  — obviously, g=1 or g=0 are bad

  — **For c = 0 (multiplicative PRNG):**

     choosing   (1) g mod 8 = 3 or 5
                (2) s_0 = odd

     results in: period = 2^{N-2},   the maximum possible period.

  — **For c > 0 (mixed PRNG):**

     choosing   (1) c relatively prime to p
                (2) (g-1) to be a multiple of every prime factor of p
                (3) (g-1) to be a multiple of 4 if p is a multiple of 4

     results in: period = 2^N,   the maximum possible period.

2 -7

Typical Linear Congruential RNGs

Multiplicative congruential method — Lehmer

$$S_{i+1} = g \cdot S_i + c \quad \bmod 2^m, \qquad\qquad 0 < S_i < 2^m$$

$$\xi_i = S_i / 2^m, \qquad\qquad 0 < \xi < 1$$

Typical parameters

		2^m	period	g	c
RACER	(KAPL)	2^{47}	2^{45}	84,000,335,758,957	0
RCP	(BAPL)	2^{48}	2^{48}	2^9+1 59,482,192,516,946	
MORSE	(ORNL)	2^{47}	2^{45}	5^{15}	0
MCNP	(LANL)	2^{48}	2^{46}	5^{19}	0
VIM	(ANL)	2^{48}	2^{46}	5^{19}	0
RANF	(CRAY)	2^{48}	2^{46}	44,485,709,377,909	0
—	(G. Marsaglia)	2^{32}	2^{32}	69069	1
MCNP5	**(LANL)**	2^{63}	2^{63}	(varies)	1

2 -8

Aside ...

For the multiplicative congruential method,
why is the period limited to a maximum of 2^{N-2} ??

$$s_{k+1} \leftarrow g \cdot s_k \mod p, \qquad s_0 \text{ odd}, \quad g \mod 8 = 3 \text{ or } 5$$

- All s_k's are odd, g is odd

$\Rightarrow \quad g \cdot s_k$ will always be odd,   reduces period by a factor of two.

- For $g \mod 8 = 3$, trailing bits of g are (...011)

or
$$g \cdot s_k = (...011) \cdot (....11) = (...11)$$
$$g \cdot s_k = (...011) \cdot (....01) = (...01)$$

$\Rightarrow$ next-to-last bit of s_k will not change,   reduces period by a factor of two.

- For $g \mod 8 = 5$, trailing bits of g are (...101)

or
$$g \cdot s_k = (...101) \cdot (...1x1) = (...1x1)$$
$$g \cdot s_k = (...101) \cdot (...0x1) = (...0x1)$$

$\Rightarrow$ third-to-last bit of s_k will not change,   reduces period by a factor of two.

Example -- CYBER-205 RANF

$$s_{k+1} \leftarrow [\, g \cdot s_k + c \,] \mod p$$

FORTRAN

```
common /q8ranfc/ seed

r = ranf()
```

META

```
LOD    s_descr, s        *load the seed

EX     g, 84000335758957 *generator

EX     e, 65489          *exponent, 2**-47

MPYL   g, s, s           *mult, keep last 47 bits

STO    s_descr, s        *store new seed

PACK   e, s, r           *insert exponent

ADDN   r,  , r           *normalized result
```

Note: (1) $0 < r < 1$
        (2) scalar timing ~320 ns / prn
        (3) to vectorize — "unroll" or "replicate", vector timing ~30 ns / rn

How long will the PRNs last ?

time to generate ALL 2^{45} RNs

Sharp EL-515s		1 M yr
CYBER-205,	scalar	4 mos
CRAY-1,	vector	15 days
CYBER-205,	2-pipe vector	12 days
CYBER-205,	4-pipe vector	6 days
CRAY-XMP/48,	vector x 4	3 days
CRAY-2,	vector x 4	30 hr
ETA-10,	vector x 8	13 hr
cray-c90,	vector x 16	4 hr

MCNP5 Random Number Generator

```
Module  mcnp_random
   ! Multiplier,  adder,  mask (to get lower bits)
   integer(I8), PRIVATE, SAVE ::           RN_MULT,  RN_ADD,  RN_MASK
   ! Factor for normalization to (0,1)
   real(R8),   PRIVATE, SAVE ::           RN_NORM

   ! Private data for a single history
   integer(I8), PRIVATE ::                    RN_SEED,  RN_COUNT,  RN_NPS
   common              /RN_THREAD/       RN_SEED,  RN_COUNT,  RN_NPS
   !$OMP THREADPRIVATE (  /RN_THREAD/  )

 CONTAINS

   function rang( )
        ! MCNP5 random number generator
        implicit none
        real(R8) ::  rang

        RN_SEED  = iand( RN_MULT*RN_SEED,  RN_MASK )
        RN_SEED  = iand( RN_SEED+RN_ADD,  RN_MASK )
        rang        = RN_SEED * RN_NORM
        RN_COUNT = RN_COUNT + 1
        return
   end function rang
```

2 -11

MCNP5 Random Number Generator – Usage

```
Program mcnp5
 . . . . .
 ! Initialize RN parameters for problem
 call RN_init_problem(   new_seed= ProblemSeed  )
 . . . . .

 do nps = 1, number_of_histories

        ! Analyze one particle history
        call RN_init_particle( nps )

        . . . . .
        if( rang() > xs ) . . .
        . . . . .

        ! Terminate history
        call RN_update_stats
        . . . . .
```

2 -12

Other PRNGs

- Middle-square method: s_{k+1} = middle digits of s_k^2

- Quadratic-congruential: s_{k+1} = $[\,a{\cdot}s_k^2 + bs_k + c\,]$ mod p

- Modified Middle-square: s_{k+1} = $[\,s_k{\cdot}(s_k{+}1)\,]$ mod p

- Additive: s_{k+1} = $[\,s_k + s_{k-i}\,]$ mod p

- Additive (or Shift): s_k = $[\,s_{k-j} + s_{k-i}\,]$ mod p

- Generalized Additive (or Shift): s_k = $[\,a_1 s_{k-1} + a_2 s_{k-2} \;....\; a_i s_{k-i}\,]$ mod p

- Quasi-random sequences ....

- etc., etc., ...

Testing PRNGs

- See Knuth, Vol. 2, pp. 38-113

- Empirical Tests:

Chi-square test	Kolmogorov-Smirnov test	Frequency test
Serial pair, triplet, ..., distributions	Gap test	Poker tests
Coupon Collector test	Run test	Maximum-of-t test
Collision test	Serial Correlation coefficients	
etc.		

- Theoretical Tests

Spectral test	Serial Correlation (global)	etc.


Reproducibility of a Particle History

- use separate, distinct random sequence for each particle

- starting seeds for separate particles are separated by "stride"
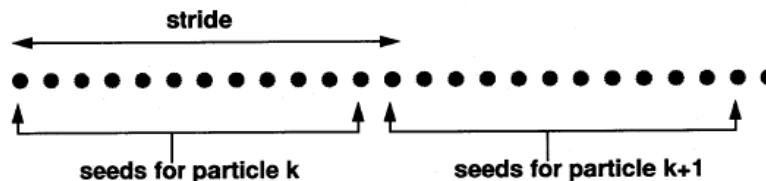
- stride should be large enough to prevent overlap (for most histories)

  — 1000 is common for reactor analysis problems

     - splitting & variance reduction not needed for in-core physics
     - reduces total random number usage

  — 4,297 is the "old" default for MCNP & VIM

  — 152,917 is the default for MCNP & VIM

     - prepared for lots of splitting & variance reduction
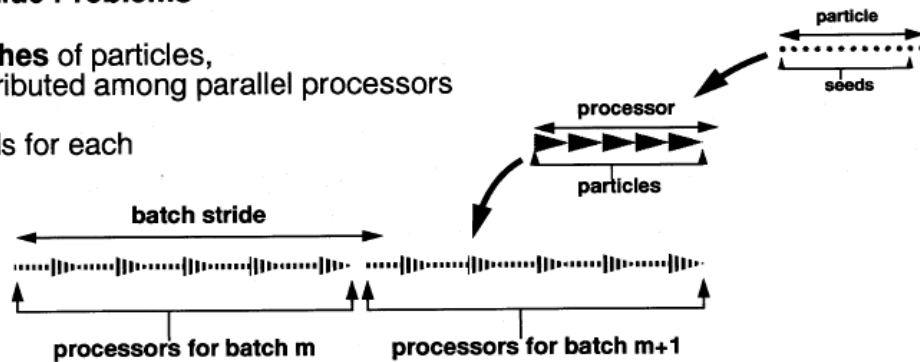   - potential for lots of secondary particles

Random Number Generators – Reproducibility

Parallel processing

- take "super-stride" in random sequence for particles on each processor
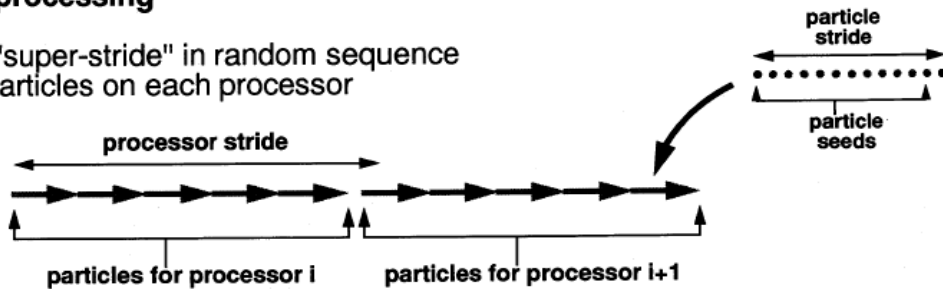
Eigenvalue Problems

- **batches** of particles, distributed among parallel processors
- seeds for each

Random Number Generators – Skip Ahead

To skip ahead k steps in the random sequence,   [initial seed] → [k^{th} seed]

$$S_k = g{\cdot}S_{k-1} + c \;\; \text{mod } 2^m$$

$$= g{\cdot}(g{\cdot}S_{k-2}+c) + c \;\; \text{mod } 2^m$$

$$= g(..... g(g(gS_0+c)+c)+c) .....) + c \;\;\; \text{mod } 2^m$$

$$= g^k \cdot S_0 \;\; + \;\; c \cdot (g^{k-1}+ g^{k-2} + ... + g + 1) \;\;\;\; \text{mod } 2^m$$

$$= g^k \cdot S_0 \;\; + \;\; c \cdot (g^k-1)/(g-1) \;\;\; \text{mod } 2^m$$

- Periodic sequence:
  negative skip k_n equivalent to positive skip **(period - k_n)**

- Can skip from any seed directly to any other:
  initial seed ➜ i^{th} seed for j^{th} particle on m^{th} processor in n^{th} batch
  particle i ➜ particle j
  batch i ➜ batch j

- All arithmetic must be performed mod 2^m, without truncation or roundoff

$$S_k = G(k){\cdot}S_0 + \;\; C(k) \;\;\;\; \text{mod } 2^m$$

Define $\quad G(k) = g^k \bmod 2^m$

$\quad\quad m \;\;= 32 \text{ or } 48 \text{ (typical)}, \quad\quad$ based on the size of a computer word

$\quad\quad -2^m < k < +2^m, \quad\quad$ based on desired "stride"

Denote the jth bit of k by $k_{[j]}$, so that

$$k = 2^{m-1} k_{[m-1]} + 2^{m-2} k_{[m-2]} + \ldots + 2^1 k_{[1]} + 2^0 k_{[0]}$$

Substituting into $G(k)$ yields

$$G(k) = g^k \bmod 2^m = g^{\sum_{j=0}^{m-1} k_{[j]} 2^j} \bmod 2^m$$

$$= \prod_{j=0}^{m-1} \left(g^{2^j}\right)^{k_{[j]}} \bmod 2^m$$

Efficient algorithms for evaluating $G(k)$ can be formulated using only m steps

Enumerating a few terms of $G(k)$ makes the algorithm obvious

$$G(k) = \left(g^1\right)^{k_{[0]}} \bullet \left(g^2\right)^{k_{[1]}} \bullet \left(g^4\right)^{k_{[2]}} \bullet \left(g^8\right)^{k_{[3]}} \cdots \left(g^{2^{m-1}}\right)^{k_{[m-1]}} \bmod 2^m$$

Note that $k_{[j]}=0$ or $k_{[j]}=1$, so that each term $\left(g^n\right)^{k_{[j]}}$ evaluates to either 1 or g^n

Algorithm G:

$\quad\quad G \leftarrow 1, \quad\quad h \leftarrow g, \quad\quad i \leftarrow k + 2^m \bmod 2^m$

$\quad\quad$ while $\quad i > 0$

$\quad\quad\quad\quad$ if $\quad i = $ odd: $\quad\quad G \leftarrow G\,h \bmod 2^m$

$\quad\quad\quad\quad h \leftarrow h^2 \bmod 2^m$

$\quad\quad\quad\quad i \leftarrow \lfloor i/2 \rfloor$

Remarks

• Algorithm G terminates after m steps, rather than k steps

• Negative strides are trivial, due to periodicity: $\quad\quad G(-s) = G(2^m - s)$

Define $\quad C(k) = c\left(\dfrac{g^k - 1}{g - 1}\right) \bmod 2^m$

$\qquad\qquad\quad = c \cdot \left(1 + g + g^2 + g^3 + \ldots + g^{k-1}\right) \bmod 2^m$

The series for C(k) can be evaluated recursively, similar to G(k), in **m** steps:

Algorithm C:

$\qquad C \leftarrow 0, \quad f \leftarrow c, \quad h \leftarrow g, \quad i \leftarrow k + 2^m \bmod 2^m$

$\qquad$ while $\quad i > 0$

$\qquad\qquad$ if $\quad i = $ odd: $\qquad C \leftarrow C\,h + f \ \bmod 2^m$

$\qquad\qquad f \leftarrow f\,(h + 1) \ \bmod 2^m$

$\qquad\qquad h \leftarrow h^2 \bmod 2^m$

$\qquad\qquad i \leftarrow \lfloor i / 2 \rfloor$

- Since most of the common random number generators use c = 0, Algorithm C is generally not required.

- Algorithm C can be included with Algorithm A, at very little extra cost

---

R. N. Generator for 32-bit machines $\qquad$ (*sparc2, rs6000, indigo, ...*)

$\qquad s \leftarrow 69069 \cdot s + 1 \quad \bmod 2^{32}$

```
                              static unsigned long      seed_c=1;
                              static double             norm=(1./4294967296.);

Random Number Generator →    double cranf_( void ) {
                                  unsigned long   g=69069, c=1;
                                  seed_c = g * seed_c + c;
                                  return  ((double) seed_c * norm );
                              }

Routine for Arbitrary Skips →  void cranfjump_(    unsigned long   *seed,
                                                   double          *jump,
                                                   unsigned long   *newseed ) {

                                  unsigned long   j, gen=1, inc=0,  g=69069,  c=1;

                                  if( *jump < 0 )   j = *jump + 4294967296.;
                                  else              j = *jump;

                                  for( ; j; j>>=1 ) {
Compute:
                                      if( j&1 ) {
gen = g^k                                 inc  = inc*g + c;
                                          gen  = gen*g;
inc = c(g^k-1)/(g-1)                  }
                                      c *= g+1;
                                      g *= g;
                                  }
                                  *newseed  = gen * (*seed)  +  inc;
                              }
```

RNG & Skip Ahead – Example

Fortran, 48-bit generator: $g=5^{19}$, c=0, m=48 (VIM & MCNP)

C, 32-bit generator: g=69069, c=1, m=32 (from Marsaglia)

			Sparc2	rs6000/350
C, 32-bit				
random number			1.0 µs	.7 µs
skip forward,	average for +1...10^5		7.4 µs	10 µs
skip backward,	average for -1...-10^5		4.0 µs	20 µs
Fortran, 48-bit				
random number			3.6 µs	2.3 µs
skip forward,	+152,917		163 µs	78 µs
skip backward,	-152,917		458 µs	215 µs
skip forward,	average for +1...10^5		160 µs	75 µs
skip backward,	average for -1...-10^5		695 µs	232 µs
skip forward,	+1,152,917		189 µs	90 µs
skip forward,	+1,152,917, **brute force**		4.1 **sec**	2.6 **sec**
skip backward,	-1,152,917		456 µs	210 µs
skip backward,	-1,152,917, **brute force**		8 **year**	5 **year**

Random Number Generators – Skip Ahead

- Algorithms for direct skip-ahead in the random sequence are simple, fast, convenient, ..., for modern Monte Carlo codes

- Arbitrary positive or negative strides can be taken, without precomputing or hardwiring specific constants

- Direct skip-ahead simplifies the initialization of random numbers for each particle, especially for parallel processing

- Algorithms described are currently used in:

parallel VIM	— ANL	— Sun, rs6000, SP1, ...
RACER	— KAPL	— Cray, Meiko CS1 & CS2, Sun, SGI, ...
KENO-Va	— CSN (Spain)	— Convex-C3440
MCNP5	**—- LANL**	**—- all machines**

MCNP5

Random Number

Generation & Testing

– **Knuth statistical tests**
– **Marsaglia's DIEHARD test suite**
– **Spectral test**
– **Performance test**
– **Results**

MCNP5 RNG: History

- **MCNP & related precursor codes**
  - 40+ years of intense use
  - Many different computers & compilers
  - Modern versions are parallel: MPI + threads
  - History based: Consecutive RNs used for primary particle,
    then for each of it's secondaries in turn, etc.
  - RN generator is small fraction of total computing time ($\sim$ 5%)

- **Traditional MCNP RN Algorithm**
  - Linear congruential, multiplicative

  $$S_{n+1} = g\, S_n \ \mathrm{mod}\ 2^{48}, \quad g = 5^{19}$$

  - 48-bit integer arithmetic, carried out in 24-bit pieces
  - Stride for new histories: 152,917
  - Skip-ahead: crude, brute-force
  - Period / stride   = 460 x 10^6 histories
  - Similar RN generators in RACER, RCP, MORSE, KENO, VIM

MCNP5 RNG: Requirements

- **Algorithm**
  - Robust, well-proven
  - Long period: $> 10^9$ particles x stride 152,917 $= 10^{14}$ RNs
  - $>10^9$ parallel streams
  - High-precision is **not** needed, low-order bits not important
  - Must have fast skip-ahead procedure
  - Reasonable theoretical basis, no correlation within or between histories

- **Coding**
  - Robust !!!!   Must never fail.
  - Rapid initialization for each history
  - Minimal amount of state information
  - Fast, but portable – must be exactly reproducible on any computer/compiler

---

MCNP5 RNG: Algorithm

- **Linear congruential generator (LCG)**

$$S_{n+1} = g\, S_n + c \ \ \text{mod}\ 2^m,$$

Period $= 2^m$ (for c>0) or 2^{m-2} (for c=0)

Traditional MCNP: m=48, c=0   Period=10^{14}, 48-bit integers
MCNP5:   m=63, c=1   Period=10^{19}, 63-bit integers

How to pick g and c ???

- **RN Sequence & Particle Histories**

····················   ·················   ·················
        1   2   3   etc.

  - Stride for new history: 152,917

MCNP5 RNG: Coding

- **RN Generation in MCNP-5**

    – RN module, entirely replaces all previous coding for RN generation

    – Fortran-90, using INTEGER(I8) internally,
      where I8=selected_int_kind(18)

    – All parameters, variables, & RN generator state are PRIVATE,
      accessible only via "accessor" routines

    – Includes "new" skip-ahead algorithm for fast initialization of histories,
      greatly simplifies RN generation for parallel calculations

    – Portable, standard, thread-safe

    – Built-in unit test, compile check, and run-time test

    – Developed on PC, tested on SGI, IBM, Sun, Compaq, Mac, alpha

---

Extended generators : 63-bit LCGs

- **Selection of multiplier, increment and modulus**

$$S_{n+1} = 5^{19} S_n + 0 \mod 2^{48} \quad \text{(MCNP4)}$$

$$5^{23}, 5^{25} \quad\quad 1 \quad\quad 2^{63}$$

- **Multiplicative LCG(g, 0, 2^β)**

$g \equiv \pm 3 \mod 8, S_0 = \text{odd}$ ⟹ Period : $2^{\beta-2}$

- **Mixed LCG(g, c, 2^β)**

$g \equiv 1 \mod 4, c = \text{odd}$ ⟹ Period : 2^β

- **MCNP5 – Extension of multiplier**
    – 5^{19} = 45-bit integer in the binary representation
    – 5^{19} seems to be slightly small in 63-bit environment.
    – Odd powers of 5 satisfy both conditions above.
    – **Try these:** $(5^{19},0,2^{63})$, $(5^{23},0,2^{63})$, $(5^{25},0,2^{63})$,
                            $(5^{19},1,2^{63})$, $(5^{23},1,2^{63})$, $(5^{25},1,2^{63})$

L'Ecuyer's 63-bit LCGs

- **L'Ecuyer suggested 63-bit LCGs with good lattice structures.**
  Math. Comp., **68**, 249–260 (1999)

  - Good multipliers were chosen based on the **spectral test**.

  - **Multiplicative LCGs**
    - LCG(3512401965023503517, 0, 2^{63})
    - LCG(2444805353187672469, 0, 2^{63})
    - LCG(1987591058829310733, 0, 2^{63})

  - **Mixed LCGs**
    - LCG(9219741426499971445, 1, 2^{63})
    - LCG(2806196910506780709, 1, 2^{63})
    - LCG(3249286849523012805, 1, 2^{63})

Tests for RNGs

- **13 different LCGs were tested:**
  - **Traditional MCNP RNG, (5^{19}, 0, 2^{48})**
  - **6 – Extended 63-bit LCGs**
  - **6 – L'Ecuyer's 63-bit LCGs**

- **Theoretical tests :**
  - Analyze the RNG algorithm of based on number theory and the theory of statistics.
  - Theoretical tests depend on the type of RNG. (LCG, Shift register, Lagged Fibonacci, etc.)
  - For LCGs, the **Spectral test** is used

- **Empirical tests :**
  - Analyze the uniformity, patterns, etc. of RNs generated by RNGs .
  - **Standard tests** – reviewed by D. Knuth, SPRNG test routines
  - **DIEHARD tests** – Bit level tests by G. Marsaglia, more stringent
  - Physical tests – RNGs are used in a practical application. The exact solutions for the tests are known. (not performed in this work)

Spectral test

- **LCGs have regular patterns (lattice structures) when overlapping t-tuples of a random number sequence are plotted in a hypercube. (Marsaglia, 1968).**

- **all the t-tuples are covered with families of parallel $(t-1)$-dimensional hyperplanes.**

- **The spectral test determines the maximum distance between adjacent parallel hyperplanes.**

2 - 33

Illustration of the spectral test

- Example: $S_{n+1} = 137\ S_n + 187 \bmod 256$

pair   pair

0.26562, 0.12109, 0.32031, 0.61328, 0.75000, …

pair   pair

2 - 34
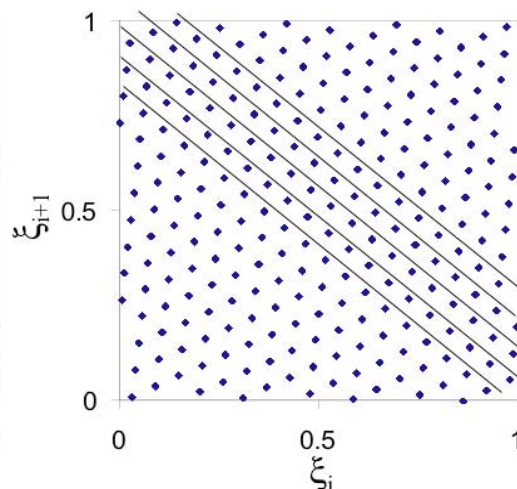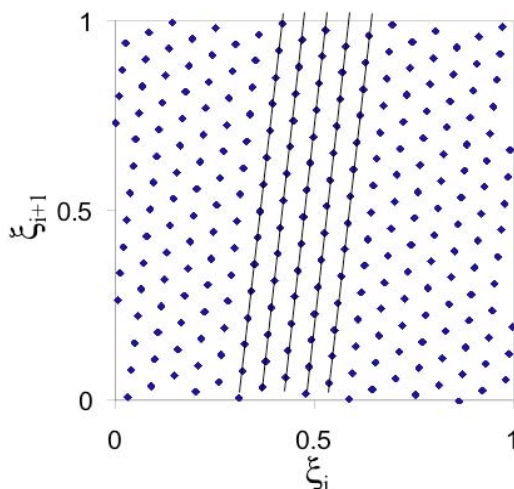
Measures for Spectral Test Criterion & Ranking

- μ **value proposed by Knuth**
  - Represent the effectiveness of a multiplier.

Knuth's criterion

$\mu_t(m,g)$ for $2 \le t \le 6$	Result
$\mu_t(m,g) > 1$	Pass with flying colors
$0.1 \le \mu_t(m,g) \le 1$	Pass
$\mu_t(m,g) \le 0.1$	Fail

- **S value**
  - Normalized maximum distance

$$S_t = \frac{d_t^*(m)}{d_t(m,g)}$$

  $d_t(m,g)$: Maximum distance between adjacent parallel hyperplanes.

  $d_t^*(m)$: Lower bound on $d_t(m,g)$.

  - The closer to 1 the S value is, the better the RNG is.

Spectral test for extended LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(5^{19},0,2^{63})							
$\mu_t(m,g)$	1.7321	2.1068	2.7781	1.4379	0.0825	2.0043	5.9276
$S_t(m,g)$	0.6910	0.7085	0.7284	0.6266	0.3888	0.6573	0.7414
LCG(5^{23},0,2^{63})							
$\mu_t(m,g)$	0.0028	1.9145	2.4655	5.4858	0.3327	0.2895	6.6286
$S_t(m,g)$	0.0280	0.6863	0.7070	0.8190	0.4906	0.4986	0.7518
LCG(5^{25},0,2^{63})							
$\mu_t(m,g)$	0.3206	1.8083	0.0450	3.0128	0.3270	3.1053	0.4400
$S_t(m,g)$	0.2973	0.6733	0.2598	0.7265	0.4892	0.6998	0.5356
LCG(5^{19},1,2^{63})							
$\mu_t(m,g)$	1.7321	2.9253	2.4193	0.3595	0.0206	0.5011	1.6439
$S_t(m,g)$	0.6910	0.7904	0.7036	0.4749	0.3086	0.5392	0.6316
LCG(5^{23},1,2^{63})							
$\mu_t(m,g)$	0.0007	2.8511	2.5256	3.1271	4.5931	1.8131	4.2919
$S_t(m,g)$	0.0140	0.7837	0.7112	0.7319	0.7598	0.6480	0.7121
LCG(5^{25},1,2^{63})							
$\mu_t(m,g)$	0.0801	3.4624	1.3077	1.0853	1.4452	0.7763	1.3524
$S_t(m,g)$	0.1486	0.8361	0.6033	0.5923	0.6266	0.5740	0.6163

Spectral test for L'Ecuyer's 63-bit LCGs

Dimension(t)	2	3	4	5	6	7	8
LCG(3512401965023503517,0,2^{63})							
$\mu_t(m,g)$	2.9062	2.9016	3.1105	4.0325	5.3992	6.7498	7.2874
$S_t(m,g)$	0.8951	0.7883	0.7493	0.7701	0.7806	0.7818	0.7608
LCG(2444805353187672469,0,2^{63})							
$\mu_t(m,g)$	2.2588	2.4430	6.4021	2.9364	3.0414	5.4274	4.6180
$S_t(m,g)$	0.7891	0.7443	0.8974	0.7228	0.7094	0.7579	0.7186
LCG(1987591058829310733,0,2^{63})							
$\mu_t(m,g)$	2.4898	3.4724	1.7071	2.5687	2.1243	2.0222	4.1014
$S_t(m,g)$	0.8285	0.8369	0.6449	0.7037	0.6682	0.6582	0.7080
LCG(9219741426499971445,1,2^{63})							
$\mu_t(m,g)$	2.8509	2.8046	3.5726	3.8380	3.8295	6.4241	6.8114
$S_t(m,g)$	0.8865	0.7794	0.7757	0.7625	0.7371	0.7763	0.7544
LCG(2806196910506780709,1,2^{63})							
$\mu_t(m,g)$	1.9599	4.0204	4.4591	3.1152	3.0728	3.0111	3.7947
$S_t(m,g)$	0.7350	0.8788	0.8199	0.7314	0.7106	0.6967	0.7012
LCG(3249286849523012805,1,2^{63})							
$\mu_t(m,g)$	2.4594	2.4281	3.7081	2.8333	3.7633	3.0844	1.9471
$S_t(m,g)$	0.8234	0.7428	0.7829	0.7176	0.7350	0.6991	0.6451

Results of spectral test

- **Results for the traditional MCNP RNG**

Dimension(t)	2	3	4	5	6	7	8
$\mu_t(m,g)$	3.0233	0.1970	1.8870	0.9483	1.8597	0.8802	1.2931
$S_t(m,g)$	0.9129	0.3216	0.6613	0.5765	0.6535	0.5844	0.6129

- **All extended 63-bit LCGs fail with Knuth's criterion.**
- **All L'Ecuyer's 63-bit LCGs pass with flying colors.**
- **Comparison of minimum S values**

RNG	Minimum $S_t(m,g)$
LCG(5^{19},0,2^{48})	0.3216
LCG(3512401965023503517,0,2^{63})	0.7493
LCG(2444805353187672469,0,2^{63})	0.7094
LCG(1987591058829310733,0,2^{63})	0.6449
LCG(9219741426499971445,1,2^{63})	0.7371
LCG(2806196910506780709,1,2^{63})	0.6967
LCG(3249286849523012805,1,2^{63})	0.6451

Standard test suite in SPRNG

- **SPRNG (Scalable Parallel Random Number Generators)**
  - Test programs are available. http://sprng.cs.fsu.edu

- **Standard test suite (Knuth)**
  - Equidistribution
  - Serial
  - Gap
  - Poker
  - Coupon collector's
  - Permutation
  - Runs-up
  - Maximum-of-t
  - Collision tests

- **Choice of test parameters**
  - L'Ecuyer's test suite : Comm. ACM **31** p.742 (1988)
  - Vattulainen's test suite : Comp. Phys. Comm. **86** p.209 (1995)
  - Mascagni's test suite : Submitted to Parallel Computing

Equidistribution test

- Check whether RNs are uniformly generated in [0, 1).
- Generate random integers in [0,d-1].
- Each integer must have the equal probability 1/d.

0.10574, 0.66509, 0.46622, 0.93925, 0.26551, 0.11361, …

$$\lfloor d * \xi_i \rfloor$$

0, 5, 3, 7, 2, 0, 2, 3, 1, 4, …

Count frequencies of 0 ~ d-1.
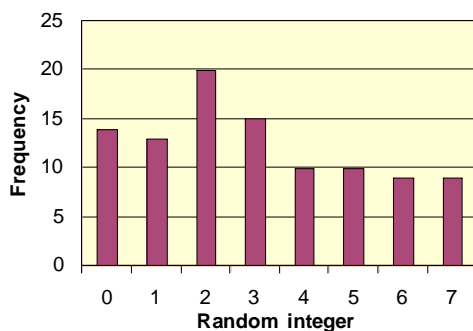
Cumulative chi-square distribution
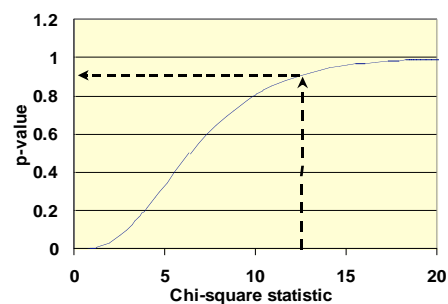
$$V = \sum_{s=1}^{k} \frac{(Y_s - np_s)^2}{np_s}$$

Criterion of "Pass or Failure"

- All empirical tests score a statistic.
- A goodness-of-fit test is performed on the test statistic and yield a p-value. (Chi-sqaure or Kolmogorov-Smirnov test)
- If the p-value is close to 0 or 1, a RNG is suspected to fail.
- Significance level : 0.01(1%)
- Repeat each test 3 times.
- **All 3 p-values are suspicious, then the RNG fails.**
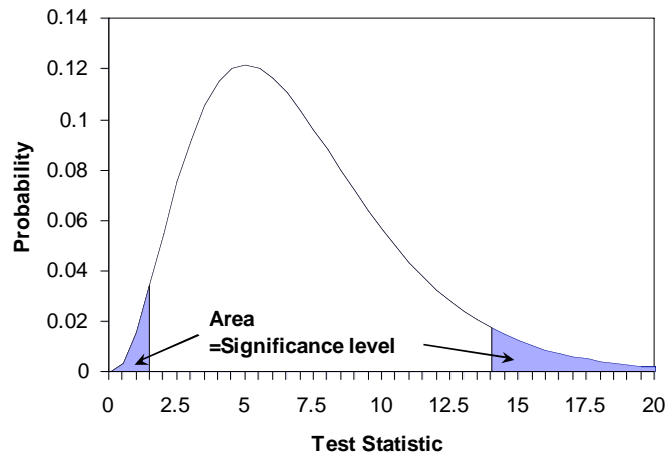
DIEHARD test suite

- **DIEHARD test**
  - A battery of tests proposed by G. Marsaglia.
  - Test all bits of random integers, not only the most significant bits.
  - More stringent than standard Knuth tests.
  - Default test parameters were used in this work.
  - Test programs are available. http://stat.fsu.edu/~geo/diehard.html
- **Included tests:**
  - **Birthday spacings**
  - **Overlapping 5-permutation**
  - **Binary rank**
  - **Bitstream**
  - **Overlapping-pairs-sparse-occupancy (OPSO)**
  - **Overlapping-quadruples-sparse-occupancy (OQSO)**
  - **DNA**
  - **Count-the-1's test on a stream of bytes**
  - **Count-the-1's test for specific bytes**
  - **Parking lot**
  - **Minimum distance**
  - **3-D spheres**
  - **Squeeze**
  - **Overlapping sums**
  - **Runs**
  - **Craps**

Overlapping-pairs-sparse-occupancy test (1)

Los Alamos

- OPSO = Overlapping-Pairs-Sparse-Occupancy test
- Preparation of 32-bit integers

0.10574, 0.66509, 0.46622, 0.93925, 0.26551, 0.11361, …

$$\left\lfloor 2^{32} * \xi_i \right\rfloor$$

454158374, 2856527213, 2002411287, 4034027575, …

Binary representation

11011000100011110100000100110,
10101010010000110010010101101101, …

- Letter : a designated string of consecutive 10 bits

1101100010001111010 0000100110,
1010101001000011001001 0101101101, …

Letter : 2^{10} = 1024 patterns
(letters)

Overlapping-pairs-sparse-occupancy test (2)

Los Alamos

- 2-letter words are formed from an alphabet of 1024 letters.
  0000100110, 0101101101, 1100010111, 0000110111, …

Decimal representation

38, 365, 791,   55, …

2-letter word   2-letter word

- Count the number of missing words (=j).

- The number of missing words should be very closely normally distributed with mean 141,909, standard deviation 290.

Cumulative normal distribution

p-value vs z
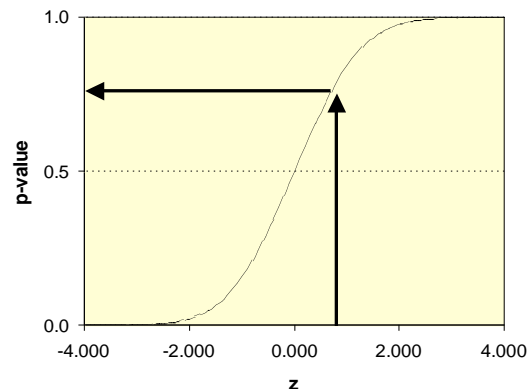
$$z = \frac{j - 141909}{290}$$

Overlapping-quadruples-sparse-occupancy test

- OQSO = Overlapping-Quadraples-Sparse-Occupancy test
- Similar to the OPSO test.
- Letter : a designated string of consecutive 5 bits

    11011000100011110100000100110,
    10101010010000110010010101101101, …

    Letter : 2^5 = 32 letters

- 4-letter words are formed from an alphabet of 32 letters.
    00110, 01101, 10111, 10111, …

    4-letter word

- The number of missing words should be very closely normally distributed with mean 141909, standard deviation 295.

DNA test

- Similar to the OPSO and OQSO tests.
- Letter : a designated string of consecutive 2 bits

    1101100010001111010000010010,
    10101010010000110010010101101101, …

    Letter : 2^2 = 4 letters

- 10-letter words are formed from an alphabet of 4 letters.
    10, 1, 11, 11, 11, 1, 10, 0, 11, 10, …

    10-letter word

- The number of missing words should be very closely normally distributed with mean 141909, standard deviation 399.

DIEHARD Test Suite

- **Criterion for DIEHARD test**

  - If the p-value is close to 0 or 1, a RNG is suspected to fail.

  - Significance level : 0.01(1%)

  - **A RNG fails the test if we get six or more p-values less than 0.01 or more than 0.99.**

- **Results for standard & DIEHARD tests**

  - **All 13 RNGs pass all standard tests with L'Ecuyer's, Vattulainen's and Mascagni's test parameters.**

  - **Extended and L'Ecuyer's 63-bit LCGs pass all the DIEHARD tests.**

  - **The traditional MCNP RNG fails the OPSO, OQSO and DNA tests in the DIEHARD test suite.**

Result of OPSO test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value
bits 23 to 32	**0.0000**	bits 11 to 20	0.7457
bits 22 to 31	**0.0000**	bits 10 to 19	0.0598
bits 21 to 30	**0.0000**	bits 9 to 18	0.1122
bits 20 to 29	**0.0000**	bits 8 to 17	0.4597
bits 19 to 28	**0.0001**	bits 7 to 16	**0.0011**
bits 18 to 27	0.6639	bits 6 to 15	0.6319
bits 17 to 26	0.0445	bits 5 to 14	0.7490
bits 16 to 25	0.0125	bits 4 to 13	0.2914
bits 15 to 24	0.7683	bits 3 to 12	0.1792
bits 14 to 23	0.9712	bits 2 to 11	0.3253
bits 13 to 22	0.1077	bits 1 to 10	0.7277
bits 12 to 21	0.0717		

Result of OQSO test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value
bits 28 to 32	**1.0000**	bits 14 to 18	0.6487
bits 27 to 31	**1.0000**	bits 13 to 17	0.5575
bits 26 to 30	**1.0000**	bits 12 to 16	0.1634
bits 25 to 29	**1.0000**	bits 11 to 15	0.6600
bits 24 to 28	**1.0000**	bits 10 to 14	0.2096
bits 23 to 27	**1.0000**	bits 9 to 13	0.3759
bits 22 to 26	**0.0000**	bits 8 to 12	0.9191
bits 21 to 25	**0.0000**	bits 7 to 11	0.8554
bits 20 to 24	**0.0000**	bits 6 to 10	0.5535
bits 19 to 23	0.1906	bits 5 to 9	0.4955
bits 18 to 22	**0.0011**	bits 4 to 8	0.0868
bits 17 to 21	0.3823	bits 3 to 7	0.1943
bits 16 to 20	0.8394	bits 2 to 6	0.8554
bits 15 to 19	0.2518	bits 1 to 5	0.7421

Result of DNA test for traditional MCNP RNG

Tested bits	p-value	Tested bits	p-value	Tested bits	p-value
bits 31 to 32	**1.0000**	bits 20 to 21	0.4937	bits 9 to 10	0.4550
bits 30 to 31	**1.0000**	bits 19 to 20	0.0613	bits 8 to 9	0.4737
bits 29 to 30	**1.0000**	bits 18 to 19	0.2383	bits 7 to 8	0.7834
bits 28 to 29	**1.0000**	bits 17 to 18	0.4831	bits 6 to 7	0.4063
bits 27 to 28	**1.0000**	bits 16 to 17	0.0925	bits 5 to 6	0.8959
bits 26 to 27	0.1777	bits 15 to 16	0.0197	bits 4 to 5	0.3438
bits 25 to 26	**0.0000**	bits 14 to 15	0.7377	bits 3 to 4	0.3972
bits 24 to 25	**0.0000**	bits 13 to 14	0.7171	bits 2 to 3	0.8986
bits 23 to 24	**0.0000**	bits 12 to 13	0.0309	bits 1 to 2	0.5407
bits 22 to 23	**0.0000**	bits 11 to 12	0.2803		
bits 21 to 22	**0.0000**	bits 10 to 11	0.8440		

• **Less significant (lower) bits of RNs fail the tests.**

• **These failures in less significant bits are caused by the shorter period than the significant bits.**

Drawback of LCGs with power-of-two modulus

The $(r+1)$-th most significant bit has period length at most 2^{-r} times that of the most significant bit.

• **However, these failures do not have a significant impact in the practical use.**

Performance test

• Los Alamos
NATIONAL LABORATORY

• Test program

```
. . . . .
integer(8) :: i
integer(8), parameter :: NumGeneratedRNs = 1000000000
!real(8)   :: rang ! For MCNP4
real(8)   :: RN_initial, RN_last
real(8)   :: dummy
 . . . . .

!call random ! For MCNP4
call RN_init_problem( new_standard_gen = 1 )

RN_initial = rang()

do i = 2, NumGeneratedRNs-1
      dummy = rang()
end do

RN_last = rang()
. . . . .
```

Results of performance test

- Comparison between MCNP4 and MCNP5
- Generate 1 billion RNs.

	MCNP4	MCNP5	MCNP4/MCNP5
CPU (sec) No optimization (/optimization:0)	290.0	97.1	3.0
CPU (sec) Local optimization (/optimization:1)	191.7	77.2	2.5
CPU (sec) Full optimization (/optimization:4)	188.4	78.1	2.4

Platform : Windows 2000, Intel Pentium III 1GHz

Compiler : Compaq Visual Fortran Ver.6.6

Summary

- The tradision MCNP RNG fails the OPSO, OQSO and DNA tests in the DIEHARD test suite.

- The 63-bit LCGs extended from the MCNP RNG fail the spectral test.

- **L'Ecuyer's 63-bit LCGs pass all the tests and their multipliers are excellent judging from the spectral test.**

- **These 63-bit LCGs are implemented in the RNG package for MCNP5**

- **The MCNP5 RNG is ~2.5 times faster than the MCNP4 RNG.**

Random Sampling

"Anyone who considers arithmetical methods of producing
random digits is, of course, in a state of sin."

John Von Neuman, 1951

Forrest B. Brown

Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Introduction

Probability ?

What are the odds of …..

• Being audited by the IRS this year	100	to 1
• Losing your luggage on a U.S. flight	176	to 1
• Being dealt 4 aces on an opening poker hand	4,164	to 1
• Being struck by lightning in your lifetime	9,100	to 1
• Being hit by a baseball at a major league game	300,000	to 1
• Drowning in your bathtub this year	685,000	to 1
• Winning Lotto in the Illinois lottery with 1 ticket	12,900,000	to 1
• Winning the grand prize in the Reader's Digest sweepstakes	199,500,000	to 1

Introduction – Probability Density Functions

Los Alamos

- **Continuous Probability Density**

  f(x) = probability density function (PDF)

  $f(x) \geq 0$

  $\text{Probability}\{a \leq x \leq b\} = \int_a^b f(x)dx$

  Normalization: $\int_{-\infty}^{\infty} f(x)dx = 1$

f(x)

a   b   x

- **Discrete Probability Density**

  $\{ f_k \},\ k = 1,...,N,\quad$ where $f_k = f(x_k)$

  $f_k \geq 0$

  $\text{Probability}\{ x = x_k \} = f_k$

  Normalization: $\sum_{k=1}^{N} f_k = 1$

f(x)

x_1 x_2 x_3 x_N x

Introduction

Los Alamos

The key to Monte Carlo methods is the notion of *random sampling*.

- **The problem can be stated this way:**

  Given a probability density, f(x), produce a sequence of $\hat{X}$'s.
  The $\hat{X}$'s should be distributed in the same manner as f(x).

f(x)

x →

- **The use of random sampling distinguishes Monte Carlo from other methods**

- **When Monte Carlo is used to solve the integral Boltzmann transport equation:**
  - **Random sampling models the outcome of physical events**
    (e.g., neutron collisions, fission process, sources, …..)
  - **Computational geometry models the arrangement of materials**

Monte Carlo & Random Sampling

- **Probability Density Function (PDF)**

   f(x) = probability density function (PDF)

   $f(x) \geq 0$

   $\text{Probability}\{a \leq x \leq b\} = \int\limits_{a}^{b} f(x)dx$

   Normalization: $\int\limits_{-\infty}^{\infty} f(x)dx = 1$
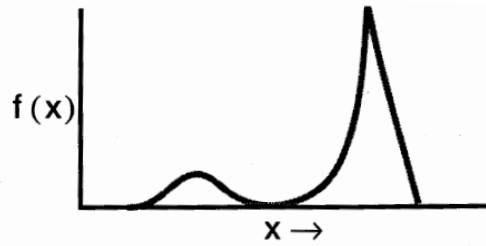
- **Cumulative Distribution Function (CDF)**

   $F(x) = \int\limits_{-\infty}^{x} f(x')dx'$

   $0 \leq F(x) \leq 1$

   $\dfrac{dF(x)}{dx} \geq 0$

   $F(-\infty) = 0, \quad F(\infty) = 1$

3 - 5

Monte Carlo & Random Sampling
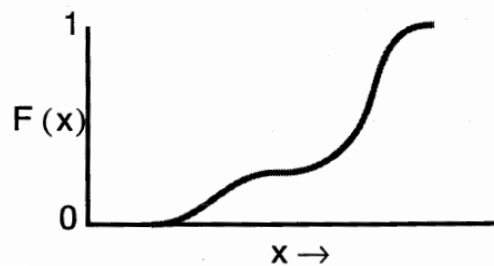
Monte Carlo Codes

Categories of random sampling
- **Random number generator** ➜ **uniform PDF on (0,1)**
- **Sampling from analytic PDFs** ➜ **normal, exponential, Maxwellian, …**
- **Sampling from tabulated PDFs** ➜ **angular PDFs, spectrum, …**

For Monte Carlo codes…
- **Random numbers, ξ, are produced by the RN generator on (0,1)**
- **Non-uniform random variates are produced from the ξ's by:**
    - **Direct inversion**
    - **Rejection methods**
    - **Transformations**
    - **Composition (mixtures)**
    - **Sums, products, ratios, …**
    - **Table lookup + interpolation**
    - **Lots (!) of other tricks**
- **Typically < 10% of total CPU time**

3 - 6

- **Pseudo-Random Numbers**
  - Not strictly "random", but good enough
    - Pass statistical tests for randomness
    - Reproducible sequence
  - Uniform PDF on (0,1)
  - Must be easy to compute
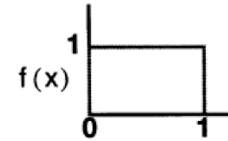
- **Linear Congruential Method**
  - Algorithm

$$S_0 = \text{initial seed, odd integer}, < M$$
$$S_k = G \cdot S_{k-1} + c \mod M, \quad k = 1, 2, \ldots$$
$$\xi_k = S_k / M$$
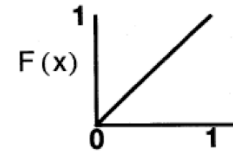
- **Usage**
  - In algorithms, usually denote RN uniform on (0,1) by ξ
  - In codes, invoke basic RN generator by: **r = ranf()**
  - Each new usage of ξ or ranf() generates a **new** RN

Direct Sampling

- **Direct solution of** $\hat{x} = F^{-1}(\xi)$

$$\text{Solve for } \hat{x}: \qquad \xi = \int_{-\infty}^{\hat{x}} f(x)dx$$

- **Sampling procedure**
  - **Generate ξ**
  - **Determine $\hat{x}$ such that F($\hat{x}$) = ξ**
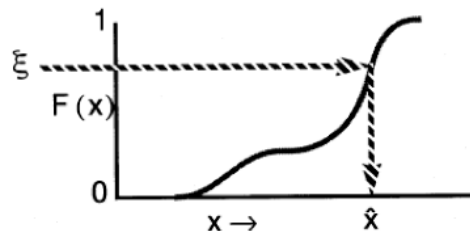
- **Advantages**
  - Straightforward mathematics & coding
  - "High-level" approach

- **Disadvantages**
  - Often involves complicated functions
  - In some cases, F(x) cannot be inverted (e.g., Klein-Nishina)

Rejection Sampling

Von Neumann

"..... it seems objectionable to compute a transcendental function of a random number."

Select a bounding function, g(x), such that
- $c \cdot g(x) \geq f(x)$ for all x
- g(x) is an easy-to-sample PDF

Sampling Procedure:
- sample $\hat{x}$ from g(x): $\quad \hat{x} \leftarrow G^{-1}(\xi_1)$

- test: $\quad \xi_2 \cdot cg(\hat{x}) \leq f(\hat{x})$

         if **true** → accept $\hat{x}$, done
         if **false** → reject $\hat{x}$, try again

cg(x)
- reject
f(x)
- accept
x →
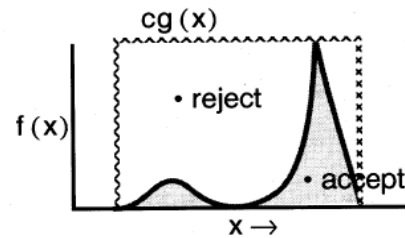
Advantages
- Simple computer operations

Disadvantages
- "Low-level" approach, sometimes hard to understand

Discrete PDF's

- **Discrete PDF**

$\{ f_k \}$, where $f_k = f(x_k)$, k=1,...,N

$f_k \geq 0$

$$\sum_{j=1}^{N} f_j = 1$$

f(x)

$x_1 \quad x_2 \quad\quad x_3 \quad x_N \quad x \rightarrow$
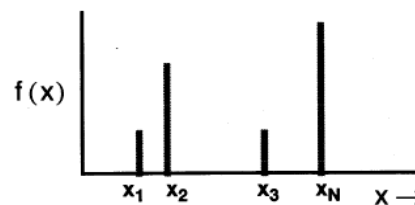
- **Discrete CDF**

$\{ F_k \}$, where $F_k = \sum_{j=1}^{k} f_j$, k=1, ..., N-1

and

$\quad F_0 = 0,$

$\quad F_N = 1$

1   F_N
F_3
F(x) F_2
F_1
0
x →
F_1
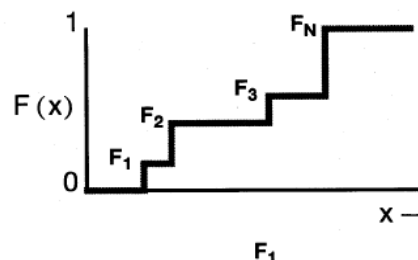
Sampling from Discrete PDF's — Conventional Procedure

Direct Solution of $\hat{x} \leftarrow F^{-1}(\xi)$

   (1) Generate ξ

   (2) Determine **k** such that $F_{k-1} \leq \xi \leq F_k$

   (3) Return $\hat{x} = x_k$
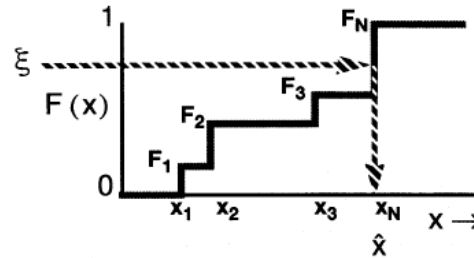
Step (2) requires a **table search**

   • **linear** table searches require O(N) time   — use when N small

   • **binary** table searches require $O(\ln_2 N)$ time   — use when N large

For some discrete PDFs, F_k's are not precomputed.

   • **linear** search, with F_k's computed on-the-fly as needed

Example — Sampling from Discrete Uniform PDF

Discrete Uniform PDF

   $f_k = 1 / N$, $k = 1, ..., N$

   $F_k = k / N$, $F_0 = 0$, $F_N = 1$

Sampling procedure:

   Could use table search method, ...
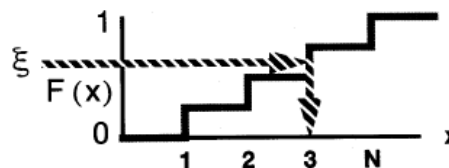
Easier, for this special case:

       $K \leftarrow \lfloor 1 + N\xi \rfloor$,   where $\lfloor y \rfloor$ is the "floor" function, largest integer < y

             Note: must be sure that $\lfloor 1 + N\xi \rfloor \leq N$

- **Multigroup Scattering**

  - Scatter from group **g** to group **g'**, where $1 \le g' \le G$

$$f_{g'} = \frac{\sigma_{g \to g'}}{\sum_{k=1}^{G} \sigma_{g \to k}}$$

- **Selection of scattering nuclide for a collision**

  - K = number of nuclides in composition

$$f_k = \frac{N^{(k)} \sigma_s^{(k)}}{\sum_{j=1}^{K} N^{(j)} \sigma_s^{(j)}}$$

Sampling from Discrete PDF's — Alias Method

Any discrete PDF can be converted into "Alias sampling" form

original PDF: $\{ f_k \}$,   k=1, ..., N

where f_k = probability of selecting x = x_k

aliased PDF: $\{ q_k, i_k, \}$,   k=1, ...,N

where $\frac{1}{N} \cdot q_k$ = prob. of selecting $\hat{x} = x_k$

$\frac{1}{N} \cdot (1 - q_k)$ = prob. of selecting $\hat{x} = x_{i_k}$

Alias sampling procedure:

Select <u>uniformly</u> for $\hat{k}$: $\hat{k} \leftarrow \lfloor 1 + N\xi_1 \rfloor$

Select <u>either</u> $\hat{k}$ or its "alias" $i_{\hat{k}}$:   if $\xi_2 < q_{\hat{k}}$, $\hat{x} \leftarrow x_{\hat{k}}$,

otherwise, $\hat{x} \leftarrow x_{i_{\hat{k}}}$

.....**(continued on next page)**

Sampling from Discrete PDF's — Alias Method (continued)

Why bother with "alias sampling" ?

→ **No table search needed, requires O(1) time**
→ **Sampling time is constant & independent of size of PDF**
→ **Vectorizes completely & efficiently**
→ **Fastest possible way to sample discrete PDFs**
→ **Invented by Brown** (who later found out Walker did it 3 yr earlier)

Creating the "aliased PDF" amounts to converting an N-way tree from

        arbitrary branching probabilities with **single** outcomes

  to

        uniform branching probabilities with **dual** outcomes

(See FB Brown & RACER coding for the set up algorithm)

3 -15
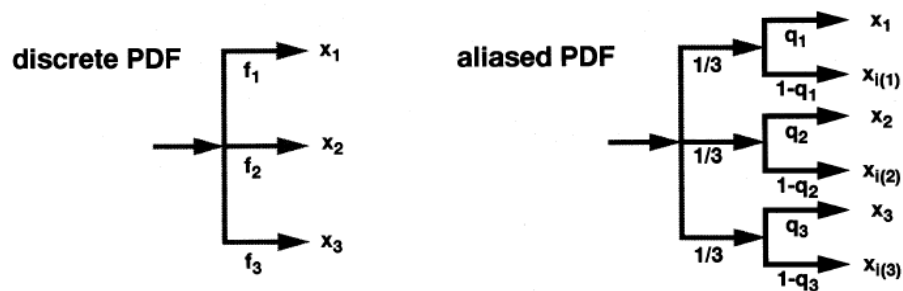
Example – Sampling from uniform PDF in range (a,b),
Histogram with 1 bin

$$x \leftarrow a + \xi \cdot (b-a)$$

3 -16
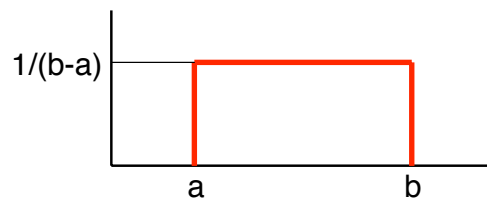
Example – Sampling from histogram with 2 bins

$A_1 = (x_1 - x_0) \cdot f_1$

$A_2 = (x_2 - x_1) \cdot f_2$

$f(x)$
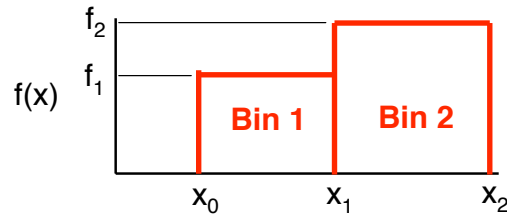
$p_1 = \text{Prob}\{ x_0 < x < x_1 \} = A_1 / (A_1 + A_2)$

$p_2 = \text{Prob}\{ x_1 < x < x_2 \} = A_2 / (A_1 + A_2)$

$p_1 + p_2 = 1$

Two-step sampling procedure:

1. **Select a bin, b:**

   If $\xi_1 < p_1$,   select $b = $ bin 1

   otherwise,   select $b = $ bin 2

2. **Sample x within bin:**

   $x \leftarrow x_{b-1} + \xi_2 \cdot (x_b - x_{b-1})$

Example – Sampling from Histogram PDF
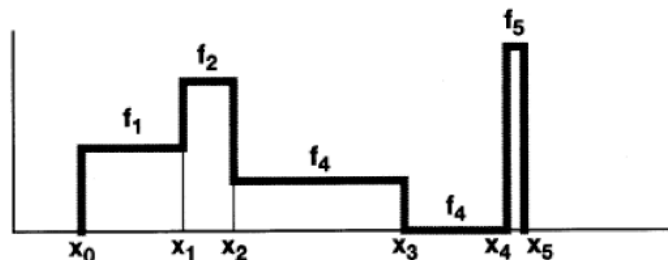
Two-step sampling: (1) Sample from discrete PDF to select a bin

           (2) Sample from uniform PDF within bin

- **Discrete PDF:** $p_k = f_k \cdot (x_k - x_{k-1})$, $k = 1, \ldots, N$, $\Sigma p_k = 1$
  - Generate ξ_1
  - Use table search or alias method to select k
- **Uniform sampling within bin k**
  - Generate ξ_2
  - Then, $x \leftarrow x_{k-1} + (x_k - x_{k-1}) \cdot \xi_2$

Examples — Sampling from Linear PDF on (0,1)

$$f(x) = 2x, \qquad 0 \le x \le 1$$

$$F(x) = \int_0^x f(x')\,dx' = \int_0^x 2x'\,dx' = x^2$$
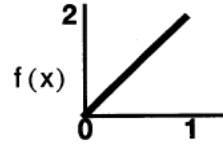
Direct Sampling:

solving $F(\hat{x}) = \xi$ or $\hat{x} \leftarrow F^{-1}(\xi)$

gives:

$$\hat{x} \leftarrow \sqrt{\xi}$$
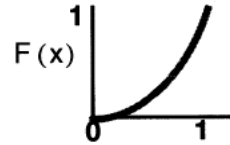
Examples — Sampling from x^n PDF on (0,1)
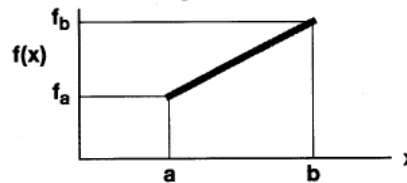
$$f(x) = (n+1)x^n, \qquad 0 \le x \le 1$$

$$F(x) = x^{n+1}$$

Solving $F(\hat{x}) = \xi$ gives: $\hat{x} \leftarrow \xi^{\frac{1}{n+1}}$

(Note: only for 0<x<1, does not apply to general intervals !)

Examples — Sampling from Arbitrary Linear PDF

Scheme 1	Scheme 2
Decompose into uniform + linear	**Decompose into linear + linear**
$p_1 = f_a(b-a)$	$p_1 = f_a(b-a)/2$
$p_2 = (f_b - f_a)(b-a)/2$	$p_2 = f_b(b-a)/2$
$p_1 + p_2 = 1$	$p_1 + p_2 = 1$
$f(x) = p_1 \cdot \square + p_2 \cdot \triangleleft$	$f(x) = p_1 \cdot \triangle + p_2 \cdot \triangleleft$
Sampling scheme:	**Sampling scheme:**
if $\xi_1 < p_1$, $\qquad \hat{x} \leftarrow a + (b-a)\xi_2$	if $\xi_1 < p_1$, $\qquad \hat{x} \leftarrow b - (b-a)\sqrt{\xi_2}$
else $\qquad \hat{x} \leftarrow a + (b-a)\sqrt{\xi_2}$	else $\qquad \hat{x} \leftarrow a + (b-a)\sqrt{\xi_2}$
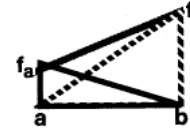
Examples — Sampling from Piecewise Linear PDF
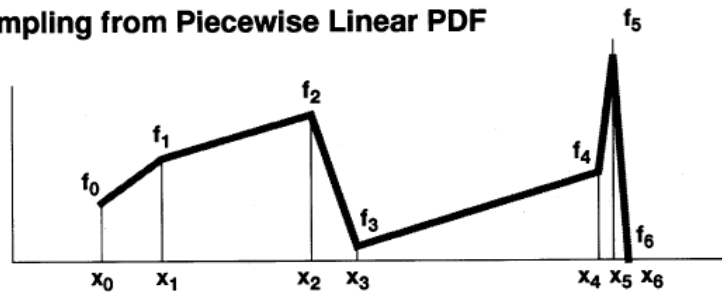
Two-step sampling: (1) **Sample from discrete PDF to select a bin**
                          (2) **Sample from linear PDF within bin**

- Discrete PDF: $p_k = \dfrac{(f_k + f_{k-1})}{2} \cdot (x_k - x_{k-1})$, $k = 1, ..., N$

   — generate ξ
   — use table search or alias method to select **K**

- Linear sampling within bin **K**:

   — generate ξ
   — then,   if $\xi_1 < \dfrac{f_{k-1}}{f_k + f_{k-1}}$, $\hat{x} \leftarrow x_k \quad - (x_k - x_{k-1})\sqrt{\xi_2}$

   otherwise $\hat{x} \leftarrow x_{k-1} + (x_k - x_{k-1})\sqrt{\xi_2}$

Examples – Sampling from an Exponential PDF

$$f(x) = \frac{1}{\lambda} \cdot e^{-x/\lambda}, \quad 0 \le x \le \infty$$

$$F(x) = \int_0^x f(x')dx' = 1 - e^{-x/\lambda}$$

Direct sampling:
   Solve for x: **F(x) = ξ**

   Solving $\xi = 1 - e^{-x/\lambda}$ gives: $x \leftarrow -\lambda \cdot \ln(1-\xi)$

   or

   $x \leftarrow -\lambda \cdot \ln\xi$

Although $(1- \xi) \ne \xi$,
both ξ and $(1- \xi)$ are uniformly distributed on (0,1),
so that we can use either in the random sampling procedure.
(I.e., the numbers are different, but the distributions are the same)
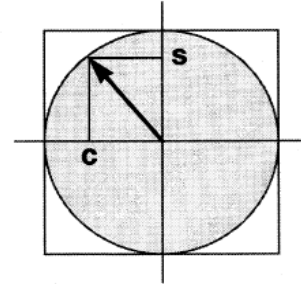
Example — 2D Isotropic

$$f(\vec{p}) = \frac{1}{2\pi}, \qquad \vec{p} = (u, v)$$

Rejection   (*old vim*)

```
        SUBROUTINE AZIRN_VIM( S, C )
        IMPLICIT DOUBLE PRECISION  (A-H, O-Z)
100 R1=2.*RANF() - 1.
        R1SQ=R1*R1
        R2=RANF()
        R2SQ=R2*R2
        RSQ=R1SQ+R2SQ
        IF(1.-RSQ)100,105,105
105 S=2.*R1*R2/RSQ
        C=(R2SQ-R1SQ)/RSQ
        RETURN
        END
```

Direct   (*racer, new vim*)

```
subroutine azirn_new( s, c )
implicit double precision  (a-h,o-z)
parameter ( twopi = 2.*3.14159265 )
phi = twopi*ranf()
c = cos(phi)
s = sin(phi)
return
end
```

3 -23

Example — Watt Spectrum

$$f(x) = \frac{2e^{-ab/4}}{\sqrt{\pi a^3 b}} e^{-x/a} \sinh \sqrt{bx}, \qquad 0 < x$$

Rejection   (*mcnp*)
- Based on Algorithm **R12** from <u>3rd Monte Carlo Sampler</u>, Everett & Cashwell

- Define $K = 1 + ab/8, \quad L = a\{K + (K^2 - 1)_{1/2}\}, \quad M = L/a - 1$

- Set $\quad x \leftarrow -\log \xi_1, \qquad y \leftarrow -\log \xi_2$

- If $\quad \{y - M(x + 1)\}^2 \le bLx, \qquad$ accept: return (Lx)

  otherwise, $\qquad\qquad\qquad\qquad$ reject

Direct   (*new vim*)
- Sample from Maxwellian in C-of-M, transform to lab

$$w \leftarrow a\left(-\log \xi_1 - \log \xi_2 \cos^2 \frac{\pi}{2}\xi_3\right)$$

$$x \leftarrow w + \frac{a^2 b}{4} + (2\xi_4 - 1)\sqrt{a^2 bw}$$

(assume isotropic emission from fission fragment moving with constant velocity in C-of-M)

- Unpublished sampling scheme, based on original Watt spectrum derivation

3 -24

Example — Linear PDF

$$f(x) = 2x, \qquad 0 \le x \le 1$$

Rejection

(strictly — this is not "rejection", but has the same flavor)

$$\text{if } \xi_1 \ge \xi_2, \quad \text{then} \quad \hat{x} \leftarrow \xi_1$$
$$\text{else} \quad \hat{x} \leftarrow \xi_2$$

or

$$\hat{x} \leftarrow \max(\xi_1, \xi_2)$$

or

$$\hat{x} \leftarrow |\xi_1 - \xi_2|$$

Direct

$$F(x) = x^2, \qquad 0 \le x \le 1$$

$$\hat{x} \leftarrow \sqrt{\xi}$$

Probability Density Function	Direct Sampling Method
Linear: $f(x) = 2x, \qquad 0 < x < 1$	$x \leftarrow \sqrt{\xi}$
Exponential: $f(x) = e^{-x}, \qquad 0 < x$	$x \leftarrow -\log\xi$
2D Isotropic: $f(\vec{p}) = \dfrac{1}{2\pi}, \qquad \vec{p} = (u, v)$	$u \leftarrow \cos 2\pi\xi_1$ $v \leftarrow \sin 2\pi\xi_1$
3D Isotropic: $f(\vec{\Omega}) = \dfrac{1}{4\pi}, \qquad \vec{\Omega} = (u, v, w)$	$u \leftarrow 2\xi_1 - 1$ $v \leftarrow \sqrt{1 - u^2}\,\cos 2\pi\xi_2$ $w \leftarrow \sqrt{1 - u^2}\,\sin 2\pi\xi_2$
Maxwellian: $f(x) = \dfrac{2}{T\sqrt{\pi}}\sqrt{\dfrac{x}{T}}\,e^{-x/T}, \qquad 0 < x$	$x \leftarrow T\left(-\log\xi_1 - \log\xi_2\cos^2\frac{\pi}{2}\xi_3\right)$
Watt Spectrum: $f(x) = \dfrac{2e^{-ab/4}}{\sqrt{\pi a^3 b}}\,e^{-x/a}\sinh\sqrt{bx}, \qquad 0 < x$	$w \leftarrow a\left(-\log\xi_1 - \log\xi_2\cos^2\frac{\pi}{2}\xi_3\right)$ $x \leftarrow w + \dfrac{a^2 b}{4} + (2\xi_4 - 1)\sqrt{a^2 bw}$
Normal: $f(x) = \dfrac{1}{\sigma\sqrt{2\pi}}\,e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	$x \leftarrow \mu + \sigma\sqrt{-2\log\xi_1}\,\cos 2\pi\xi_2$

Machine Considerations

Vector Hardware

- Since ~1980, direct methods have been recommended for vectorization & high performance on *cray, cyber-205, sx-3, cm-2,* .....

  ➜ Vector concepts apply directly to pipelined RISC cpu's (e.g., *rs6000, i860, Fujitsu* μ-*vp*,...)

Math Libraries

- Many routines in math libraries are now *table-driven,* hence very fast

  ➜ fast *sin, cos, sqrt, log,* & *exp* functions

RISC + Compiler Technology

- Pipelining, concurrent ops, simple instructions, register-to-register ops, 64-bit hardware, better instruction scheduling, ...

  ➜ fast arithmetic   (even for double-precision)

  ➜ Today, the most expensive operations are
    - *load/store*   (memory access)
    - *IF...GOTO...*   (flush/fill instruction stack)

Software Considerations

"Rules of thumb" for M.C. algorithm design have changed

- Never take the square root of a random number
- Avoid using *sin, cos, log, exp,* .....
- Use *IF...GOTO...* to avoid arithmetic
- Random numbers are cheap, arithmetic is expensive

Direct sampling methods have advantages

- Clear, succinct coding — easier to verify & maintain
- Cpu time is comparable to rejection
- Direct methods vectorize efficiently

Random Sampling — Stratified Sampling

If a specific number of samples, M, is needed from a single distribution:

• Naive approach — repeat the sampling procedure M times

• **Stratified sampling** approach

    — partition the sample space into M disjoint regions of **equal probability**

    — produce 1 sample from each region
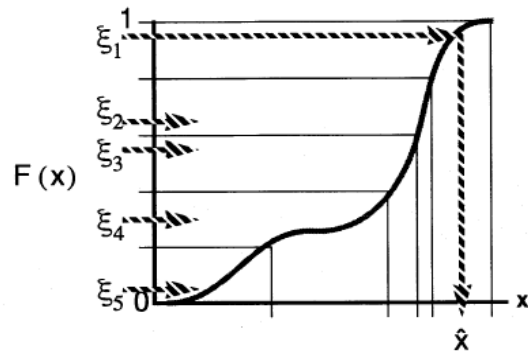
• Stratified sampling considerations

    — F(x) must be known & easy to partition

    — The number of partitions, M, must be known in advance

    — Must be relatively easy to sample **within** each given partition

    — Stratification improves the "coverage"

    — Stratified sampling reduces variance, at little or no computing cost

---

Random Sampling — Rejection Method

• **Rejection sampling methods are useful when it is difficult or impossible to invert F(x), or when F(x) is no known**

• **Example – Selection of initial source sites in a reactor**

    – **Select a trial site:**

$$x' \leftarrow x_1 + (x_2 - x_1) \cdot \xi_1$$
$$y' \leftarrow y_1 + (y_2 - y_1) \cdot \xi_2$$

    – **If (x',y') is inside a fuel pin (shaded region), then <u>accept</u> (x',y').**

    – **Otherwise, <u>reject</u> (x',y') and repeat**

    – **Efficiency of rejection sampling ~ (volume source region) / (total volume)**
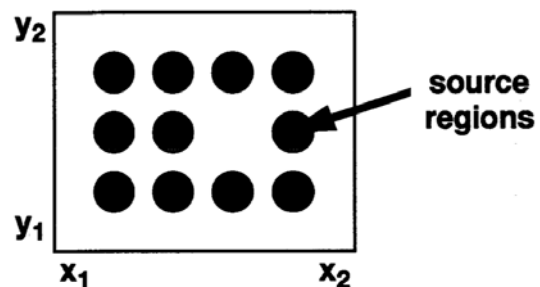
It is sometimes useful to sample from an alternate PDF

$$f(x)\,dx = \left[\frac{f(x)}{g(x)}\right]g(x)\,dx = h(x)\,g(x)\,dx$$

& then "correct" the result via either weight factors or a 2nd sampling stage

Weighted Sampling

- To sample $\hat{x}$ from f(x),
    - first, sample $\hat{x}$ from g(x)
    - then, multiply the "weight" assigned to $\hat{x}$ by $\dfrac{\text{right answer}}{\text{wrong answer}} = \dfrac{f(\hat{x})}{g(\hat{x})}$

- Note that g(x) must be >0 whenever f(x)>0.
- Also, g(x) must be normalized so that $\int g(x)\,dx = 1$

- **Example — survival biasing of collisions**

    If a collision occurs, $P_{survive} = \dfrac{\Sigma_S}{\Sigma_T}$ is the probability of surviving.

    Instead of sampling the outcome, always choose survival & multiply the "weight" by $P_{survive}$

Combined Russian Rouletting & Splitting

- Russian Roulette — kill off some particles, but conserve total weight
- Splitting — create extra identical particles, but conserve total weight

- Definitions

    wgt = Particle weight

    For the region containing the particle:

    w_{high} = upper bound on weight, if wgt larger — split
    w_{low} = lower bound on weight, if wgt lower — roulette
    w_{ave} = weight to assign survivors, $w_{low} < w_{ave} < w_{high}$

    Then,

    wgt / w_{ave} = probability of surviving split/roulette

- Combined game for split/roulette:

    if $\quad$ wgt $< w_{low}$ $\quad$ or $\quad$ $w_{high} <$ wgt ,

    create **n** particles of weight w_{ave}, $\quad$ where $\quad n \leftarrow \left\lfloor \dfrac{\text{wgt}}{w_{ave}} + \xi \right\rfloor$

Weighted Sampling Example — Effective Free-gas Model for Scatter with Bound Hydrogen

- Given a neutron with initial energy E_0, $E_0 > .625$ eV

- For scattering with **free** hydrogen (target-at-rest), PDF for scatter to E is

$$f_{FREE}(E_0 \rightarrow E) = \frac{1}{E_0}, \qquad 0 \leq E \leq E_0$$

- For scattering with **bound** hydrogen (free-gas), PDF for scatter to E is

down-scatter: $\qquad f_{BOUND}(E_0 \rightarrow E) = \frac{erf\sqrt{E/kT}}{E_0 - kT/2}, \qquad 0 \leq E \leq E_0$

up-scatter: $\qquad f_{BOUND}(E_0 \rightarrow E) = \delta(E - E_0), \qquad E > E_0$

$$P(upscatter) = \frac{kT}{E_0 + kT/2}$$

- Sampling scheme for $\hat{E}, \hat{\mu}$:

First, sample $\hat{E}, \hat{\mu}$ using target-at-rest scattering model.
Then,

$\qquad$ If $\xi < P(upscatter)$, $\qquad$ set $\hat{E} \leftarrow E_0$, $\quad \hat{\mu} \leftarrow 1$, then exit

Otherwise, modify weight by factor

$$\frac{f_{BOUND}\left(E_0 \rightarrow \hat{E}\right)}{f_{FREE}\left(E_0 \rightarrow \hat{E}\right)} = \frac{erf\sqrt{\hat{E}/kT}}{1 - kT/(2E_0)}$$

and set $\hat{\mu} \leftarrow \mu_{BOUND}\left(E_0 \rightarrow \hat{E}\right)$

Random Sampling

- ## Source
    - Fixed sites — uniform PDF + rejection
    - Fission sites — discrete PDF + stratified sampling
    - Energy — piecewise-linear PDF (binary table search + linear PDF)
    - Direction — isotropic 3D PDF

- ## Tracking
    - free-flight distance — exponential PDF
    - delta-tracking — rejection sampling of pseudo- & real collisions

- ## Russian Roulette & Splitting
    - discrete PDF + weights

- ## Collisions
    - Survival biasing — weights
    - Select phase & nuclide — discrete PDF (on-the-fly)
    - Epithermal
        - Scattering angle — equally-probable histograms (uniform discrete PDF + uniform in bin)
        - Inelastic: $\qquad$ energy — discrete PDF (aliased), then uniform within group
            $\qquad\qquad$ n2n — weights
        - modified free-gas — weights
    - Thermal
        - multigroup — discrete PDF for group-to-group (aliased), linear PDF for μ
        - $S(\alpha,\beta)$ — discrete PDF (aliased) & uniform PDF sampling
    - Direction — polar angle from uniform PDF
    - Fission bank — discrete PDF + weights

Random Sampling — References

**Every Monte Carlo code developer who works with random sampling should own & read these references:**

– D. E. Knuth, <u>The Art of Computer Programming, Vol. 2: Semi-numerical Algorithms</u>, 3rd Edition, Addison-Wesley, Reading, MA (1998).

– L. Devroye, <u>Non-Uniform Random Variate Generation</u>, Springer-Verlag, NY (1986).

– J. von Neumann, "Various Techniques Used in Conjunction with Random Digits," *J. Res. Nat. Bur. Stand. Appl. Math Series* **3**, 36–38 (1951).

– C. J. Everett and E. D. Cashwell, "A Third Monte Carlo Sampler," LA9721-MS, Los Alamos National Laboratory, Los Alamos, NM (1983).

– H. Kahn, "Applications of Monte Carlo," AECU-3259, Rand Corporation, Santa Monica, CA (1954).

Computational Geometry

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

4 -1

Engineering Model vs. Computational Model

Model Generation **Large-scale Computation** **Post-processing**

Engineering Model **Computational Model** **Engineering Model**
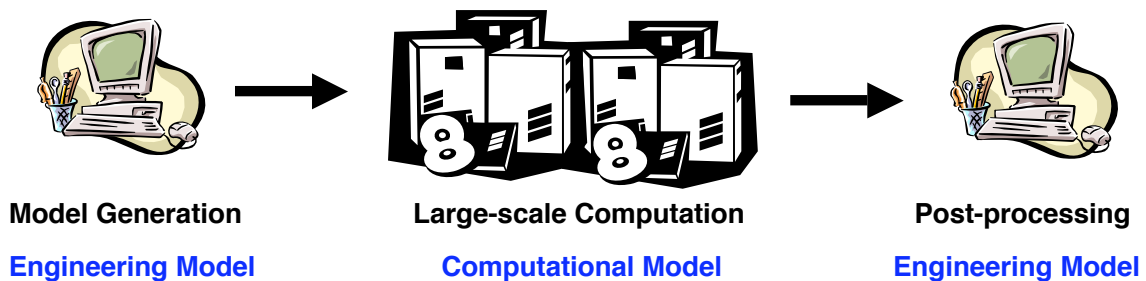
- Model Generation
  - Focus on engineering productivity
  - Describes "reality" to computer
  - Interactive, batch, or CAD

- Large-scale Computation
  - Focus on efficiency & capabilities
  - Data structures should be compact & regular
  - Computational model often hidden from user

- Post-Processing
  - Interpretation of results
  - Visualization

4 -2

Modeling vs. Computation

- Element geometry

- Elements —> assemblies

- Assemblies —> core

- Core + peripherals
  —> 3D model
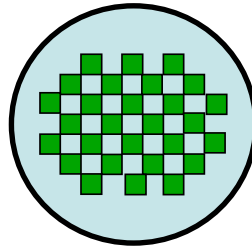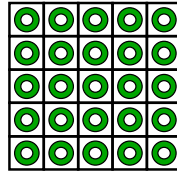
Model construction

Geometry computation

Monte Carlo Geometry

Geometry
- Which cell is particle in?
- What will it hit next?
- How far to boundary?
- What's on other side?
- Survival?

Physics
- How far to collision?
- Which nuclide?
- New E, direction?
- Secondaries?
- Survival?

Tallies
- Tally events of interest
- Compute results
- Compute statistics
- Balances
- Performance stats
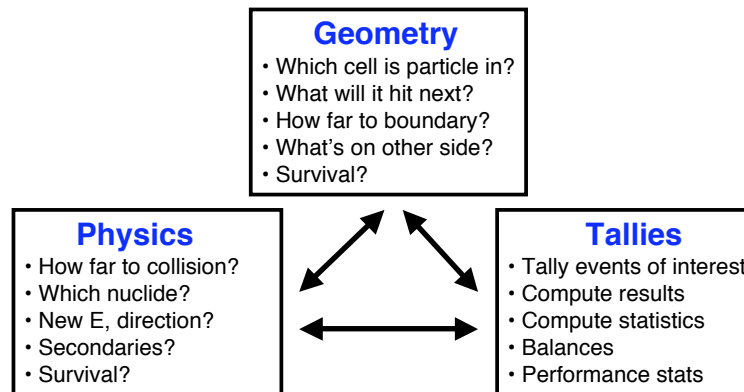
mcnp, rcp, vim, racer, sam-ce, tart, morse, keno, tripoli, mcbend, monk, o5r, recap, andy,.....

Development of particular geometric capabilities is driven by applications:

- **Shielding & experiment analysis**
  - Irregular geometry
  - Moderate number of regions & compositions
- **Reactor core analysis**
  - Regular geometry
  - Very many regions & compositions

Computational Algorithm – Geometric View

Repeat for all **cycles**
.   Repeat for all **histories** in cycle
.   .   Repeat until **collision**
.   .   .   Repeat for each universe **level**
.   .   .   .   Repeat for **surfaces** of 3D region
.   .   .   .   .   **Distance calculation**
.   .   .   .
.   .   .   . . .
.   .   .   Boundary crossing
.   .   .   Neighbor search
.   .   .   Roulette/split
.   .   . . .
.   .   Collision analysis
.   .   Roulette/split
.   . . .
. . .

> **1 reactor calculation requires ~10^{10} distance calculations**

Computational Geometry

- Every point in space that a particle could possibly reach must be defined in the geometry model

- 3D **volumes** are defined by their bounding **surfaces**
  – Boundary representation
  – Combinatorial geometry, with either surfaces or primitive bodies
  – CSG – constructive solid geometry, tree structure with boolean operators
  – Mesh geometry

- A **cell number** is assigned to each 3D volume
  – For some codes, disjoint volumes must have different cell numbers
  – For MCNP & others, disjoint volumes may have the same cell number

- A **material number** is assigned to each cell
  – Composition is assumed to be uniform & homogeneous within cell

- **Tallies** are defined for particular cells or surfaces, reaction types, & estimator types

## Basic Geometry Operations

- **Locate**

  Given a point in space, determine what cell it is in

- **Distance to surface**

  Given a point & direction in a particular cell,
    determine the distance to the next surface of that cell

- **Neighbor search**

  For a particle which has hit a bounding surface of a cell,
    determine the cell to be entered next

- **Boundary conditions**

  For a particle which has hit a cell bounding surface
    declared to be periodic or reflecting,
    determine the new position & direction and cell to be entered next

---

Simple Case – Mesh Geometry

- **Particle**

  Position = (x,y,z),   Direction = (u,v,w)

- **Cell number**

  (i,j,k), indices in mesh

- **Locate**

  i: binary search to find x-interval containing x
  j: binary search to find y-interval containing y
  k: binary search to find z-interval containing z

- **Distance**

  - Use signs of (u,v,w) to select surfaces,
    then compute 3 distances:

    if u>0, $d_x = (x_{i+1}-x)/u$,   otherwise $d_x = (x_i-x)/u$
    … similar for d_y & d_z

  - Distance: $d = min(d_x, d_y, d_z)$

- **Neighbor search**
- **Boundary conditions**
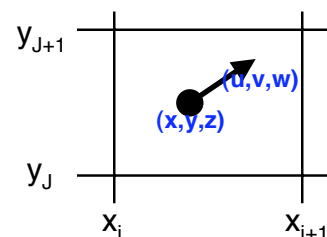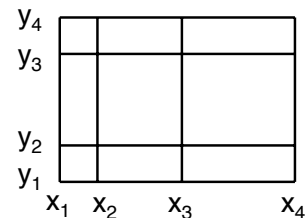
MCNP Geometry

- MCNP uses a "combinatorial geometry" based on surfaces

    – Define **surfaces**

    – Define **cells** using surfaces & operators (intersection, union, complement)

    – Can also group cells together into a **universe**, and embed that universe inside another cell

    – Can also group cells together into a universe, repeat that universe in a **lattice** arrangement, and embed that universe inside another cell

    – Assign **materials** to cells

    – Assign **other properties** to cells   (e.g., importance weights)

    – Define **tallies** using cell or surface numbers

Surfaces

- In MCNP, surface types include:

    1st order: planes
    2nd order: spheres, cylinders, cones, ellipsoid,
                          hyperboloid, paraboloid, general quadric
    4th order: elliptical & circular torus (axes parallel to x, y, or z)

    [see tables on next 2 slides]

- Quadratic polynomial for surface:

    $$F(x,y,z) = ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + jz + k$$

    – Surface is defined by: $F(x,y,z) = 0$
    – Surface is either infinite or closed
    – Normalization convention: factor of leading 2nd order term is positive

MCNP Surfaces

Table 3.1:   MCNP Surface Cards

Mnemonic	Type	Description	Equation	Card Entries
P	Plane	General	$Ax + By + Cz - D = 0$	ABCD
PX		Normal to X-axis	$x - D = 0$	D
PY		Normal to Y-axis	$y - D = 0$	D
PZ		Normal to Z-axis	$z - D = 0$	D
SO	Sphere	Centered at Origin	$x^2 + y^2 + z^2 - R^2 = 0$	R
S		General	$(x-\bar{x})^2 + (y-\bar{y})^2 + (z-\bar{z})^2 - R^2 = 0$	$\bar{x}\,\bar{y}\,\bar{z}\,R$
SX		Centered on X-axis	$(x-\bar{x})^2 + y^2 + z^2 - R^2 = 0$	$\bar{x}\ R$
SY		Centered on Y-axis	$x^2 + (y-\bar{y})^2 + z^2 - R^2 = 0$	$\bar{y}\ R$
SZ		Centered on Z-axis	$y^2 + y^2 + (z-\bar{z})^2 - R^2 = 0$	$\bar{z}\ R$
C/X	Cylinder	Parallel to X-axis	$(y-\bar{y})^2 + (z-\bar{z})^2 - R^2 = 0$	$\bar{y}\,\bar{z}\,R$
C/Y		Parallel to Y-axis	$(x-\bar{x})^2 + (z-\bar{z})^2 - R^2 = 0$	$\bar{x}\,\bar{z}\,R$
C/Z		Parallel to Z-axis	$(x-\bar{x})^2 + (y-\bar{y})^2 - R^2 = 0$	$\bar{x}\,\bar{y}\,R$
CX		On X-axis	$y^2 + z^2 - R^2 = 0$	R
CY		On Y-axis	$x^2 + z^2 - R^2 = 0$	R
CZ		On Z-axis	$x^2 + y^2 - R^2 = 0$	R

MCNP Surfaces

Mnemonic	Type	Description	Equation	Card Entries
K/X	Cone	Parallel to X-axis	$\sqrt{(y-\bar{y})^2 + (z-\bar{z})^2} - t(x-\bar{x}) = 0$	$\bar{x}\,\bar{y}\,\bar{z}\,t^2 \pm 1$
K/Y		Parallel to Y-axis	$\sqrt{(x-\bar{x})^2 + (z-\bar{z})^2} - t(y-\bar{y}) = 0$	$\bar{x}\,\bar{y}\,\bar{z}\,t^2 \pm 1$
K/Z		Parallel to Z-axis	$\sqrt{(x-\bar{x})^2 + (y-\bar{y})^2} - t(z-\bar{z}) = 0$	$\bar{x}\,\bar{y}\,\bar{z}\,t^2 \pm 1$
KX		On X-axis	$\sqrt{y^2 + z^2} - t(x-\bar{x}) = 0$	$\bar{x}\,t^2 \pm 1$
KY		On Y-axis	$\sqrt{x^2 + z^2} - t(y-\bar{y}) = 0$	$\bar{y}\,t^2 \pm 1$
KZ		On Z-axis	$\sqrt{x^2 + y^2} - t(z-\bar{z}) = 0$	$\bar{z}\,t^2 \pm 1$ ± 1 used only for 1 sheet cone
SQ	Ellipsoid Hyperboloid Paraboloid	Axis parallel to X-, Y-, or Z-axis	$A(x-\bar{x})^2 + B(y-\bar{y})^2 + C(z-\bar{z})^2$ $+ 2D(x-\bar{x}) + 2E(y-\bar{y})$ $+ 2F(z-\bar{z}) + G = 0$	A B C D E F G $\bar{x}\ \bar{y}\ \bar{z}$
GQ	Cylinder Cone Ellipsoid Hyperboloid Paraboloid	Axes not parallel to X-, Y-, or Z-axis	$Ax^2 + By^2 + Cz^2 + Dxy + Eyz$ $+ Fzx + Gx + Hy + Jz + K = 0$	A B C D E F G H J K
TX	Elliptical or circular torus. Axis is parallel to X-, Y-, or Z-axis		$(x-\bar{x})^2/B^2 + (\sqrt{(y-\bar{y})^2 + (z-\bar{z})^2} - A)^2/C^2 - 1 = 0$	$\bar{x}\,\bar{y}\,\bar{z}$ A B C
TY			$(y-\bar{y})^2/B^2 + (\sqrt{(x-\bar{x})^2 + (z-\bar{z})^2} - A)^2/C^2 - 1 = 0$	$\bar{x}\,\bar{y}\,\bar{z}$ A B C
TZ			$(z-\bar{z})^2/B^2 + (\sqrt{(x-\bar{x})^2 + (y-\bar{y})^2} - A)^2/C^2 - 1 = 0$	$\bar{x}\,\bar{y}\,\bar{z}$ A B C
XYZP		Surfaces defined by points		See pages 3–15 and 3–17

Sense

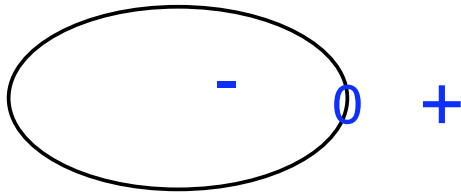- For a given point in space, (x,y,z), and surface equation, F(x',y',z')=0, the **sense** of the point with respect to the surface is defined as:

  Inside the surface,   sense < 0,   if   F(x,y,z) < 0

  Outside the surface,   sense > 0,   if   F(x,y,z) > 0

  On the surface,   sense = 0,   if   F(x,y,z) = 0
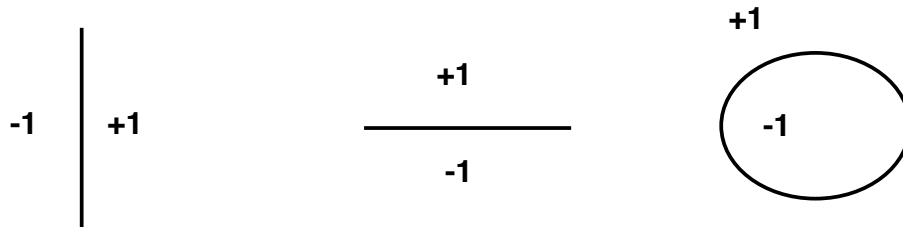
  [Must be careful to consider computer roundoff!]

4 - 13

Sides

- A surface divides space into positive & negative sides

  – MCNP convention: +1 = positive side of surface 1
                  -1 = negative side of surface 1

  – If not sure which side is + or -, pick a point & substitute
    into surface function, F(x,y,z) — see if result is + or -

4 - 14

Intersection of Sides

MCNP convention: +1 -2 == intersection of positive side of surface 1
                                        and negative side of surface 2

+1 -2   +1 +2

-1 -2   -1 +2

Surface 1   Surface 2

Surface 1 ➔

-1 +2
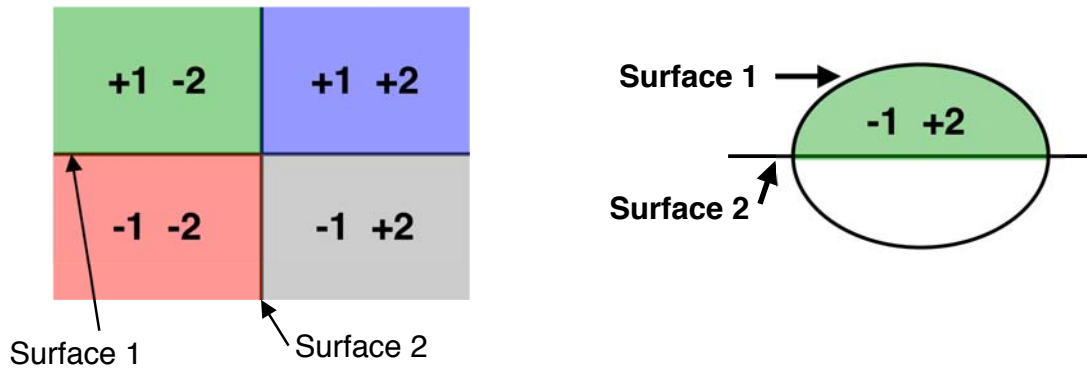
Surface 2

Union of Sides

– MCNP convention: colon signifies a union operator

-1 **:** 2 == union of negative side of surface 1
                    with positive side of surface 2

Surface 1 ➔

-1 : +2
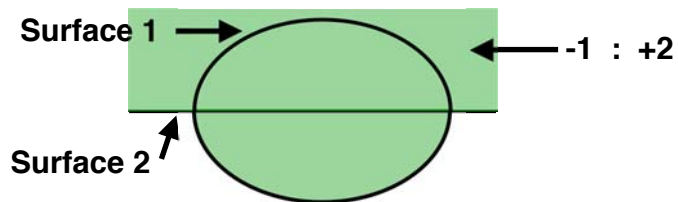
Surface 2

Cells

- • A cell is defined to be the
  - – **Intersection of half-spaces defined by a list of signed surface numbers**

    Example: **cell 1   -5**

                    cell 2   +1 -2 +3 -4 +5
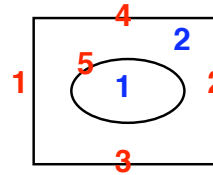
  - – **Union of half-spaces defined by signed surface numbers**

    Example: **cell 1   +1 : -2**

  - – **The complement of another cell   (i.e., volume NOT in other cell)**

    Example: **cell 1   #5**

  - – **A combination of the above**

    Example: **cell 1   (+1 -2) : 3   #5**

Cells

- • Cells do not have to be convex

  (-3 +1) : (-3 -2 -1)

  or

  (-3 +1) : (-3 -2)

  or

  -3 (+1 : -2)
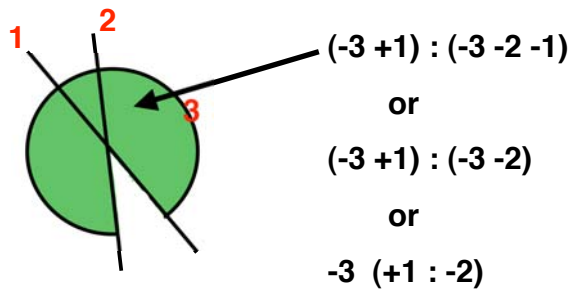
- • Cells may involve discontiguous regions

  Cell 1   -1 : -2 : -3

  Cell 2   #1

  or   1 2 3
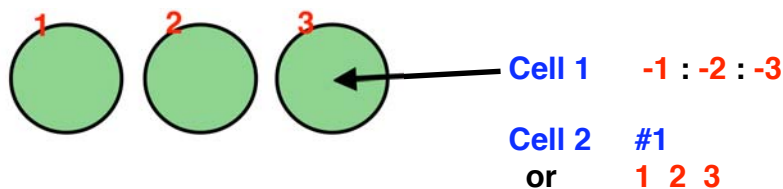
Locate Operations

Given point (x,y,z), determine which cell it is contained in:

```
For( cell = 1 ... n_cells ) {

        Foreach  surf  in  cell {

                Evaluate  S_surf = sign{ F_surf(x,y,z) }

                Does  S_surf  match the sense from the cell definition?
        }

        If all surface-senses for (x,y,z) matched the cell definition,
             then exit & return cell as the result
}
```

Distance Calculation

Given point (x,y,z) in cell I,
     determine the distance to the cell boundary

```
        d <- infinity

        Foreach  surf  in  cell I {

                If  surf is part of the external boundary of cell I {

                        Evaluate  d_surf = smallest positive root of
                                        F_surf( x+du, y+dv, z+dw ) = 0

                        d = min( d, d_surf )
                }
        }
        return the value of  d
```

Neighbor Search

- When a cell boundary is reached, what's on the other side?

Easy case

1   2

Hard case

1   2
       3
       4
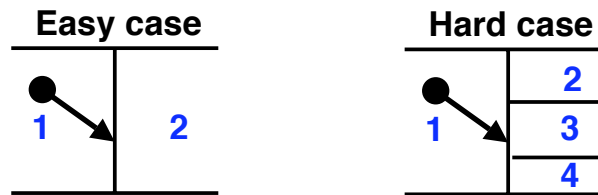
- Most codes build "neighbor lists" during tracking
  - For each bounding surface of cell, remember list of neighbors
  - Initially, neighbor lists are empty
  - Check all cells having surface in common, until one is found satisfying all sense conditions for the particle position
  - Save it
  - Later, check neighbor lists first, only do search if necessary

- Neighbor search is expensive at first, cheap later

- Tracking speeds up as calculation progresses

Embedded Geometry – Universes

- In most real-world applications, there is a need for modeling detailed geometry with many repeating units
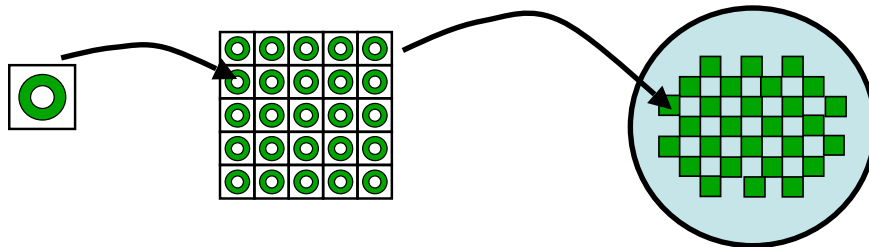
- All production Monte Carlo codes provide capabilities for multiple levels of nested geometry
  - Called "universes" in MCNP
  - A collection of cells may be grouped into a "universe"
  - Universe may be embedded in another cell, with the universe 'clipped' by the cell boundaries

Universes & Lattices

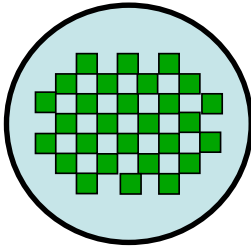Universe 1 – cells for detailed fuel pin
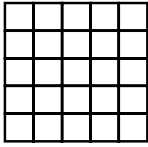
Universe 2 – lattice of cells for fuel assembly
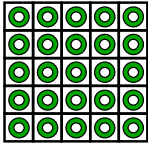
Universe 2, with cells filled by Universe 1
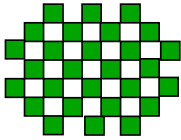
Universe 3 – lattice of cells for reactor

"Real world" – final geometry

Body Geometry

- Some Monte Carlo codes use primitive bodies rather than surfaces for defining cells (e.g., MORSE, KENO, ITS, VIM)

SPH – sphere	ELL – ellipsoid
BOX – box	REC – right elliptic cylinder
RPP – box	RHP – hexagonal prism
RCC – cylinder	HEX – hexagonal prism
WED – wedge	ARB – arbitrary polyhedron
	TRC – truncated cone

- Usually called "combinatorial geometry"
  - Invented by MAGI in ~1956, used in SAM-CE & other codes
  - Space **inside** the body has a **negative** sense, **outside** a **positive** sense
  - Boolean operators AND, OR, NOT may be used to combine bodies (like MCNP's intersection, union, & complement operators)
  - MCNP allows body geometry input (calls them "macrobodies"), but internally converts them to lists of surfaces

Los Alamos
NATIONAL LABORATORY

- **Simple cells** are those which can be constructed using only **intersections**, with no union operators

- Some Monte Carlo codes require that all cells be simple cells. Union operators are not allowed.

- Tracking through simple cells is fast, at the expense of more complex geometry input & setup
  – For simple cells, the logic to find the distance to boundary is simple – check the distance to each of the cell surfaces & keep only the smallest positive distance

---

Los Alamos
NATIONAL LABORATORY
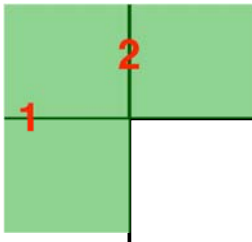
Consider the example at the left.

Using the union operator, the cell is described by: +1 : -2

Without the union operator, separate cells must be defined & then assigned the same material properties:

+1,	-1 -2	
or -2,	+1 +2	
or +1 -2,	+1 +2,	-1 -2

- 3D Surface
  - $F(x,y,z) = 0$
    - Linear: $\nabla F = $ constant
    - Quadratic: $\nabla F = f(x,y,z)$, $\nabla^2 F = $ constant
- Distance calculation
  - **S** = directed distance from (x_0, y_0, z_0) along (u,v,w) to $F(x,y,z)=0$
    = smallest positive root of   **F(x_0+su, y_0+sv, z_0+sw) = 0**

  - General form: **$As^2 + 2Bs + C = 0$, $D = B^2 - AC$**
    - 27 combinations of A, B, C $>0, =0, <0$
    - Only 12 yield valid solutions:

      $s = -C/(2B)$ if   (A=0, C<0, B>0)   or   (A=0, C>0, B<0)

      $s = (-B-\sqrt{D})/A$ if   (A>0, C>0, B<0, D>0) or   (A<0, C>0, B>0, D>0)
                          or (A<0, C>0, B<0, D>0) or   (A<0, C>0, B=0, D>0)

      $s = (-B+\sqrt{D})/A$ if   (A>0, C<0, B>0, D>0) or   (A>0, C<0, B<0, D>0)
                          or (A>0, C<0, B=0, D>0) or   (A<0, C<0, B>0, D>0)
                          or (A>0, C=0, B<0, D>0) or   (A<0, C=0, B>0, D>0)

      $s = \infty$ otherwise

  - Noting that $C = F(x_0,y_0,z_0) = $ sense at (x_0,y_0,z_0),
    the valid solutions can be simplified using the known surface sense §:

      $s' = -C/(2B)$ if   (A=0, D>0)

      $s' = (-B-\sqrt{D})/A$ if   (A≠0, D>0, §>0)

      $s' = (-B+\sqrt{D})/A$ if   (A≠0, D>0, §<0)

      $s' = \infty$ otherwise

  And

      $s = s'$ if s'>0
        $= \infty$ otherwise

Special Topics – Common Surfaces

- If 2 surfaces coincide, neighbor searches become more complicated & tracking can slow down significantly
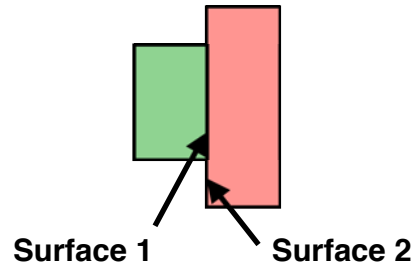
Surface 1   **Surface 2**

- Most MC codes check for coincident surfaces & eliminate one of them (replacing it by the other)

- The tolerance for coincident surfaces usually defaults to a small separation distance (e.g., 1.e-4 cm). For problems with unusual geometry (very small or very large), this may have to be changed in the code or code input.

4 -29

4 -30

Stochastic Geometry
& HTGR Modeling

Forrest B. Brown
X-5, Los Alamos National Laboratory
fbrown@lanl.gov

Introduction

- **Much interest lately in analyzing HTGRs**
  - **Fuel kernels with several layers of coatings**
  - **Very high temperatures**
  - **Contain fission products**
  - **Safety aspects …**

- **Double heterogeneity problem**
  - **Fuel kernels randomly located within fuel elements**
  - **Fuel elements may be "compacts" or "pebbles" (maybe random)**
  - **Challenging computational problem**

- **Monte Carlo codes can faithfully model HTGRs**
  - **Full 3D geometry**
  - **Multiple levels of geometry, including embedded lattices**
  - **Random geometry ?????**

Los Alamos
NATIONAL LABORATORY

~1 mm

Pyrolytic Carbon
Silicon Carbide
Porous Carbon Buffer
Uranium Oxycarbide
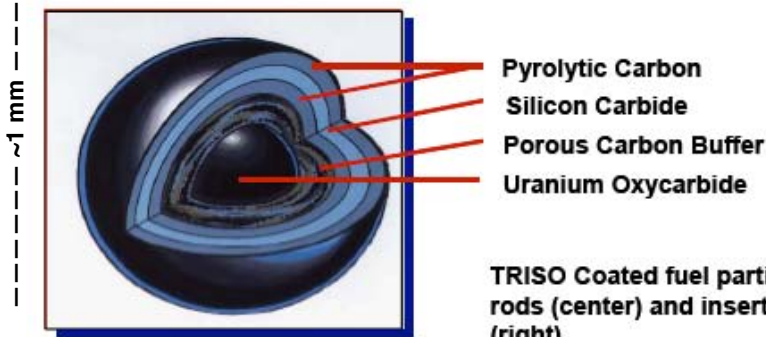
TRISO Coated fuel particles (left) are formed into fuel rods (center) and inserted into graphite fuel elements (right).

PARTICLES   COMPACTS   FUEL ELEMENTS

P. E. MacDonald, et al., "NGNP Preliminary Point Design – Results of the Initial Neutronics and Thermal-Hydraulic Assessments During FY-03", INEEL/EXT-03–00870 Rev. 1, Idaho National Engineering and Environmental Laboratory (2003).

4 -33

Example – GT-MHR Modeling

Los Alamos
NATIONAL LABORATORY

Side permanent reflector
Side replaceable reflector
Active core

core

Fig.3

Inner reflector
Graphite with B$_4$C
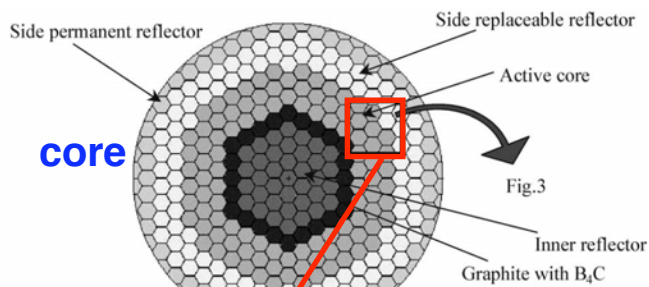
Fig. 2. A cross-section of the GT-MHR homogeneous core with hexagonal structure (HTR1).
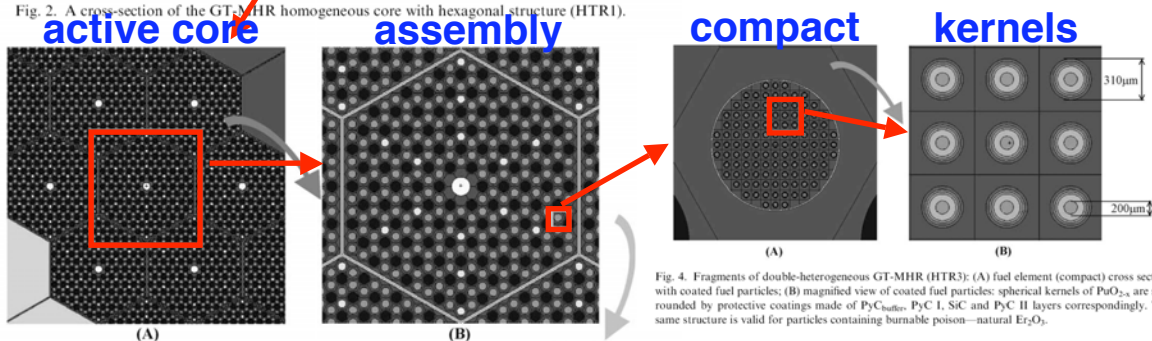
Plukiene, R. and Ridikas, D., Modeling of HTGRs with Monte Carlo: from a homogeneous to an exact heterogeneous core with microparticles. Annals of Nuclear Energy 30, 1573–1585 (2003).

active core   assembly   compact   kernels

310μm

200μm

(A)   (B)   (A)   (B)

Fig. 4. Fragments of double-heterogeneous GT-MHR (HTR3): (A) fuel element (compact) cross section with coated fuel particles; (B) magnified view of coated fuel particles: spherical kernels of PuO$_{2-x}$ are surrounded by protective coatings made of PyC$_{buffer}$, PyC I, SiC and PyC II layers correspondingly. The same structure is valid for particles containing burnable poison—natural Er$_2$O$_3$.

Fig. 3. Fragments of single-heterogeneous GT-MHR (HTR2): (A) an active core structure: three rings of hexagonal fuel columns; (B) magnified view of a separate fuel assembly. Fuel compacts are presented in small grey circles, burnable poison compacts in light grey. Bigger diameter holes stand for He channels, while the rest material represents the graphite matrix.

4 -34

Core

Difilippo, F.C., Monte Carlo Calculations of Pebble Bed Benchmark Configurations of the PROTEUS Facility. Nucl. Sci. Eng. 143, 240–253 (2003).
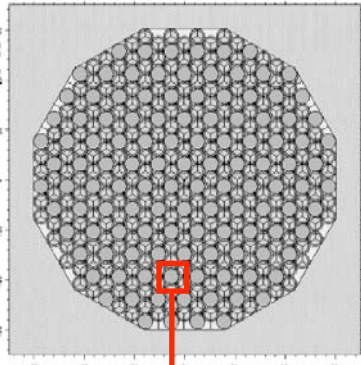
Fig. 3. Cross section of the odd layers of the hcp configurations, case 4 with polyethylene rods. The fi[ ] with the visualization tools of the MCNP program.
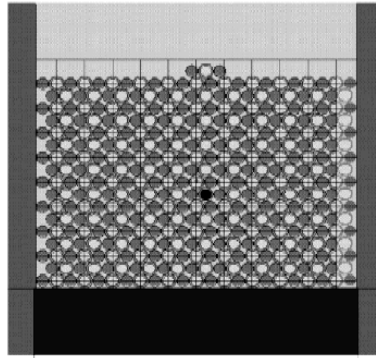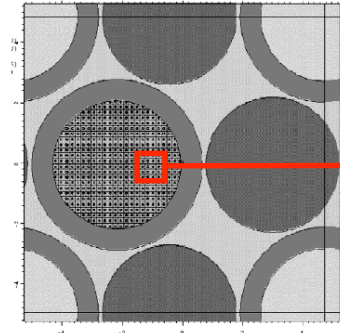
Fig. 5. Case 4, vertical cross section. The black pebble (fuel type) appears magnified in Figs. 6, 7, and 8 to show the heterogeneous details of the fuel pebble.

Pebbles

Fuel kernel lattice

Fuel kernel

Fig. 6. Magnified fuel pebble of Fig. 5.
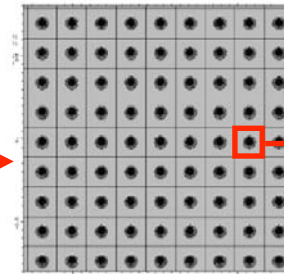
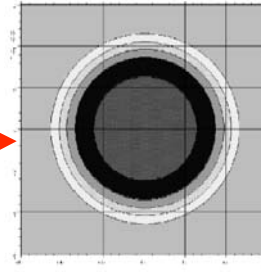Fig. 7. Details of the cubic lattice of fuel kernels inside the pebble of Fig. 5.

Fig. 8. Fuel kernel with the four coatings at each location of the cubic lattice and inside the fuel region of the pebble shown in Fig. 6.

4 -35

MCNP Models for HTGRs

· Los Alamos NATIONAL LABORATORY

- **Existing MCNP geometry can handle:**
  - 3D description of core
  - **Fuel compacts or lattice of pebbles**
    - Typically, **hexagonal lattice** with close-packing of spherical pebbles
    - Proteus experiments: ~ 5,000 fuel pebbles
      ~ 2,500 moderator pebbles

  - **Lattice of fuel kernels** within compact or pebble
    - Typically, **cubic lattice** with kernel at center of lattice element
    - Proteus experiments: ~ 10,000 fuel kernels per pebble
      ~ 50 M   fuel kernels, total

  - Could introduce random variations in locations of a few thousand cells in MCNP input, but **not** a few million.

  - See papers by: Difilippo,   Plukiene et al, Ji-Conlin-Martin-Lee, etc.

4 -36

MCNP5 Stochastic Geometry

- When a neutron enters a new lattice element, a transformation is made to the neutron's position & direction to the local coordinates of the universe embedded in that lattice element. [standard MCNP]

- **Users can flag selected universes as "stochastic"** [new]
  - A neutron entering a lattice element containing a stochastic universe undergoes the normal transformations.
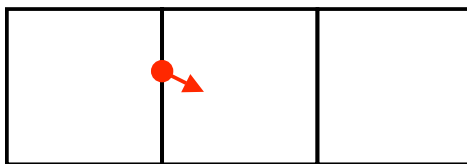
  - Then, additional **random translations** are made:

$$x \leftarrow x + (2\xi_1 - 1) \cdot \delta_x$$
$$y \leftarrow y + (2\xi_2 - 1) \cdot \delta_y$$
$$z \leftarrow z + (2\xi_3 - 1) \cdot \delta_z$$

  - Then, tracking proceeds normally, with the universe coordinates fixed until the neutron exits that lattice element
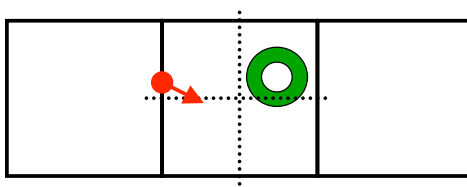
MCNP5 Stochastic Geometry

- Neutron on lattice edge, about to enter embedded universe

- Embedded universe,
 **before** random translation **after** random translation

- Track normally, until neutron exits the lattice element

MCNP5 Stochastic Geometry

- **On-the-fly random translations of embedded universes in lattice**
  - Does not require any extra memory storage
  - Very little extra computing cost - only 3 random numbers for each entry into a stochastic universe

- **For K-effective calculations (KCODE problems)**
  - If fission occurred within fuel kernel, should have source site in next cycle be at same position within fuel kernel
  - Need to save $\delta_x, \delta_y, \delta_z$ along with neutron coordinates in fission bank
  - On source for next cycle, apply $\delta_x, \delta_y, \delta_z$ after neutron pulled from bank

- **To preserve mass exactly**, rather than on the average stochastically, must choose $\delta_x, \delta_y, \delta_z$ so that fuel kernels are not displaced out of a lattice element

maximum δ_x
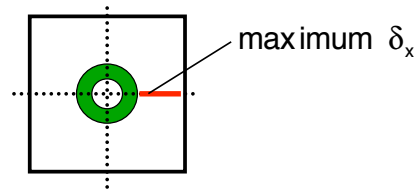
Numerical Results — HTGR Fuel Kernels

- **Infinite array of TRISO fuel kernels in graphite matrix**
  - Fuel kernel geometry & composition taken from the NGNP Point Design (MacDonald et al. 2003)

TRISO Fuel Kernel Geometry and Composition

Region #	Name	Outer radius (μ)	Composition	Density (g/cc)
1	Uranium oxycarbide	175	UCO (UC$^{.5}$O$^{1.5}$)	10.5
2	Porous carbon buffer	275	C	1.0
3	Inner pyrolytic carbon	315	C	1.9
4	Silicon carbide	350	SiC	3.2
5	Outer pyrolytic carbon	390	C	1.9

- **Calculations run 4 ways:**
  1. **Fixed lattice with centered kernels**
  2. **Fixed lattice with random kernels   [MCNP stochastic geometry]**
  3. **Multiple lattice realizations**
  4. **Box of randomly place kernels**
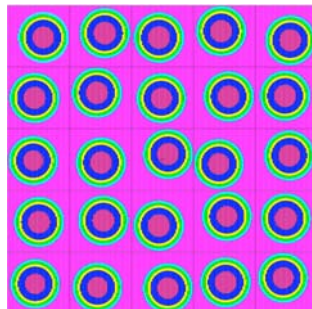
Calculations – Case #1

- **Fixed lattice with centered kernels**
  - 5x5x5 cubical lattice
  - Lattice edge chosen to preserve the specified packing fraction.
  - Fuel kernels centered within the cubical cells
  - Reflecting boundaries on the outer surfaces
  - Essentially same as Difilipo, Plukiene et al, Ji-Conlin-Martin-Lee
  - No random geometry, standard MCNP5 calculations
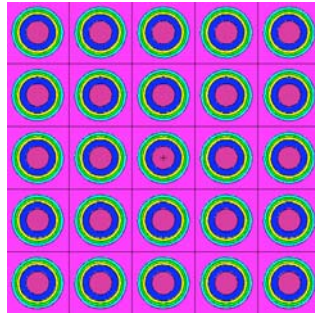
4 -41

Calculations – Case #2

- **Fixed lattice with random kernels   [MCNP stochastic geometry]**
  - 5x5x5 cubical lattice
  - Lattice edge chosen to preserve the specified packing fraction.
  - Fuel kernels randomly placed **on-the-fly** within the cubical cells
  - Reflecting boundaries on the outer surfaces
 - Uses new MCNP5 stochastic geometry

**Fuel kernel displaced randomly
 within lattice element each time
 that neutron enters**

4 -42

Calculations – Case #3

- Multiple lattice realizations
    - 5x5x5 cubical lattice
    - Lattice edge chosen to preserve the specified packing fraction.
    - Fuel kernels randomly placed **in job input** within the cubical cells
    - Reflecting boundaries on the outer surfaces
    - Uses standard MCNP5
   - **25 separate calculations, each with different location of kernels in the input files**
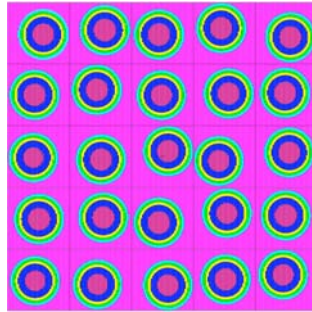
1 realization, fixed lattice with kernel locations chosen randomly in problem input & held constant during each MCNP calculation

4 -43

Calculations – Case #4

- Box of randomly placed fuel kernels
    - Single box with 125 fuel kernels
    - Box size chosen to preserve the specified packing fraction.
    - Fuel kernels randomly placed **in job input** within the box (using RSA algorithm, Random Sequential Addition)
    - Reflecting boundaries on the outer surfaces
    - Uses standard MCNP5
    - **25 separate calculations, each with different location of kernels in the input files**
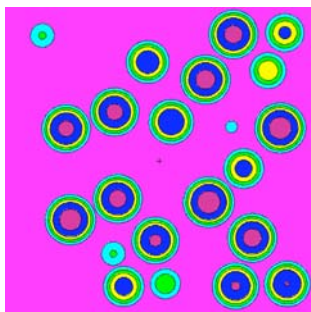
2 different realizations of "truly random" cases:

4 -44

Numerical Results
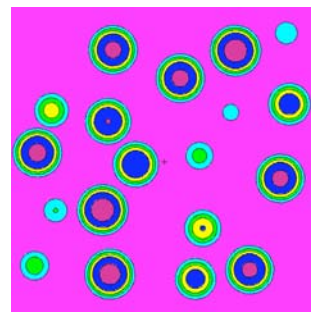
MCNP5 Results for Infinite Lattices of Fuel Kernels

#	Method	K-effective
1	Fixed 5x5x5 lattice with centered spheres	1.1531 ± 0.0004
2	Fixed 5x5x5 lattice with randomly located spheres ("on the fly")	1.1515 ± 0.0004
3	Multiple (25) realizations of 5x5x5 lattice with randomly located spheres	1.1513 ± 0.0004
4	Multiple (25) realizations of randomly packed (RSA) fuel "box"	1.1510 ± 0.0003

Conclusions

- **The new stochastic geometry treatment for MCNP5 provides an accurate and effective means of modeling the particle heterogeneity in TRISOL particle fuel**
  - **Same results as (brute-force) multiple realizations of random geometry input with standard MCNP**
  - **Negligible difference from "truly random" multiple realizations**

- **The results indicate that:**
  - **The neutronic effect of using a fixed lattice is negligible**
  - **The effect of choosing either a centered spheres or randomly located spheres is also small, at least for the specific fuel geometry that was analyzed during this study**

- **Future work**
  - **Examination of finite geometries, including cylindrical fuel compacts, hexagonal fuel blocks, and full core configurations.**
  - **We will also consider lattices other than simple cubic lattices, such as BCC, FCC, and HCP lattices.**

References – HTGR Models & Stochastic Geometry

- Armishaw, M., Smith, N., and Shuttlesworth, E., Particle Packing Considerations for Pebble Bed Fuel Systems. Proc. ICNC 2003 Conference, JAERI-Conf-2003–019, Tokai-mura, Japan (2003).

- Brown, F.B., Sweezy, J.E., and Hayes, R., Monte Carlo Parameter Studies and Uncertainty Analyses with MCNP5. Proc. PHYSOR 2004, Chicago, Illinois (2004).

- Brown, F.B. and Martin, W.R., Stochastic Geometry in MCNP5 for the Analysis of Particle Fuel. Annals of Nuclear Energy, (2004).

- Difilippo, F.C., Monte Carlo Calculations of Pebble Bed Benchmark Configurations of the PROTEUS Facility. Nucl. Sci. Eng. 143, 240–253 (2003).

- Donovan, T. and Danon, Y., Application of Monte Carlo Chord-Length Sampling Algorithms to Transport Through a Two-Dimensional Binary Statistical Mixture. Nucl. Sci. Eng. 143, 226–239 (2003).

- Donovan, T., Sutton, T., and Danon, Y., Implementation of Chord Length Sampling for Transport Through a Binary Stochastic Mixture. Proc. ANS Topical Conf. in Mathematics and Computation, Gatlinburg, TN (2003).

- Donovan, T. and Danon, Y., HTGR Unit Fuel Pebble k-infinity Results Using Chord Length Sampling. Trans. Am. Nucl. Soc. 89, 37–39 (2003).

- Ji, W., Conlin, J., Martin, W.R., and Lee, J.C., Reactor Physics Analysis of the VHTGR Core. Submitted for presentation at the Winter Meeting of the American Nuclear Society (2004).

- Johnson, J.R., Lebenhaft, J.R., and Driscoll, M.J., Burnup Reactivity and Isotopics of an HTGR Fuel Pebble. Trans. Am. Nucl. Soc. 85, 273–274 (2001).

- MacDonald, P.E., et al., NGNP Preliminary Point Design – Results of the Initial Neutronics and Thermal-Hydraulic Assessments During FY-03, INEEL/EXT-03–00870 Rev. 1. Idaho National Engineering and Environmental Laboratory (2003).

- Massimo, L., Physics of High-Temperature Reactors. Pergamon Press (1976).

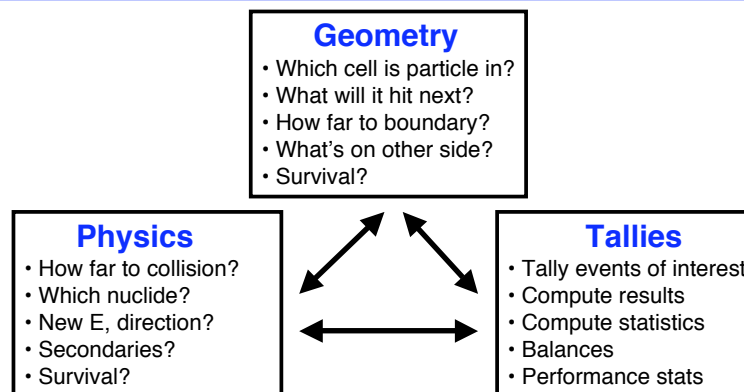## References – HTGR Models & Stochastic Geometry

- MICROX-2, Code System to Create Broad-Group Cross Sections with Resonance Interference and Self-Shielding from Fine-Group and Pointwise Cross Sections, PSR-374. Oak Ridge National Laboratory (1999).

- Mori, T., Okumura, K., and Nagaya, Y., Status of JAERI's Monte Carlo Code MVP. Proc. Monte Carlo 2000 Conference, Lisbon, 625–630 (2000).

- Murata, I., Mori, T., and Nakagawa, M., Continuous Energy Monte Carlo Calculations of Randomly Distributed Spherical Fuels in High-Temperature Gas-Cooled Reactors Based on a Statistical Geometry Model. Nucl. Sci. Eng. 123, 96–109 (1996).

- Murata, I., et al., New Sampling Method in Continuous Energy Monte Carlo Calculation for Pebble Bed Reactors. J. Nucl. Sci. Tech. 34, 734–744 (1997).

- Plukiene, R. and Ridikas, D., Modelling of HTRs with Monte Carlo: from a homogeneous to an exact heterogeneous core with microparticles. Annals of Nuclear Energy 30, 1573–1585 (2003).

- Torquato, S., Random Heterogeneous Materials: Microstructure and Macroscopic Properties, Springer-Verlag (2002).

- Widom, S., Random Sequential Addition of Hard Spheres to a Volume. J. Chem. Phy. 44, 3888–3894 (1966).

- X-5 Monte Carlo Team, MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume I: Overview and Theory, LA-UR-03–1987. Los Alamos National Laboratory (2003).

Collision

Physics

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Monte Carlo Calculations

Geometry
• Which cell is particle in?
• What will it hit next?
• How far to boundary?
• What's on other side?
• Survival?

Physics
• How far to collision?
• Which nuclide?
• New E, direction?
• Secondaries?
• Survival?

Tallies
• Tally events of interest
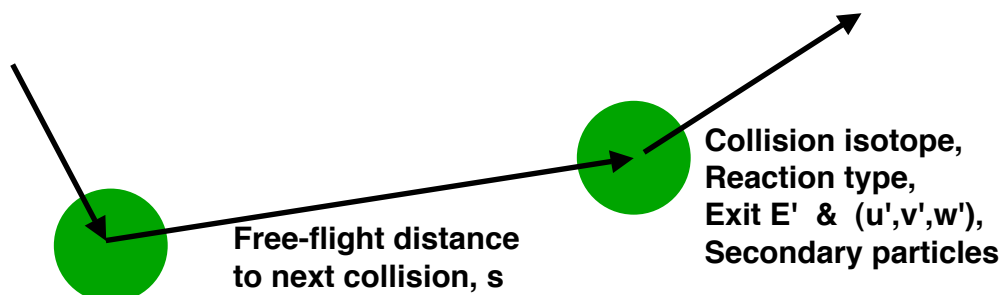• Compute results
• Compute statistics
• Balances
• Performance stats

mcnp, rcp, vim, racer, sam-ce, tart, morse, keno, tripoli, mcbend, monk, o5r, recap, andy,…..

- **Geometry routines** determine the cell & material in that cell

- **Collision routines** model the physical interactions with the material
  - Random sampling from PDFs determined by cross-section data
    - Continuous: flight distance, exit E & direction,   …..
    - Discrete: select nuclide, select interaction type, secondaries, …..

Collision Physics

**Free-flight distance
to next collision, s**

**Collision isotope,
Reaction type,
Exit E' & (u',v',w'),
Secondary particles**

5 -3

Monte Carlo Calculations

- **After a particle emerges from source or collision, or if the particle is on a cell bounding surface:**

  – **Randomly sample the free-flight distance to the next interaction**

  – **If the distance-to-interaction is less than the distance to cell boundary, then move the particle to the interaction point**

  – **Collision physics at the interaction point:**
    - **Determine which isotope the interaction is with**
    - **Determine which interaction type for that isotope**
    - **Determine the energy & direction of the exiting particle**
    - **Determine if secondary particles were produced**
    - **Biasing + weight adjustments**
    - **Tallies of quantities of interest**

5 -4

- **Given a particle at (x_0, y_0, z_0) with direction (u,v,w) in cell I containing material M, sample the free-flight distance to the next interaction**

    - Σ_T = **total macroscopic cross-section** in material M

        = sum$\{ N^j \sigma_T^j \}$, where j = isotopes in material M

        = probability of any interaction per unit distance, units cm^{-1}

    - **PDF for flight distance s**, where $0 \leq s \leq \infty$,

        f(s) = {prob interaction p.u.d} • {prob travelling dist s w/o interact}

        $= \Sigma_T \exp(-\Sigma_T s)$

    - **Sampling procedure**

        $F(s) = 1 - \exp(-\Sigma_T s)$ —> **$s = -\ln(1-\xi) / \Sigma_T$**

        > **We are assuming here that material M is uniform & homogeneous**

- $\displaystyle \Sigma_T = \sum_j N^{(j)} \sigma_T^{(j)}$ where j = isotopes in material M

- **Probability that collision is with isotope j**

    $$p_j = \frac{N^{(j)} \sigma_T^{(j)}}{\displaystyle\sum_k N^{(k)} \sigma_T^{(k)}}$$

- $\{ p_j \}$ = set of discrete probabilities for selecting collision isotope

- $\{ P_j \}$ = discrete CDF, P_j = sum$\{ p_i, i=1,j \}$, $P_0=0$

- **Discrete sampling** for collision isotope **k**

    table search to determine **k** such that $P_{k-1} \leq \xi \leq P_k$

Selecting the Reaction Type

- For collision isotope k,

$$\sigma_T = \sigma_{elastic} + \sigma_{inelastic} + \sigma_{capture} + \sigma_{fission} + \ldots..$$

- $p_j = \sigma_j/\sigma_T$ = probability of reaction type j for isotope k

- { p_j } = set of discrete probabilities for selecting reaction type j

- { P_j } = discrete CDF, P_j = sum{ p_i, i=1,j }, P_0=0

- **Discrete sampling** for reaction type **j**

    table search to determine **j** such that $P_{j-1} \leq \xi \leq P_j$

Selecting the Reaction Type – Modified

- In many applications, **survival biasing** is an effective variance reduction technique
    - Survival biasing is also called **implicit absorption, nonabsorption weighting,** or (loosely) **implicit capture**
    - $\sigma_T = \sigma_{absorption} + \sigma_{scatter}$ (absorption = disappearance)

    - Probability that particle survives collision = $P_{surv} = \sigma_{scatter}/\sigma_T$
    - Probability that particle is absorbed (killed) = $1 - P_{surv}$

- Disallow absorption of particle, & then adjust particle weight to ensure a fair game
    - Tally absorption of wgt•$(1-P_{surv})$
    - Multiply particle weight by P_{surv}
    - When selecting reaction type, **don't consider probability of absorption**

Sampling Exit Energy & Direction

- **Given a collision isotope k & reaction type j, the random sampling techniques used to determine the exit energy and direction, E' and (u',v',w'), depend on**
  - Conservation of energy & momentum
  - Scattering laws – either equations or tabulated data

- **Examples**
  - Isotropic scattering in lab system
  - Multigroup scattering
  - Elastic scattering, target-at-rest
  - Inelastic scattering, MCNP
  - Other collision physics, MCNP

Isotropic Scatter in Lab System

- **Elastic scattering from infinite-mass target nucleus**

  - No change in energy:

    $$E' = E$$

  - Sample direction from isotropic scattering PDF, $f(u',v',w') = 1 / 4\pi$

    $$\phi = 2\pi \, \xi_1$$

    $$u' = 2\xi_2 - 1$$
    $$v' = sqrt(1-u'^2) \cos(\phi)$$
    $$w' = sqrt(1-u'^2) \sin(\phi)$$

Multigroup Scattering

- **Multigroup approach**
  - Divide energy range into intervals (groups)
  - Use average cross-sections for each group,

    σ_{Tg} = total cross-section for group g
  - Use discrete transfer matrix for group-to-group scatter,

 $\sigma_{gg'}$ = cross-section for scatter from group g to group g'

- **Multigroup scattering**
  - For particle with energy E, determine initial energy group g
  - Select exit energy group g' by **discrete sampling** from $\sigma_{gg'}$

$$p_{g'} = \frac{\sigma_{g \to g'}}{\displaystyle\sum_{k=1}^{G} \sigma_{g \to k}}$$

  - Sample exit energy uniformly within bound of group g'
  - Direction
    - For P0 scattering – use procedure for isotropic lab scatter
    - For P1 scattering – sample mu from linear PDF, then select new direction
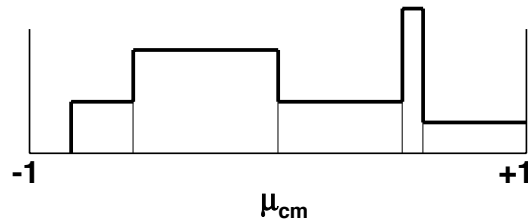   (see next section on elastic scatter)
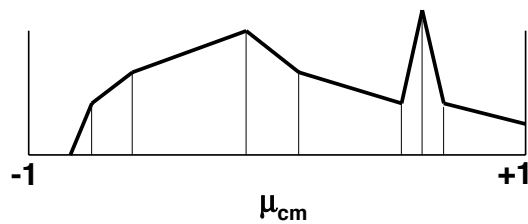

Elastic Scattering, Target-at-rest
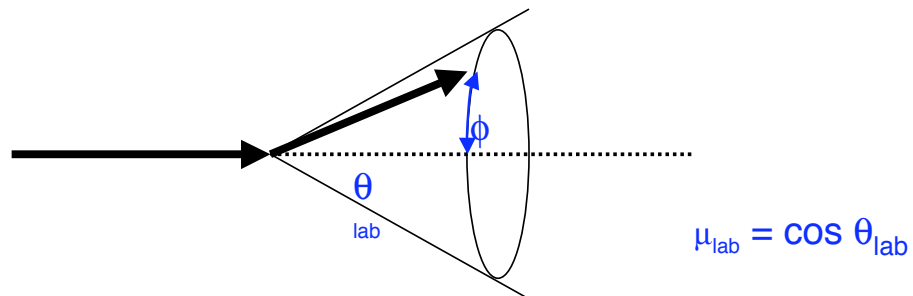
$$\mu_{lab} = \cos \theta_{lab}$$

- Sample μ_{cm} from tabulated PDF data, $f(\mu_{cm})$

- Use kinematics to get E'_{lab} & μ_{lab}

- Sample polar angle ϕ uniformly on $(0, 2\pi)$

- Rotate particle direction using μ_{lab} & ϕ

Sampling the Scattering Direction-cosine, μ_{cm}

- Typical representations for $f(\mu_{cm})$
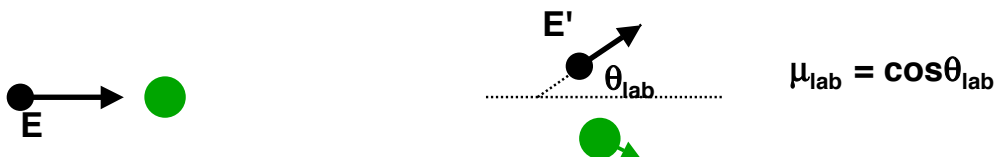
  - Histogram or Equiprobable Histogram PDF

- Piecewise linear PDF

Elastic Scatter – E' & μ_{lab}

- Target-at-rest elastic scatter in lab system – kinematics

$$E' = E \bullet \frac{A^2 + 2A\mu_{cm} + 1}{(A+1)^2}$$

$$\mu_{lab} = \frac{1 + A\mu_{cm}}{\sqrt{A^2 + 2A\mu_{cm} + 1}}$$

Where A = (mass target)/(mass particle)

Exit Direction

- Rotation from (u,v,w) to (u',v',w') using μ_{lab} & ϕ
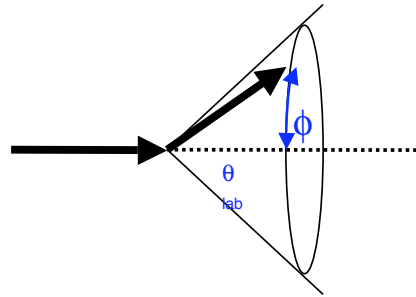
$$\mu = \mu_{lab}$$

$$\phi = 2\pi\xi$$

$$u' = \mu u + \frac{\sqrt{1-\mu^2}(uw\cos\phi - v\sin\phi)}{\sqrt{1-w^2}}$$

$$v' = \mu v + \frac{\sqrt{1-\mu^2}(vw\cos\phi + u\sin\phi)}{\sqrt{1-w^2}}$$

$$w' = \mu w - \sqrt{1-\mu^2}\sqrt{1-w^2}\cos\phi$$

If μ close to 1, special coding may be used to avoid roundoff

Inelastic Scattering – MCNP

- Law 1   ENDF law 1 – Equiprobable energy bins
- Law 2   Discrete photon energies
- Law 3   ENDF law 3 – Inelastic scatter from nuclear levels
- Law 4   ENDF law 4 – Tabular distribution
- Law 5   ENDF law 5 – General evaporation spectrum
- Law 7   ENDF law 7 – Simple Maxwell fission spectrum
- Law 9   ENDF law 9 – Evaporation spectrum
- Law 11   ENDF law 11 – Energy dependent Watt spectrum
- Law 22   UK law 2 – Tabular linear functions of incident energy out
- Law 24   UK law 6 – Equiprobable energy multipliers
- Law 44   ENDF law 1, lang 2, Kalbach-87 correlated energy-angle scatter
- Law 61   ENDF law 11, lang 0,12, or 14 – correlated energy-angle scatter
- Law 66   ENDF law 6 – N-body phase space distribution
- Law 67   ENDF law 7 – correlated energy-angle scatter

Other Collision Physics – MCNP

– Emission from fission
– Delayed neutron emission
– $S(\alpha,\beta)$ scattering for thermal neutrons
– Free-gas scattering for neutrons
– Probability tables for the unresolved resonance energy range for neutrons

– Photoelectric effect
– Pair production
– Compton scattering (incoherent)
– Thomson scattering (coherent)
– Fluorescent emission

– Photonuclear reactions

– Electron interactions – condensed history approach
   • Stopping power, straggling, angular deflections
   • Bremsstrahlung
   • K-shell impact ionization & Auger transitions
   • Knock-on electrons

Secondary Particle Creation

• Consider a collision which results in fission

  $wgt \cdot \nu\sigma_F/\sigma_T$ = expected number of fission neutrons
  produced per collision

• To sample the number of neutrons produced in the collision

  Let $r = wgt \cdot \nu\sigma_F/\sigma_T$

              $n = int[\, r\,]$

  Then,   Produce n fission neutrons with probability 1
              and an additional fission neutron with probability $r-n$

              Assign a weight of r/n to each

  Example: $wgt \cdot \nu\sigma_F/\sigma_T = 1.75$

              If $\xi < .75$, produce 2 neutrons, otherwise produce 1

       or

              Produce $int[\, 1.75 + \xi\,]$ neutrons

Alternative Schemes for Flights/Collisions
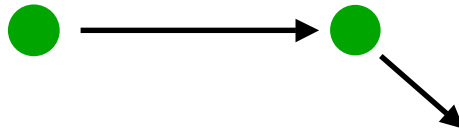
- **Conventional scheme**
    - Particle weight constant during flight
    - Use Σ_T to determine distance-to-collision, $s = -\ln\xi / \Sigma_T$
    - Change weight only on collisions
    - For pathlength absorption estimator, tally $wgt \cdot s \cdot \Sigma_A$
    - Most common scheme for reactors & shielding applications

- **Continuous absorption**
    - Particle weight decreases continuously during flight, due to absorption

    $$wgt(s) = wgt_0 \cdot e^{-\Sigma_A s}$$

    - Use Σ_S to determine distance-to-scattering, $s = -\ln\xi / \Sigma_s$
    - For pathlength absorption estimator, tally $wgt_0 \cdot (1 - e^{-\Sigma_A s})$
    - No absorption in collision
    - Typical use in astrophysics (Implicit Monte Carlo codes)

Random Sampling — Flight Distance

Sampling the free-flight distance, s

- **To simulate the free-flight of particles through the problem geometry, need to randomly sample the distance to collision**
- **PDF for free-flight distance, s, along the current direction:**

$$f(s) = \Sigma_T(s) \cdot \exp\left(-\int_0^s \Sigma_T(x)dx\right)$$

  - **If $\Sigma_T(x)$ is constant within a region, the PDF simplifies to**

$$f(s) = \Sigma_T \cdot \exp\left(-\Sigma_T \cdot s\right)$$

  - **Sampling procedure is then:**

$$\hat{s} \leftarrow \frac{-\ln\xi}{\Sigma_T}$$

- **For multiple regions, can stop particle at each boundary & resample s. Why is this OK ?**
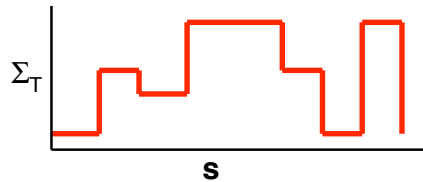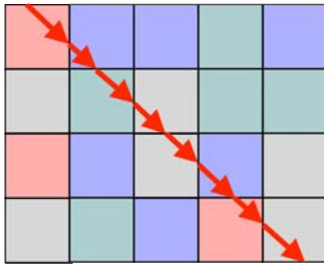 - **Note that prob. of traversing region is** $\quad \mathrm{Prob}\{\hat{s} \geq s\} = 1 - \int_0^s \Sigma_T e^{-\Sigma_T x}dx = e^{-\Sigma_T s}$

  - **For 2 regions, note that** $\quad e^{-\Sigma x_1} \cdot e^{-\Sigma x_2} = e^{-\Sigma(x_1+x_2)}$
    = prob. of traversing <u>both</u> regions

"Regular" Tracking

- **Move particles through one region at a time, until collision occurs**
- **Can be expensive if many regions must be traversed before collision**

Σ_T

s

$$f(s) = \Sigma_T(s) \cdot \exp\left(-\int_0^s \Sigma_T(x)dx\right)$$

- **"Regular" tracking procedure, when Σ_T constant within each region:**
  - **Sample a flight distance, s', using Σ_T for current region:** $\quad s' \leftarrow \dfrac{-\ln\xi}{\Sigma_T}$

  - **If $s' < d_{boundary}$,**   **move particle by s', then analyze the collision**

  - **Otherwise,**   **move particle by $d_{boundary}$, enter next region, repeat until collision occurs**

5 -23

Delta Tracking
- **A type of rejection method for sampling the free-flight distance**
- **Also called Woodcock tracking, fast tracking, or hole tracking**
- **Useful when Σ_T varies rapidly over the flight path**

Σ^*

Σ_T

s
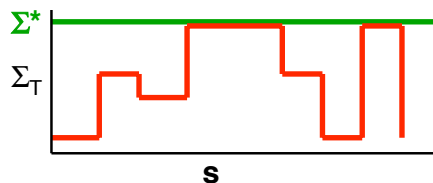
$$f(s) = \Sigma^* \cdot \exp\left(-\Sigma^* \cdot s\right)$$

- **For delta tracking, a fictitious cross-section Σ^* is used, rather than $\Sigma_T(s)$**
  - **Σ^* should be chosen to be $\geq \Sigma_T(s)$ for all possible points along path**
  - **Σ^* may be a function of energy, or region, or not**
  - **$\Sigma^* = \Sigma_T(s) + \Sigma_\delta(s) = $ constant, $\Sigma_\delta(s) \geq 0$ for all s>0**

    where $\Sigma_\delta(s) = $ cross-section for "delta-scattering", i.e., scatter with no change in energy or direction, a fictitious scattering event, or "pseudo-collision"

5 -24

- **For many problems of interest, Σ_T varies within a cell**
  - Charged particle transport – continuous slowing down along the flight path due to interactions with electron field in material
  - Σ_T increases along the flight path

$\Sigma_T(s)$ **Flight distance, s**
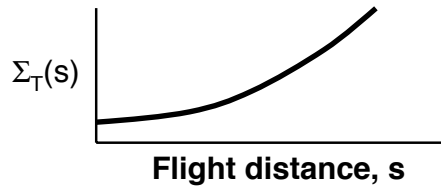
- **For most MC codes, a procedure called delta tracking is used in sampling the free-flight distance**
  - Also called Woodcock tracking, fast tracking, pseudo-collision method, hole tracking, …
  - Involves biased sampling using a larger Σ_T, followed by rejection sampling to assure a fair game

---

Σ^*
$\Sigma_T(s)$ **Flight distance, s**
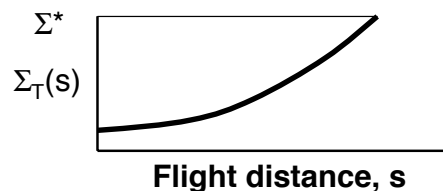
- **Introduce Σ for a "delta" collision**
  - Let $\Sigma^* = \Sigma_T(s) + \Sigma_\delta(s) = $ **constant**,

    where $\Sigma_\delta(s) \geq 0$

    $\Sigma_\delta(s) = $ cross-section for "delta" collision - no change in E, (u,v,w), or wgt

    $\Sigma_* \geq \Sigma_T(s)$

  - $\Sigma_T(s) / \Sigma^* = $ probability of a "real" collision
  - $\Sigma_\delta(s) / \Sigma^* = $ probability of a "delta" collision

- **Basic idea:** Sample flight distance using Σ^*, then reject collision point if $\xi > \Sigma_T(s) / \Sigma^*$

- Using Σ^* rather than $\Sigma_T(s)$ gives an interaction probability per unit distance that is too large, hence a flight distance that is too short. Rejection scheme compensates for this.

- **Sampling procedure**
  - Sample s' from $f(s) = \Sigma^* \exp(-\Sigma^* s)$: $s' = -\ln(1-\xi_1)/\Sigma^*$
  - Move the particle a distance s'
  - if $\xi_2 < \Sigma_T(s')/\Sigma^*$, "real" collision: do collision physics
    otherwise, "delta" collision: no change in E, (u,v,w), wgt
  - Repeat until a real collision occurs

- **Delta tracking can be effective if Σ^* is not too different from the "average" $\Sigma_T(s)$**

- **Delta tracking can be ineffective if $\Sigma^* \gg \Sigma_T(s)$ for most values of s, so that sampling efficiency is low**

- **Delta tracking is also frequently used for tracking through reactor fuel assemblies, where the geometry is a regular lattice.**

Proof: **Delta tracking is an unbiased method for sampling the free-flight distance**

Consider the probability of traversing a distance **s** along the flight path without undergoing a (real) collision, **P(s)**

- $\Sigma^* = \Sigma_T(s) + \Sigma_\delta(s) =$ **constant**, $\Sigma^* \geq \Sigma_T(s)$ and $\Sigma_\delta(s) \geq 0$ for all s>0

- For convenience, define optical thickness for real & delta scatter:

$$\tau(s) = \int_0^s \Sigma_T(x)dx \qquad \tau_\delta(s) = \int_0^s \Sigma_\delta(x)dx$$

Note that, by definition,

$$\Sigma^* s = \tau(s) + \tau_\delta(s), \qquad \Sigma^* s \geq \tau(s)$$

For a particular flight, there could be <u>exactly</u> 0, 1, 2, …, ∞ delta-collisions before a real collision occurs

Let $P(s|n)$ = probability of traversing distance s along the flight path with <u>exactly</u> n delta collisions

Then,

$$P(s) = \sum_{n=0}^{\infty} P(s|n)$$

$$P(s|0) = e^{-\Sigma^* s}$$

$$P(s|1) = \int_0^s P(x|0)\Sigma_\delta(x)P(s-x|0)dx = \int_0^s e^{-\Sigma^* x}\Sigma_\delta(x)e^{-\Sigma^*(s-x)}dx = \tau_\delta(s)e^{-\Sigma^* s}$$

$$P(s|2) = \int_0^s P(x|1)\Sigma_\delta(x)P(s-x|0)dx = \int_0^s \tau_\delta(x)e^{-\Sigma^* x}\Sigma_\delta(x)e^{-\Sigma^*(s-x)}dx$$

$$= \int_0^s \tau_\delta(x)\Sigma_\delta(x)e^{-\Sigma^* s}dx = \frac{[\tau_\delta(x)]^2}{2}e^{-\Sigma^* s}$$

$$P(s|n) = \int_0^s P(x|n-1)\Sigma_\delta(x)P(s-x|0)dx = \frac{[\tau_\delta(s)]^n}{n!}e^{-\Sigma^* s}$$

Then, the total probability of traversing a distance s without undergoing a (real) collision is

$$P(s) = \sum_{n=0}^{\infty} P(s|n) = \sum_{n=0}^{\infty} \frac{[\tau_\delta(s)]^n}{n!}e^{-\Sigma^* s} = e^{\tau_\delta(s)-\Sigma^* s} = e^{-\tau(s)} = \exp\left(-\int_0^s \Sigma_T(x)dx\right)$$

**This is the correct result, identical to the normal sampling of the flight path (without delta tracking)**

Varying Material Properties

For many problems of interest, Σ_T varies within a cell

- **Charged particle transport**
  - **Continuous slowing down along the flight path due to interactions with electron field in material**
  - **Σ_T increases along the flight path**

  $\Sigma_T(s)$

  Flight distance, s
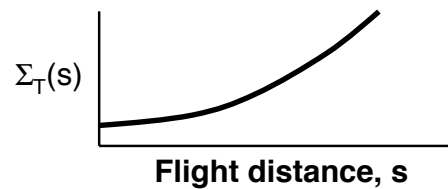
- **Atmospheric transport**
  - **Air density varies with altitude**

  $\rho(h)$

  Altitude, h
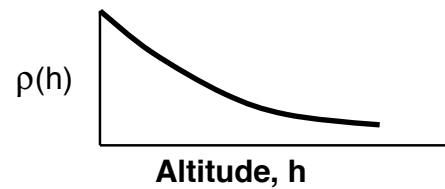
- **Depleted reactor**
  - **Fuel & poison distribution varies due to burnup**

  ρ^{B10}

  Radius in control rod, r

Varying Material Properties
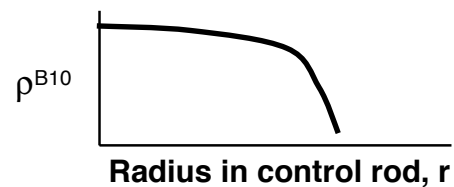
Conventional techniques for handling varying material properties:

- **Stepwise approximation**
  - **Subdivide geometry**
  - **Constant material properties within each step**

$\Sigma_T(s)$
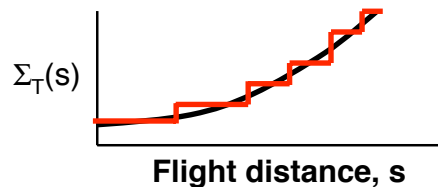
Flight distance, s

- **Woodcock tracking**
  - **Also called delta tracking, fast tracking, pseudo-collision method, hole tracking, …**
  - **Involves biased sampling the flight distance using a larger Σ_T, followed by rejection sampling to assure a fair game**

Sampling the flight distance in varying media

- Optical depth along flight path

$$\tau(s) = \int_{x}^{x+s} \Sigma_T(x')dx' \qquad \Sigma T(x) \text{ is finite,} \quad \Sigma T(x) \geq 0$$

  - Note that
  $$\frac{d\tau(s)}{ds} = \Sigma_T(x+s), \qquad 0 \leq \frac{d\tau}{ds} \leq \infty$$

- To explicitly allow for the case of no collision,
  - PNC = probability of no collision

$$P_{NC} = e^{-\tau(\infty)}$$

- Probability density function (pdf) for the flight distance s:

$$f(s) = P_{NC} \cdot \delta(s = \infty) \;+\; (1 - P_{NC}) \cdot \frac{1}{G}\frac{d\tau}{ds}e^{-\tau(s)}$$

  - Where $\quad G = \int_{0}^{\infty} \frac{d\tau(s)}{ds}e^{-\tau(s)}ds = 1 - e^{-\tau(\infty)} = 1 - P_{NC}$

Sampling the flight distance in varying media

- Random sampling of the Monte Carlo free-flight path requires solving the following equation for s, the flight path:

$$\xi = \int_0^s f(x)dx$$

or

$$\xi = P_{NC} \cdot H(s,\infty) \; + \; (1 - P_{NC}) \cdot \frac{1}{G} \cdot \left(1 - e^{-\tau(s)}\right)$$

- Common case: Σ_T independent of x

$$\tau(s) = \Sigma_T \cdot s, \quad \frac{d\tau}{ds} = \Sigma_T, \quad P_{NC} = 0, \quad G = 1, \quad f(s) = \Sigma_T \cdot e^{-\Sigma_T \cdot s}$$

  – With solution:

$$s = -\frac{\ln(1 - \xi)}{\Sigma_T}$$

Sampling the flight distance in varying media

Direct Numerical Sampling for the free-flight distance:

Step [1]

   If $\xi < PNC$,   Then: No collision, set s=∞, exit

                              Otherwise: Do Steps 2 & 3

Step [2]

   Define $\hat{\tau} = \tau(s)$

   Sample $\hat{\tau}$ by solving $\xi = \frac{1}{G} \int_0^{\hat{\tau}} e^{-\tau} d\tau, \quad with \; 0 \le \hat{\tau} \le \tau(\infty)$

   That is, sample from a truncated exponential PDF:

$$\hat{\tau} = -\frac{\ln(1 - \xi \cdot G)}{\Sigma_T}$$

Step [3]

   Solve for s: $\hat{\tau} = \tau(s) = \int_0^s \Sigma_T(x + s')ds'$

   Analytic solution if possible, otherwise use Newton iteration

- **Newton iteration to numerically solve for s:**

$$s_0 = \hat{\tau} / \Sigma_T(x_0)$$

$$n = 0$$

$$Iterate:$$

$$n = n+1$$

$$g = \hat{\tau} - \tau(s_{n-1})$$

$$g' = dg/ds = -\Sigma_T(x_0 + s_{n-1})$$

$$s_n = s_{n-1} - g/g'$$

$$Stop\ if\quad |s_n - s_{n-1}| < \varepsilon$$

- Notes:
  - Because g'<0, g(s) is monotone & there can be <u>only one root</u>
  - For cases where Σ_T>0, Newton iteration <u>guaranteed to converge</u>
  - If $\Sigma_T(x)$=0 or very small, g' may be 0, leading to numerical difficulties
  - Remedied by combining Newton iteration with bisection if g' near zero
  - Typically only 1–5 iterations needed to converge s to within 10–6

- Represent material density by high-order, orthogonal polynomial expansion within each cell
  - Legendre polynomial representation for material density in cell

$$\rho(x) = \sum_{n=0}^{N} \frac{2n+1}{2} \cdot a_n \cdot P_n\left[\frac{2}{\Delta x}(x - x_{min}) - 1\right]$$

$$a_n = \frac{2}{\Delta x} \int_{x_{min}}^{x_{max}} \rho(x) P_n\left[\frac{2}{\Delta x}(x - x_{min}) - 1\right] dx$$

- Sample the free-flight distance to next interaction using a direct numerical sampling scheme (Brown & Martin)

$$\Sigma(x) = \frac{\rho(x)}{\rho_0} \cdot \Sigma_0, \qquad \tau(s) = \frac{\Sigma_0}{\rho_0} \cdot \int_{x}^{x+s} \rho(x') \frac{dx'}{\mu}$$

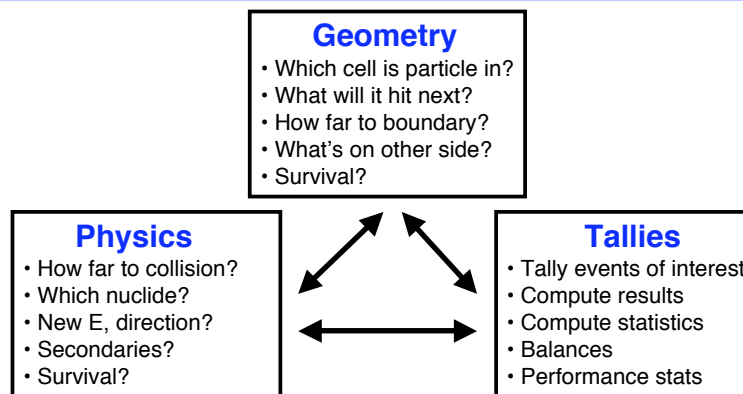 - Use Newton iteration to solve nonlinear equation for flight path

- FB Brown, D Griesheimer, & WR Martin, "Continuously Varying Material Properties and Tallies for Monte Carlo Calculations", PHYSOR-2004, Chicago, IL (April, 2004)

- FB Brown & WR Martin, "Direct Sampling of Monte Carlo Flight Paths in Media with Continuously Varying Cross-sections", ANS Mathematics & Computation Topical Meeting, Gatlinburg, TN (April, 2003).

- DP Griesheimer & WR Martin, "Estimating the Global Scalar Flux Distribution with Orthogonal Basis Function Expansions", Trans. Am. Nucl. Soc. 89 (Nov, 2003)

- DP Griesheimer & WR Martin, "Two Dimensional Functional Expansion Tallies for Monte Carlo Simulations," PHYSOR-2004, Chicago, IL (April, 2004)

- ER Woodcock, T Murphy, PJ Hemmings, TC Longworth, "Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry," *Proc. Conf. Applications of Computing Methods to Reactor Problems*, ANL-7050, p. 557, Argonne National Laboratory (1965).

- LL Carter, ED Cashwell, & WM Taylor, "Monte Carlo Sampling with Continuously Varying Cross Sections Along Flight Paths", *Nucl. Sci. Eng.* **48**, 403–411 (1972).

- J. Spanier, "Monte Carlo Methods for Flux Expansion Solutions of Transport Problems," Nucl. Sci. Eng., **133**, 73 (1999).

Tallies
&
Statistics

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

6 -1

Monte Carlo Calculations

Geometry
- Which cell is particle in?
- What will it hit next?
- How far to boundary?
- What's on other side?
- Survival?

Physics
- How far to collision?
- Which nuclide?
- New E, direction?
- Secondaries?
- Survival?

Tallies
- Tally events of interest
- Compute results
- Compute statistics
- Balances
- Performance stats

mcnp, rcp, vim, racer, sam-ce, tart, morse, keno, tripoli, mcbend, monk, o5r, recap, andy,…..

- **During a history, tally the events of interest**

- **Upon completing a history, accumulate total scores & squares**

- **After completing all histories, compute mean scores & standard deviations**

6 -2

Monte Carlo Estimates of Integrals

Given a function R(x), where x is a random variable with PDF f(x),

- Expected value of R(x) is $\mu = \int R(x)\, f(x)\, dx$

- Variance of R(x) is

$$\sigma^2 = \int R^2(x)\, f(x)\, dx - \mu^2$$

Monte Carlo method for estimating μ

- make N random samples $\hat{x}_j$ from f(x)
- Then

$$\overline{R} \approx \frac{1}{N} \sum_{j=1}^{N} R(\hat{x}_j)$$

- Central Limit Theorem states that for large N, the PDF of $\overline{R}$ approaches a Gaussian distribution
- That is, if the Monte Carlo problem is repeated, $\overline{R}$ will be normally distributed

Laws of Large Numbers

Let $x_1, x_2, \ldots, x_N$ be a sequence of independent, identically distributed random variables each with a finite mean $E[x_j]=\mu$ and let

$$\overline{x}_N = \frac{1}{N} \sum_{j=1}^{N} x_j$$

- **Weak Law of Large Numbers**

  For any $\varepsilon > 0$

$$\lim_{N \to \infty} P(|\overline{x}_N - \mu| > \varepsilon) = 0$$

  Tells how a sequence of probabilities converges

- **Strong Law of Large Numbers**

$$P\left(\lim_{N \to \infty} |\overline{x}_N - \mu| > \varepsilon \right) = 0$$

  Tells how the sequence of IID random variables behaves in the limit

Central Limit Theorem

- **Central Limit Theorem**

$$\lim_{N\to\infty} \mathbf{Prob}\left\{ \mu - a\,\frac{\sigma}{\sqrt{N}} \;\leq\; \overline{x} \;\leq\; \mu + b\,\frac{\sigma}{\sqrt{N}} \right\} = \frac{1}{\sqrt{2\pi}} \int_{-a}^{b} e^{-t^2}\,dt$$

   – If $a = b = 1$, $\mathbf{Prob}\left\{ \mu - \frac{\sigma}{\sqrt{N}} \;\leq\; \overline{x} \;\leq\; \mu + \frac{\sigma}{\sqrt{N}} \right\} = 68\%$

        Note: 32% of the time, $\overline{x}$ should be <u>outside</u> range $\mu \pm \dfrac{\sigma}{\sqrt{N}}$

   – If $a = b = 2$, $\mathbf{Prob}\left\{ \mu - \frac{2\sigma}{\sqrt{N}} \;\leq\; \overline{x} \;\leq\; \mu + \frac{2\sigma}{\sqrt{N}} \right\} = 95\%$

        Note: 5% of the time, $\overline{x}$ should be <u>outside</u> range $\mu \pm \dfrac{2\sigma}{\sqrt{N}}$

Tallies & Statistics

- For a given history, tally events of interest
  - Example – surface crossings
    - For each particle crossing surface A, accumulate the weight each time a particle crosses that surface
    - A particular particle may cross the surface more than once
    - Progeny of that particle (e.g., another particle created by splitting) may also cross that surface one or more times

- When the history is complete, add the score & score^2 to accumulators for the problem

  $S1_{problem} = S1_{problem} + (S_{history})$
  $S2_{problem} = S2_{problem} + (S_{history})^2$

- When all N histories are complete, compute final mean score & standard deviation

$$\mathbf{mean\ score} \;=\; \frac{1}{N} \bullet S1$$

$$\mathbf{std\ dev\ of\ mean} \;=\; \sqrt{\frac{1}{N-1}\left[\frac{S2}{N} - \left(\frac{S1}{N}\right)^2 \right]}$$

- Given a set of random samples, $x_1, x_2, \ldots, x_N,$

  - Mean

  $$\overline{x} = \frac{1}{N}\sum_{j=1}^{N} x_j$$

  - Population variance

  $$\sigma^2 = \frac{1}{N}\sum_{j=1}^{N} x_j^2 - \left(\frac{1}{N}\sum_{j=1}^{N} x_j\right)^2 = \frac{1}{N}\sum_{j=1}^{N} x_j^2 - \overline{x}^2$$

  - Variance **of the mean**

  $$\sigma_{\overline{x}}^2 = \frac{\sigma^2}{N}$$

- **Tallies can be made for selected events & portions of phase space:**

  - Range of energies, $E_1 - E_2$
  - Range of particle times, $t_1 - t_2$
  - Specified cells
  - Specified surfaces
  - Specified range of $n \cdot \Omega$ for surface crossings
  - Specified reaction cross-sections Σ_x
  - Secondary particle production
  - Energy deposited in cell
  - Conditional events, e.g., absorption in cell B due to source in cell A
  - Energy of neutrons causing fission
  - Scattering from energy range E_1-E_2 to range E_3-E_4
  - Etc.

- **Angular flux**

$$\Psi(\mathbf{r}, E, \Omega)$$

- **Flux**

$$\phi(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{4\pi} d\Omega \, \Psi(\mathbf{r}, E, \Omega)$$

  – Scalar quantity
  – Total distance traveled by all particles in a cm^3 per second
  – Units: distance / cm^3-sec   =   1 / cm^2-sec

- **Current**
  – Number of particles crossing surface per second per unit area
  – Units: 1 / cm^2-sec
  – Partial current: in + or – direction only, J^+ or J^-
  – Net current = $J = J^+ - J^-$

$$\mathbf{J}(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{4\pi} d\Omega \, \vec{n} \bullet \Omega \, \Psi(\mathbf{r}, E, \Omega)$$

$$J^+(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{\vec{n}\bullet\Omega>0} d\Omega \, \vec{n} \bullet \Omega \, \Psi(\mathbf{r}, E, \Omega) \qquad J^-(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{\vec{n}\bullet\Omega<0} d\Omega \, \vec{n} \bullet \Omega \, \Psi(\mathbf{r}, E, \Omega)$$

- For a particular reaction "x"

$$R_X(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{4\pi} d\Omega \, \Psi(\mathbf{r}, E, \Omega) \, \Sigma_X(\mathbf{r}, E)$$

  – Reactions per cm^3 per sec

- Collision density

$$C(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{4\pi} d\Omega \, \Psi(\mathbf{r}, E, \Omega) \, \Sigma_T(\mathbf{r}, E)$$

- Energy deposition (average per collision)

$$E_{deposited}(\mathbf{r}) = \int\limits_{E_1}^{E_2} dE \int\limits_{4\pi} d\Omega \, \Psi(\mathbf{r}, E, \Omega) \, \Sigma_T(\mathbf{r}, E) \, K(\mathbf{r}, E)$$

where $K(\mathbf{r}, E) = $ average E deposited per collision

Analog vs. Weighted Monte Carlo

- **Analog Monte Carlo**
  - Faithful simulation of particle histories
  - No alteration of PDFs (I.e., no biasing or variance reduction)
  - At collision, particle is killed if absorption occurs
  - Particle is born with weight = 1.0
  - Weight unchanged throughout history until particle is killed
  - Score 1.0 when tallying events of interest

- **Weighted Monte Carlo (non-analog)**
  - Alter the PDFs to favor events of interest
  - Particle is born with weight = 1.0
  - Weight is altered if biased PDF is used
  - Typically, particle always survives collision & weight is reduced by P_{surv}
  - Weight can also be changed by Russian roulette/splitting & other variance reduction techniques
  - Score **wgt** when tallying events of interest

Tally Types

- **Current tallies**
  - Surface crossing estimator

- **Flux tallies**
  - Pathlength estimator
  - Collision estimator
  - Surface crossing estimator
  - Next event estimator (point detector)

- **Reaction rate tallies**
  - Any of the above flux estimators times a cross-section

- **Energy deposition tallies**
  - Any of the above flux estimators times Σ_T times energy deposited per collision

Current Tallies

- For each particle crossing surface, tally the particle weight

- Divide by total starting weight & surface area to get current

$$J = \frac{1}{W\,A} \sum_{\substack{\text{all} \\ \text{particles} \\ \text{crossing} \\ \text{surface}}} wgt_j$$

W = total starting weight

A = surface area
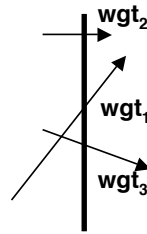
- Typically, keep separate tally for outward partial current for each surface of a cell

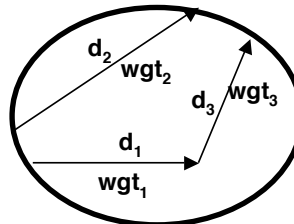- Can get net current by combining partial currents

---

Flux Tally – Pathlength

- For each particle flight within a cell, tally (pathlength*weight)

- Divide by cell volume & total starting weight to get flux estimate

$$\phi = \frac{1}{W\,V} \bullet \sum_{\substack{\text{all} \\ \text{particle} \\ \text{flights} \\ \text{in cell}}} d_j \bullet wgt_j$$

W = total starting weight

V = cell volume
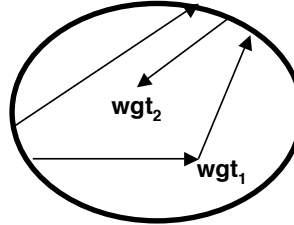
Flux Tally – Collisions

- Since $(\Sigma_T \phi)$ is collision rate, for each collision, tally (wgt/Σ_T) to estimate flux
- Divide by total starting weight & cell volume

$$\phi = \frac{1}{W\,V} \bullet \sum_{\substack{\text{all} \\ \text{collisions} \\ \text{in cell}}} \frac{wgt_j}{\Sigma_T(E_j)}$$

wgt_j = weight of particle **entering** collision

W = total starting weight

V = cell volume

6 -15

Flux Tally – Surface Crossing

- Consider particles crossing a surface
  - Put a "box" of thickness **a** around the surface
  - Pathlength estimate of flux in the box

$$\phi = \frac{1}{W\,aA} \bullet \sum_{\substack{\text{all} \\ \text{particles} \\ \text{crossing} \\ \text{surface}}} wgt_j \bullet \frac{a}{|\mu_j|}$$

where $\mu_j = \cos\theta_j$

  - Note that **a** cancels out
  - Take the limit as **a**—>0
- Surface crossing estimate of flux

$$\phi = \frac{1}{W\,A} \bullet \sum_{\substack{\text{all} \\ \text{particles} \\ \text{crossing} \\ \text{surface}}} \frac{wgt_j}{|\mu_j|}$$

where $\mu_j = \vec{\Omega}_j \bullet \vec{S}$

6 -16
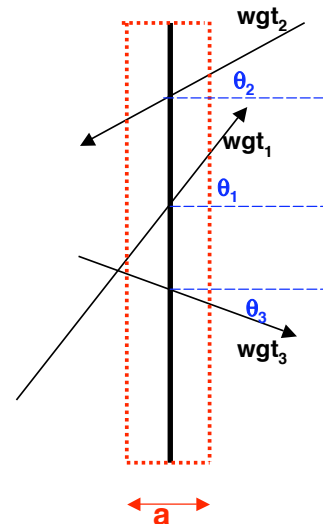
## Flux Tally – Surface Crossing

- Complication: wgt_j/μ_j can be very large for small μ_j
  - Usual solution, based on theory from FH Clark, "Variance of Certain Flux Estimators Used in Monte Carlo Calculations", Nucl.Sci. Eng. **27**, 235–239 (1967)

  - For small |μ|, that is, -ε<μ<ε, (where ε is small), if it is assumed that the flux is only isotropic or linearly anisotropic, then the expected value of 1/|μ| is 2/ε.

- Actual tally procedure:
  - If |μ|<ε, then replace |μ| by ε/2 to score an expected flux.
  - This results in a reliable variance, without affecting the flux estimate significantly.

- MCNP uses ε=.1. Many other codes use ε=.01

Flux at a Point

- Instead of estimating flux for a cell or surface, it may be useful to estimate flux at a point
  - Probability of a history trajectory going through a particular point is zero

- Use a "next event estimator" to get flux at a point
  - Regardless of the actual outcome of simulating a collision, estimate what would happen if the particle scattered exactly in the direction of a point detector

$$\text{Expected } \phi \text{ score} = \text{wgt}' \bullet \frac{p_{sc}(\mu)}{2\pi R^2} \bullet \exp\left\{-\int_0^R \Sigma_T(E')ds\right\}$$

Actual path after collision
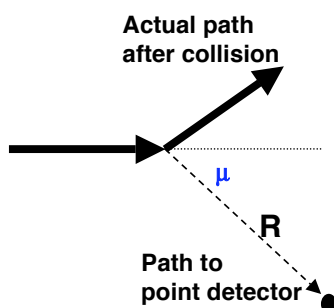
where $\text{wgt}' = $ **weight after collision**

$p_{sc}(\mu) = $ **scatter PDF evaluated at** μ

$E' = $ **energy corresponding to** μ

μ

R

Path to point detector

Flux at a Point

- Expected score has $1/R^2$ singularity – collisions close to detector can result in large scores
  - Point detector estimator has finite mean, but infinite variance due to $1/R^2$ singularity

- To keep variance finite:
  - For collisions within radius $\mathfrak{R}$ of detector, replace the factor

$$\frac{\exp\left\{-\int_0^R \Sigma_T(E')ds\right\}}{R^2}$$

  by volume average assuming uniform collisions inside sphere

$$\frac{\int_0^{\mathfrak{R}} e^{-\Sigma_T(E')s}ds}{\int_0^{\mathfrak{R}} s^2 ds} = \frac{1-e^{\Sigma_T(E')\mathfrak{R}}}{\frac{1}{3}\mathfrak{R}^3\Sigma_T(E')}$$

  - Typically choose $\mathfrak{R}$ to be ~half a mean free path

---

Reaction Rate Tallies
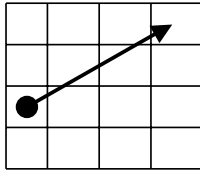
- Tally   (flux-estimator)•(cross-section)
- Example – pathlength tallies

  After each flight,   tally

  - Flux $\mathbf{wgt \bullet d_j}$

  - Total absorption $\mathbf{wgt \bullet d_j \bullet \Sigma_A}$

  - Nu-fission $\mathbf{wgt \bullet d_j \bullet \nu\Sigma_F}$

  - U235 absorption $\mathbf{wgt \bullet d_j \bullet N^{U^{235}} \sigma_A^{U^{235}}}$

Mesh Tallies & Fission Matrix

- **Mesh Tallies**
  - Impose a grid over the problem & tally flux or reaction rates in each grid cell

- **Fission matrix**
  - Impose a grid over problem
  - Tally F(I—>J) for source in cell I causing fission in cell J
  - For N cells in grid, N^2 tallies

RE & FOM

- Some codes (e.g., MCNP) report the mean score & **relative error**

$$\mathbf{RE} \ = \ \frac{\sigma_{\bar{x}}}{\bar{x}}$$

- Some codes report a **Figure-of-Merit** for selected tallies

$$\text{FOM} \ = \ \frac{1}{\text{RE}^2 \bullet \text{T}}$$

Where T = computer time used

  - $RE^2 \sim 1/N$, where N is the total number of histories
  - $T \sim N$
  - Therefore, FOM should be roughly constant
  - Used for comparing effectiveness of different variance reduction schemes

Cautions

- RE should decrease smoothly with $1/\sqrt{N}$ dependence as more histories are run

- Tallies are reliable only if "enough" histories traverse the portions of problem phase space being tallied
    - **Undersampling** can lead to questionable or erroneous values of the mean score & relative error
    - Indicators of undersampling:
        - Large RE,   RE > .1
        - RE does not decrease smoothly as $1/\sqrt{N}$
        - A few histories have very large scores

- MCNP performs statistical checks on selected tallies to try to detect undersampling effects
    - Large RE
    - Variance of the variance (VOV)
    - Tally fluctuation charts (distribution of scores)
    - Slope of tails in tally fluctuation charts
    - Etc.

6 - 23

Combining Independent MC Results

Given N sets of (mean,std-dev) for independent Monte Carlo calculations, (x_1,σ_1), (x_2,σ_2), …, how should the results be combined?

$$w_j = \frac{1}{\sigma_j^2} \qquad W = \sum_{j=1}^{N} \frac{1}{\sigma_j^2}$$

$$\bar{x} = \sum_{j=1}^{N} \frac{w_j}{W} x_j$$

$$\sigma_{\bar{x}}^2 = \sum_{j=1}^{N} \frac{w_j}{W^2} = \frac{1}{W}$$

Weighting factors ~ $1/\sigma^2$

6 - 24

Los Alamos

- Suppose 2 estimators, x and y, are correlated, such as the path & collision estimator for Keff

$$\bar{x} = \frac{1}{N} \sum_{j=1}^{N} x_j \qquad\qquad \bar{y} = \frac{1}{N} \sum_{j=1}^{N} y_j$$

$$\sigma_x^2 = \frac{1}{N} \sum_{j=1}^{N} x_j^2 - \bar{x}^2 \qquad \sigma_y^2 = \frac{1}{N} \sum_{j=1}^{N} y_j^2 - \bar{y}^2 \qquad \sigma_{xy}^2 = \frac{1}{N} \sum_{j=1}^{N} x_j y_j - \bar{x} \cdot \bar{y}$$

Minimum variance combination of x & y

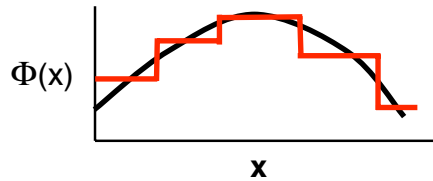$$\alpha = \frac{\sigma_y^2 - \sigma_{xy}^2}{\sigma_x^2 - 2\sigma_{xy}^2 + \sigma_y^2}$$

$$\text{mean}_{x,y} = \alpha \bar{x} + (1-\alpha) \bar{y}$$

$$\text{std} - \text{dev}_{x,y} = \sqrt{\frac{\alpha^2 \sigma_x^2 + 2\alpha(1-\alpha)\sigma_{xy}^2 + (1-\alpha)^2 \sigma_y^2}{N-1}}$$

Los Alamos

- **Conventional Monte Carlo codes tally integral results**
  - **Tallies summed into bins**
  - **Zero-th order quantities**
  - **Stepwise approximation to results**

$\Phi(x)$

x

- **Higher order tallies**
  - **Represent results by high-order, orthogonal polynomial expansion within each cell**
  - **Make tallies for expansion coefficients**
  - **Legendre polynomial representation for continuous tallies**

$$\Phi(x) = \sum_{n=0}^{N} \frac{2n+1}{2} \cdot b_n \cdot P_n \left[\frac{2}{\Delta x}(x - x_{\min}) - 1 \right]$$

$$b_n = \frac{2}{\Delta x} \int_{x_{\min}}^{x_{\max}} \Phi(x) P_n \left[\frac{2}{\Delta x}(x - x_{\min}) - 1 \right] dx$$

- **Make tallies for the Legendre coefficients at each collision or flight:**

$$b_n = \frac{2}{\Delta x} \int_{x_{\min}}^{x_{\max}} \Phi(x) P_n \left[\frac{2}{\Delta x}(x - x_{\min}) - 1 \right] dx$$

- **At collisions, tally** $\dfrac{wgt}{\Sigma_T} \cdot P_n \left[\frac{2}{\Delta x}(x - x_{\min}) - 1 \right]$ **for n=1..N**

- **At flights, tally** $wgt \cdot \dfrac{1}{\mu} \displaystyle\int_{x}^{x+s} P_n \left[\frac{2}{\Delta x}(x' - x_{\min}) - 1 \right] dx'$ **for n=1,N**

- **Reconstruct** $\Phi(x)$ **and** $\sigma_\Phi^2(x)$ **from tallied coefficients**
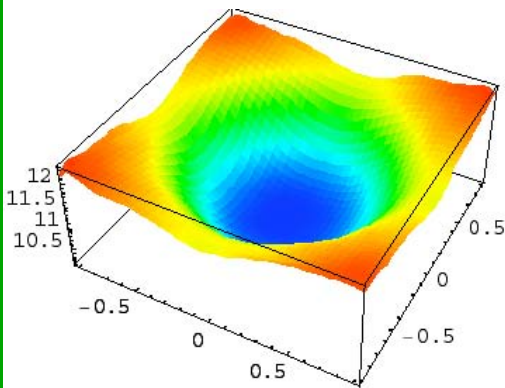
Continuous 2D Tallies – Example

Figure 2a. 9x9 Legendre expansion tally for thermal neutron flux across the fuel pin obtained in a 2 million history simulation.
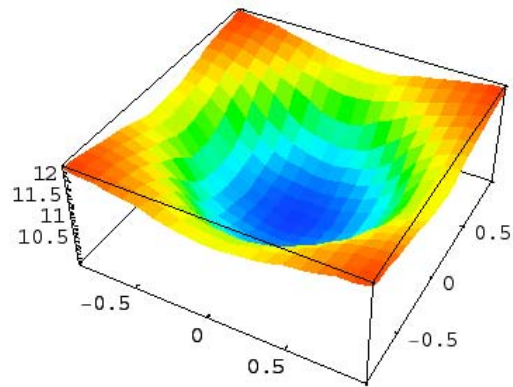
Figure 2b. MCNP5 20×20 mesh tally for thermal neutron flux across the fuel pin obtained in a 2 million history simulation.

DP Griesheimer & WR Martin, **"Two Dimensional Functional Expansion Tallies for Monte Carlo Simulations,"** PHYSOR-2004, Chicago, IL (2004)

References – Continuous Materials & Tallies

- FB Brown, D Griesheimer, & WR Martin, "Continuously Varying Material Properties and Tallies for Monte Carlo Calculations", PHYSOR-2004, Chicago, IL (April, 2004)

- FB Brown & WR Martin, "Direct Sampling of Monte Carlo Flight Paths in Media with Continuously Varying Cross-sections", ANS Mathematics & Computation Topical Meeting, Gatlinburg, TN (April, 2003).

- DP Griesheimer & WR Martin, "Estimating the Global Scalar Flux Distribution with Orthogonal Basis Function Expansions", Trans. Am. Nucl. Soc. 89 (Nov, 2003)

- DP Griesheimer & WR Martin, "Two Dimensional Functional Expansion Tallies for Monte Carlo Simulations," PHYSOR-2004, Chicago, IL (April, 2004)

- ER Woodcock, T Murphy, PJ Hemmings, TC Longworth, "Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry," *Proc. Conf. Applications of Computing Methods to Reactor Problems*, ANL-7050, p. 557, Argonne National Laboratory (1965).

- LL Carter, ED Cashwell, & WM Taylor, "Monte Carlo Sampling with Continuously Varying Cross Sections Along Flight Paths", *Nucl. Sci. Eng.* **48**, 403–411 (1972).

- J. Spanier, "Monte Carlo Methods for Flux Expansion Solutions of Transport Problems," Nucl. Sci. Eng., **133**, 73 (1999).

Eigenvalue Calculations Part I

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Time-dependent Transport

- **Time-dependent neutron transport with (prompt) fission source**

$$\frac{1}{v}\frac{\partial \psi(\vec{r},E,\vec{\Omega},t)}{\partial t} = \left[-\vec{\Omega}\cdot\nabla - \Sigma_T(\vec{r},E)\right]\psi + \iint \psi(\vec{r},E',\vec{\Omega}',t)\,\Sigma_S(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\,d\vec{\Omega}'dE'$$

$$+ \frac{\chi(E)}{4\pi}\iint \nu\Sigma_F(\vec{r},E')\,\psi(\vec{r},E',\vec{\Omega}',t)\,d\vec{\Omega}'dE' + S(\vec{r},E,\vec{\Omega},t)$$

This equation can be solved directly by Monte Carlo
- Simulate time-dependent transport for a neutron history
- If fission occurs, bank any secondary neutrons. When original particle is finished, simulate secondaries till done.
- Tallies for time bins, energy bins, cells, …

Overall time-behavior $\psi(r,E,\Omega,t) = \Psi(r,E,\Omega)\,e^{\alpha t}$ **can be estimated by**

$$\alpha \approx \frac{\ln W_2 - \ln W_1}{t_2 - t_1} \qquad \text{where} \quad W_j = \sum_{k=1}^{N_{particles}} wgt_k(t_j)$$

Alpha Eigenvalue Equations

- For problems which are separable in space & time, it may be advantageous to solve a **static eigenvalue problem**, rather than a fully time-dependent problem
- **If it is assumed that** $\psi(r,E,\Omega,t) = \Psi_\alpha(r,E,\Omega)\, e^{\alpha t}$, then substitution into the time-dependent transport equation yields

$$\left[\vec{\Omega}\cdot\nabla + \Sigma_T(\vec{r},E) + \frac{\alpha}{v} \right] \Psi_\alpha(\vec{r},E,\vec{\Omega}) = \iint \Psi_\alpha(\vec{r},E',\vec{\Omega}')\Sigma_S(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\,d\vec{\Omega}'dE'$$

$$+ \ \frac{\chi(E)}{4\pi} \iint v\Sigma_F(\vec{r},E')\Psi_\alpha(\vec{r},E',\vec{\Omega}')\,d\vec{\Omega}'dE'$$

- This is a **static** equation, an **eigenvalue problem for α and Ψ_α** without time-dependence
- α is often called the time-eigenvalue or time-absorption
- α-eigenvalue problems can be solved by Monte Carlo methods

K_eff Eigenvalue Equations

- Another approach to creating a static eigenvalue problem from the time-dependent transport equation is to introduce K_{eff}, a scaling factor on the multiplication (v)
- **Setting $\partial\psi/\partial t = 0$ and introducing the K_{eff} eigenvalue gives**

$$\left[\vec{\Omega}\cdot\nabla + \Sigma_T(\vec{r},E) \right] \Psi_k(\vec{r},E,\vec{\Omega}) = \iint \Psi_k(\vec{r},E',\vec{\Omega}')\Sigma_S(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')\,d\vec{\Omega}'dE'$$

$$+ \ \frac{1}{K_{eff}}\cdot\frac{\chi(E)}{4\pi} \iint v\Sigma_F(\vec{r},E')\Psi_k(\vec{r},E',\vec{\Omega}')\,d\vec{\Omega}'dE'$$

- This is a **static** equation, an **eigenvalue problem for K_{eff} and Ψ_k** without time-dependence
- K_{eff} is called the effective multiplication factor
- K_{eff} and Ψ_k should **never** be used to model time-dependent problems. [Use α and Ψ_α instead]
- K_{eff}-eigenvalue problems can be solved by Monte Carlo methods

• **Criticality**

Supercritical: $\alpha > 0$ or $K_{eff} > 1$

Critical: $\alpha = 0$ or $K_{eff} = 1$

Subcritical: $\alpha < 0$ or $K_{eff} < 1$

• **K_{eff} vs. α eigenvalue equations**

– $\Psi_K(r,E,\Omega) \neq \Psi_\alpha(r,E,\Omega)$, except for a critical system

– α eigenvalue & eigenfunction used for time-dependent problems

– K_{eff} eigenvalue & eigenfunction used for reactor design & analysis

– Although $\alpha = (K_{eff}-1)/\Lambda$, where Λ = lifetime,
there is **no** direct relationship between $\Psi_K(r,E,\Omega)$ and $\Psi_\alpha(r,E,\Omega)$

• K_{eff} eigenvalue problems can be simulated directly using Monte Carlo methods

• α eigenvalue problems are solved by Monte Carlo indirectly using a series of K_{eff} calculations

7 -5

K-Eigenvalue Calculations

• **Eigenvalue problems – reactor analysis & criticality safety**

$$\Psi(p) = \int \Psi(p')R(p' \to p)dp' \; + \; \frac{1}{K_{eff}} \int \Psi(p')F(p' \to p)dp'$$

$$\Psi = R \bullet \Psi + \frac{1}{K_{eff}} F \bullet \Psi$$

Iterative solution, using power iteration method

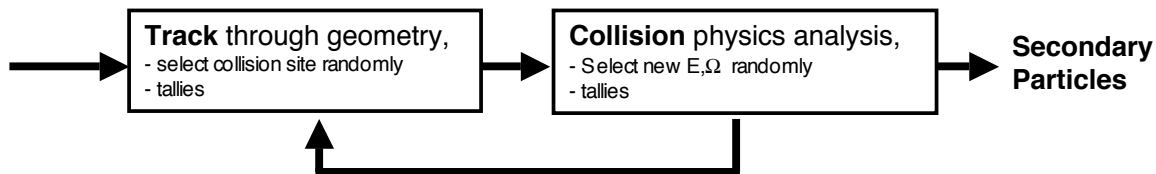$$\Psi^{(i+1)} = R \bullet \Psi^{(i+1)} + \frac{1}{K_{eff}^{(i)}} F \bullet \Psi^{(i)}$$

$$\Psi^{(i+1)} = \frac{1}{K_{eff}^{(i)}} [I - R]^{-1} F \bullet \Psi^{(i)} \qquad K_{eff}^i = \int F \bullet \Psi^{(i)} dp dp'$$
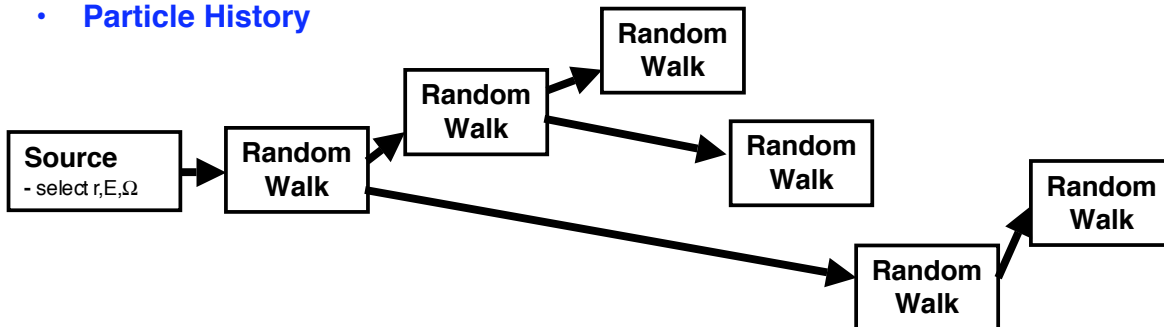
• **Monte Carlo approach:**

– Guess $\Psi^{(0)}$, $K_{eff}^{(0)}$

– Follow a "batch" of histories, estimate $\Psi^{(i)}$, $K_{eff}^{(i)}$

– Repeat until converged (discard tallies)

– After converging, begin tallies, iterate until variances small enough

7 -6

Particle Histories

- **Random Walk for particle**

| Track through geometry, - select collision site randomly - tallies | → | Collision physics analysis, - Select new E,Ω randomly - tallies | → | Secondary Particles |

- **Particle History**

| Source - select r,E,Ω | → | Random Walk | → | Random Walk | → | Random Walk |

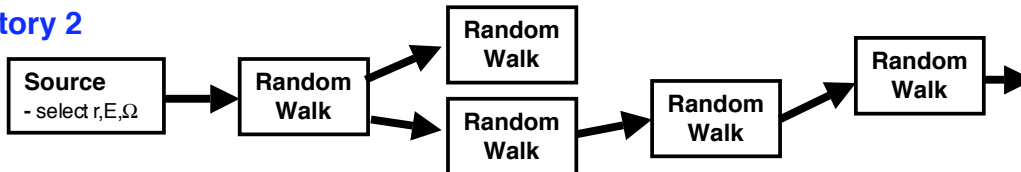
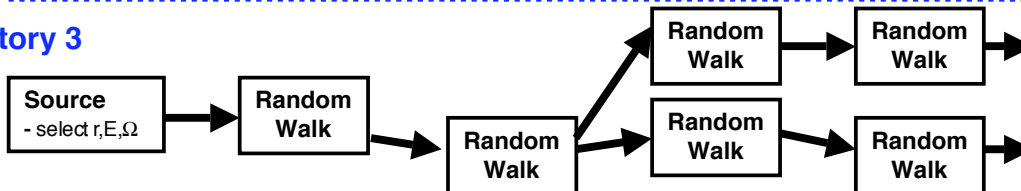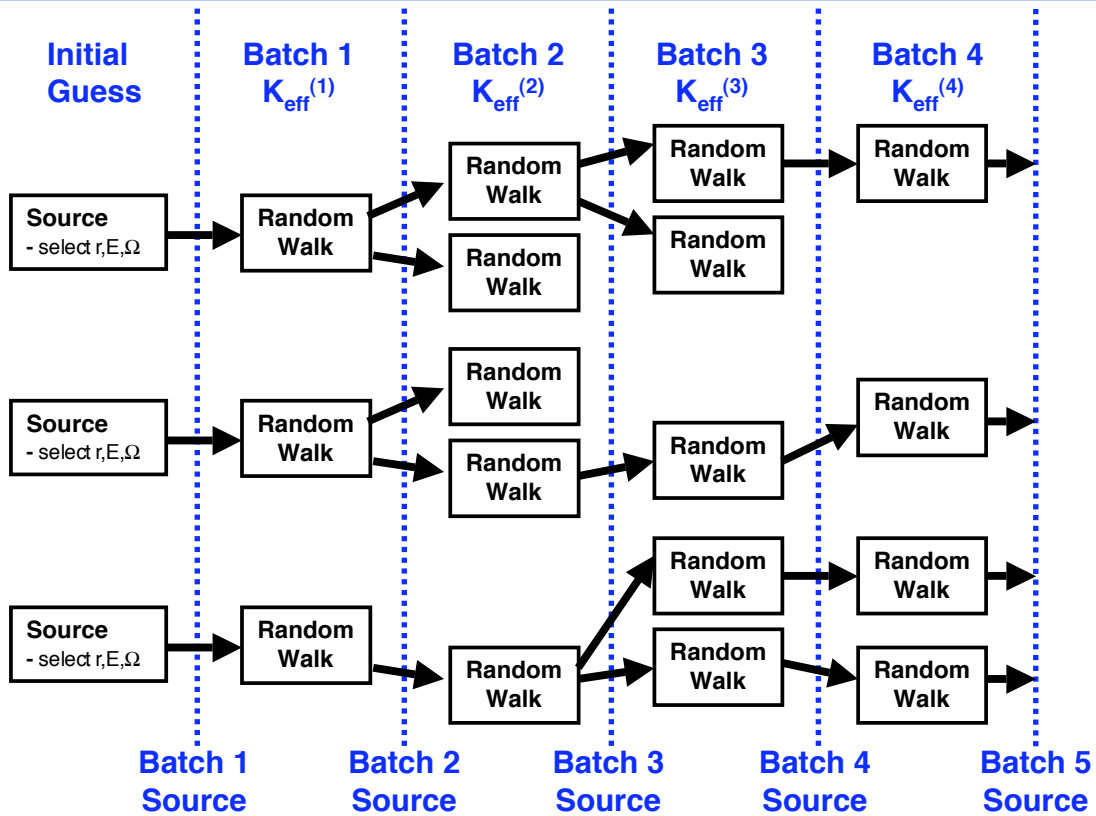# Fixed-source Monte Carlo Calculation

History 1
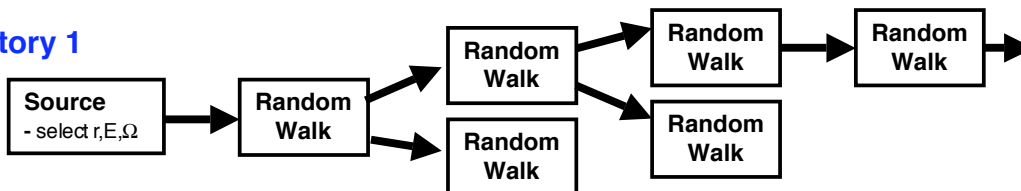
History 2

History 3

Monte Carlo Eigenvalue calculation

Monte Carlo Solution of K$_{eff}$ Problems

Note: batch = cycle = iteration = generation

- **Initialize**
  - Assume a value for the initial K$_{eff}$ (usually, K$_0$ = 1)
  - Sample **M** fission sites from the initial source distribution

- **For each cycle n,   n = 1 … N+D**
  - Follow histories for all source particles in cycle
    - If fissions occur, bank the sites for use as source in next cycle
    - Make tallies for K$_{cycle}^{(n)}$ using path, collision, & absorption estimators
    - If   n ≤ D,   discard any tallies
    - If   n > D,   accumulate tallies
  - Estimate K$_{cycle}^{(n)}$

- Compute final results & statistics using last **N** cycles

K-Calculations — Convergence
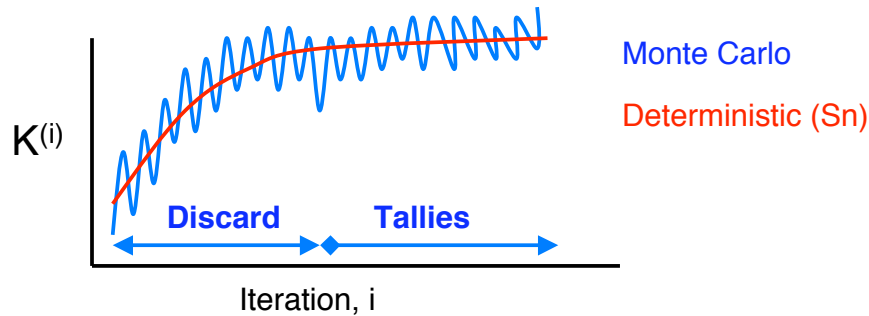
$K^{(i)}$

Monte Carlo

Deterministic (Sn)

Discard **Tallies**

Iteration, i

- Guess an initial source distribution
- Iterate until converged (How do you know ???)
- Then
  - For Sn code: done, print the results
  - For Monte Carlo: start tallies, keep running until uncertainties small enough
- **Batch size? Convergence? Stationarity? Bias? Statistics?**

K-Calculations — Banking Fission Sites

- During a particle random walk,

$$\mathbf{wgt} \cdot \frac{\nu \Sigma_F}{\Sigma_T}$$ = expected number of fission neutrons created at collision point

- Averaged over all collisions for all histories, the expected value for $wgt \cdot \nu\Sigma_F/\Sigma_T$ is K_{eff}.

- In order to bank approximately the same number of fission sites in each cycle, the current value of Keff is used to bias the selection of fission sites at a collision:

$$\mathbf{R = wgt} \cdot \frac{\nu \Sigma_F}{\Sigma_T} \cdot \frac{\mathbf{1}}{\mathbf{K}}, \qquad \mathbf{n = \lfloor R \rfloor}$$

If $\xi < \mathbf{R - n}$, **store** $\mathbf{n + 1}$ **sites in bank with** $\mathbf{wgt' = K}$

Otherwise, **store** $\mathbf{n}$ **sites in bank with** $\mathbf{wgt' = K}$

- N_J = number of particles starting cycle J,
  N'_J = number of particles created by fission in cycle J
  (number of particles stored in fission bank)

  – The expected value for N'_J is: $E[\,N'_J\,] = K_{eff} \cdot N_J$
  – (N'_J/N_J) is a single-cycle estimator for K_{eff}

- To prevent the number of particles per cycle from growing exponentially (for K>1) or decreasing to 0 (for K<1), the particle population is **renormalized** at the end of **each cycle**:

  – **In some Monte Carlo codes, the number of particles starting each cycle is a constant N**. Russian roulette or splitting are used to sample N particles from the N' particles in the fission bank. (All particles in fission bank have a weight of 1.0)

  – **In other codes, the total weight W starting each cycle is constant.** The particle weights in the fission bank are renormalized so that the total weight is changed from W' to W. (Particles in fission bank have equal weights, but not necessarily 1.0)

Single-cycle Keff Estimators

- Pathlength estimator for Keff

$$K_{path} = \left(\sum_{\substack{all \\ flights}} wgt_j \cdot d_j \cdot \nu\Sigma_F \right) \Big/ W$$

W = total weight starting each cycle

- Collision estimator for Keff

$$K_{collision} = \left(\sum_{\substack{all \\ collisions}} wgt_j \cdot \frac{\nu\Sigma_F}{\Sigma_T} \right) \Big/ W$$

- Absorption estimator for Keff

$$K_{absorption} = \left(\sum_{\substack{all \\ absorptions}} wgt_j \cdot \frac{\nu\Sigma_F}{\Sigma_A} \right) \Big/ W$$

K-Calculations — Overall Keff

- The Keff estimators from each cycle (K_{path}, $K_{collision}$, $K_{absorption}$) are used to compute the overall K_{path}, $K_{collision}$, & $K_{absorption}$ for the problem & the standard deviations.

- The Keff estimators from each cycle (K_{path}, $K_{collision}$, $K_{absorption}$) can also be combined to produce a minimum-variance combined result, $K_{combination}$. This combination must account for correlations between the path, collision, & absorption estimators

K-Calculations — Bias

- The renormalization procedure used at the end of each cycle introduces a small bias into the computed Keff
  – Renormalization involves multiplying particle weights by (W/W'), where W = total weight starting a cycle,
        W'= total weight at the end of a cycle.
  – W' is a random variable, due to fluctuations in particle random walks.

- Theoretical analysis of the MC iteration process & propagation of history fluctuations gives

$$\text{bias in } K_{eff} \quad = \quad -\frac{\sigma_k^2}{K_{eff}} \cdot \left(\begin{array}{c} \text{sum of correlation coeff's} \\ \text{between batch K's} \end{array} \right)$$
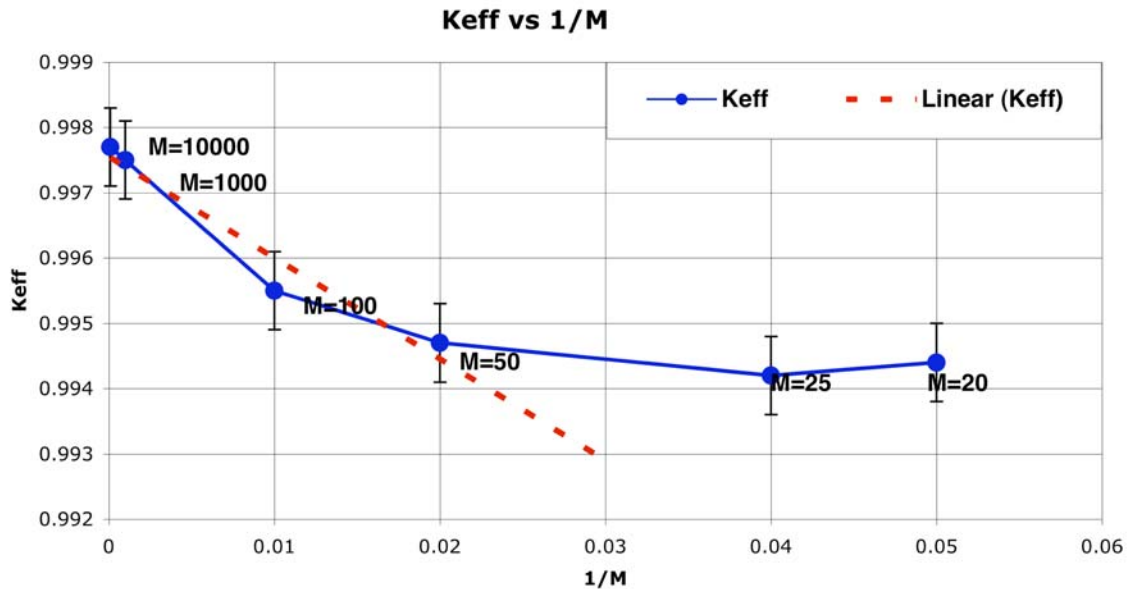
  – M = histories/cycle
  – **Bias in Keff $\sim$ 1/M**
    - Smaller M $\Rightarrow$ larger cycle correlation $\Rightarrow$ larger bias in Keff & source
    - Larger M $\Rightarrow$ smaller cycle correlation $\Rightarrow$ smaller bias

[T Ueki, "**Intergenerational Correlation in Monte Carlo K-Eigenvalue Calculations**", Nucl. Sci. Eng. (2002)]

K-Calculations — Bias

- For a simple Godiva reactor calculation:

K-Calculations — Bias

- Observed PDF for single-cycle Keff, for varying M
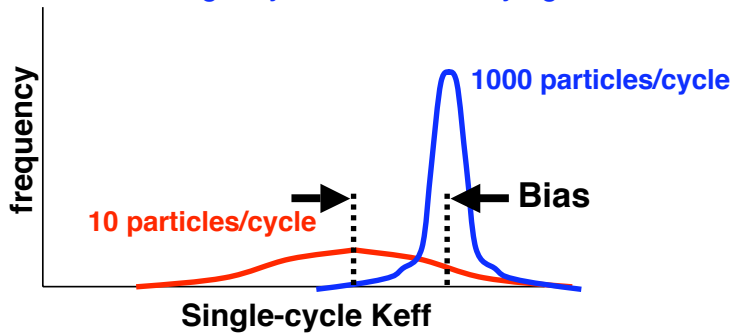
- Bias in Keff is negative: $K_{calc} < K_{true}$

- Bias is
significant	for	$M < 10$	particles/cycle
small	for	$M \sim 100$	
negligible	for	$M > 1000$	
0	for	$M \to \infty$	

- Recommendation: **Always use 1000 or more particles/cycle, preferably 5000, 10000, or more**

K-Calculations — Convergence

- **Some number of initial cycles must be discarded**
  - The source distribution & Keff are not known initially
  - Guess at the source & Keff
  - Iterate, discarding tallies
  - When converged, iterate to accumulate tallies

- **Number of iterations to discard depends on the dominance ratio**
 - Dominance Ratio = K_1 / K_{eff}
    - K_{eff} = eigenvalue of fundamental eigenmode
    - K_1 = eigenvalue of first higher eigenmode, $K_1 < K_{eff}$
  - If DR close to 1 (e.g., .999…), 100s or 1000s of initial iterations may be required for initial source distribution errors to die away
  - Most statistical tests for convergence are *ex post facto* tests to look for trends
  - Most common practice is to examine plots of Keff vs. cycles

K-Calculations — Convergence

- Plots of single-cycle Keff vs. cycle number

koode data from file runtpe
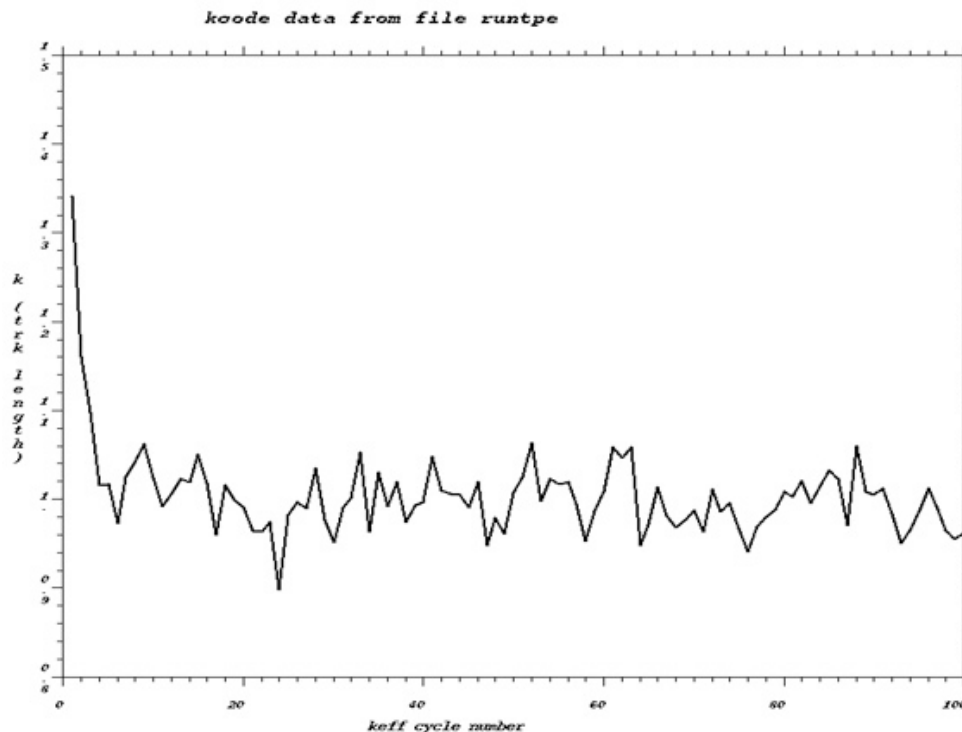
keff cycle number

K-Calculations — Convergence

- Plots of cumulative Keff vs. cycle number
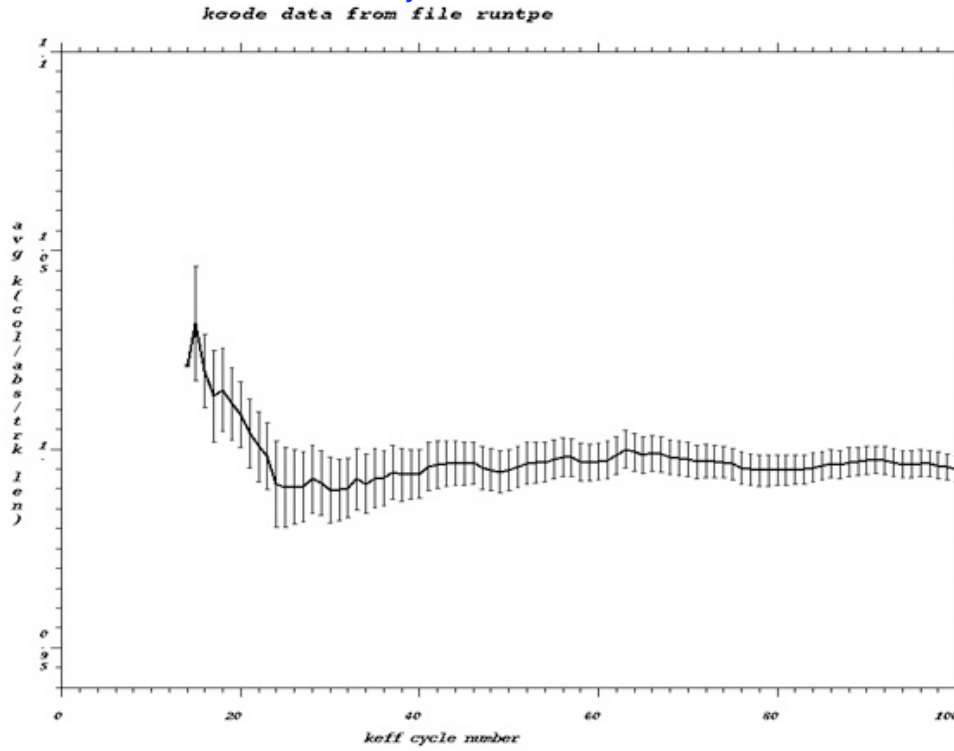
7 -21

K-Calculations — Convergence

- Plots of cumulative Keff vs. number of initial cycles discarded
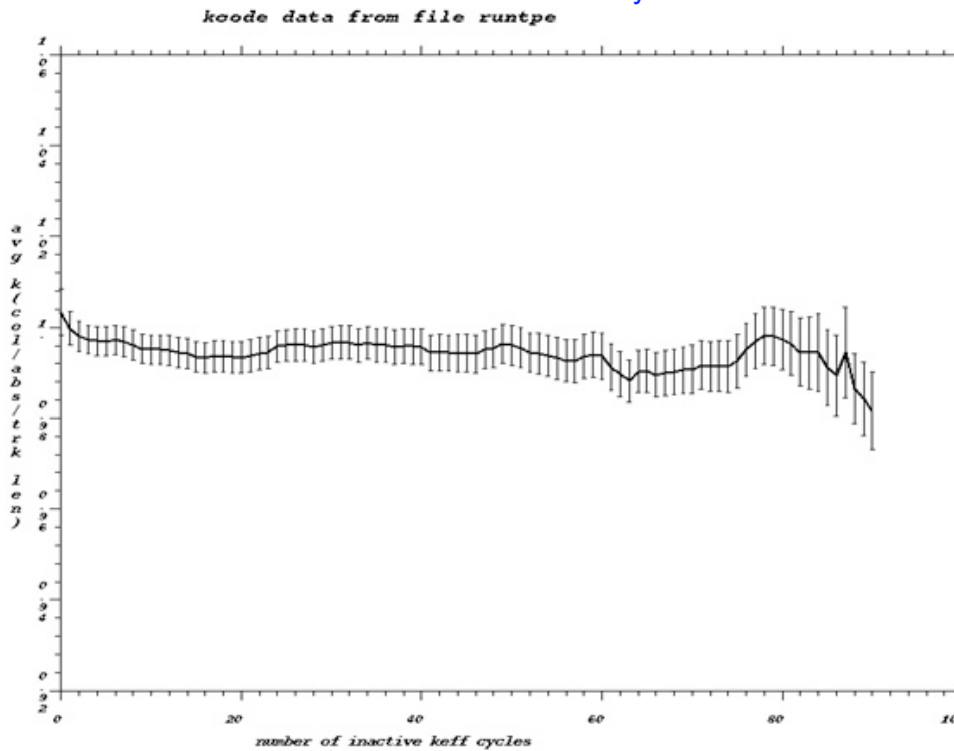
7 -22

K-Calculations — Convergence

- Keff is an integral quantity – converges faster than source shape

Keff calculation for 2 nearly symmetric slabs, with Dominance Ratio = .9925
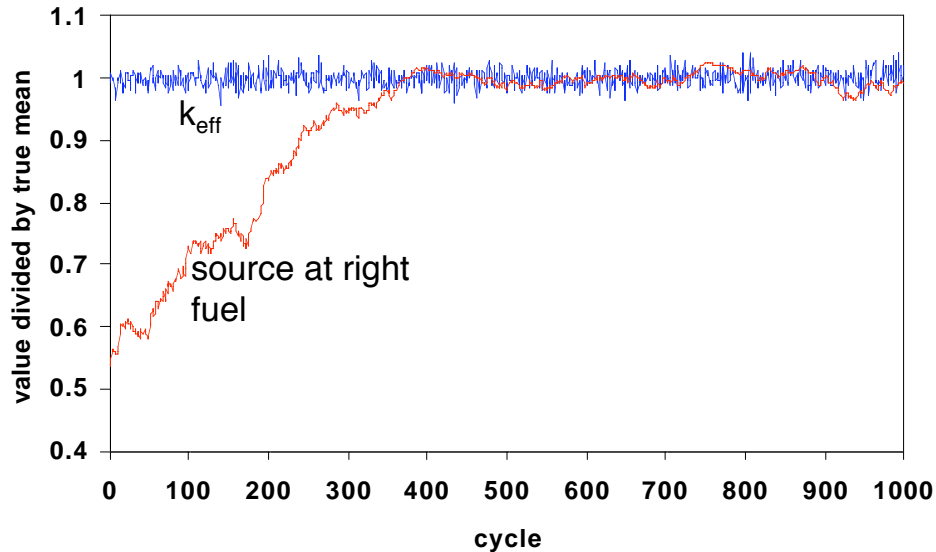
7 -23

K-Calculations — Convergence

- Choose the number of cycles to discard by examining convergence plots

- Then, choose the total number of cycles to be large enough so that relative errors are "small enough"
  - Always run >25 cycles for tallies, to get good estimates of σ^2
  - Always try to run a few 100 or 1000 cycles for tallies
    - Statistical tests on convergence more reliable if more cycles
    - Better plots for assessing convergence

- **Summary**
  - **Particles per cycle - > 1000**
  - **Discarded cycles -   varies, check plots**
  - **Tally cycles -   > 100**

7 -24

α-Eigenvalue Calculations

- Eigenvalue equation with **both** K_{eff} & α
  - **α is a fixed number**, not a variable

$$\left[\vec{\Omega} \cdot \nabla + \Sigma_T(\vec{r},E) + \max(\frac{\alpha}{v},0) \right] \Psi_\alpha(\vec{r},E,\vec{\Omega})$$

$$= \max(\frac{-\alpha}{v},0)\Psi_\alpha(\vec{r},E,\vec{\Omega}) + \iint \Psi_\alpha(\vec{r},E',\vec{\Omega}')\Sigma_S(\vec{r},E' \to E,\vec{\Omega}\cdot\vec{\Omega}')d\vec{\Omega}'dE'$$

$$+ \frac{1}{K_{eff}} \cdot \frac{\chi(E)}{4\pi} \iint v\Sigma_F(\vec{r},E')\Psi_\alpha(\vec{r},E',\vec{\Omega}')d\vec{\Omega}'dE'$$

- Note on the max(α/v, 0) and max(-α/v, 0) terms
  - If $\alpha < 0$, real absorption plus time absorption could be negative
  - If $\alpha < 0$, move α/v to right side to prevent negative absorption,
  - If $\alpha < 0$, -α/v term on right side is treated as a delta-function source

- Select a fixed value for α
- Solve the K-eigenvalue equations, with fixed time-absorption α/v
- Select a different α and solve for a new Keff
- Repeat, searching for value of α which results in Keff = 1

Special Topic – Stationarity Tests

- Plots of single-cycle Keff or cumulative Keff are difficult to interpret when assessing convergence

Cycle k_eff
50000 histories per cycle
— uniform initial source
— initial source at (1,3) lattice
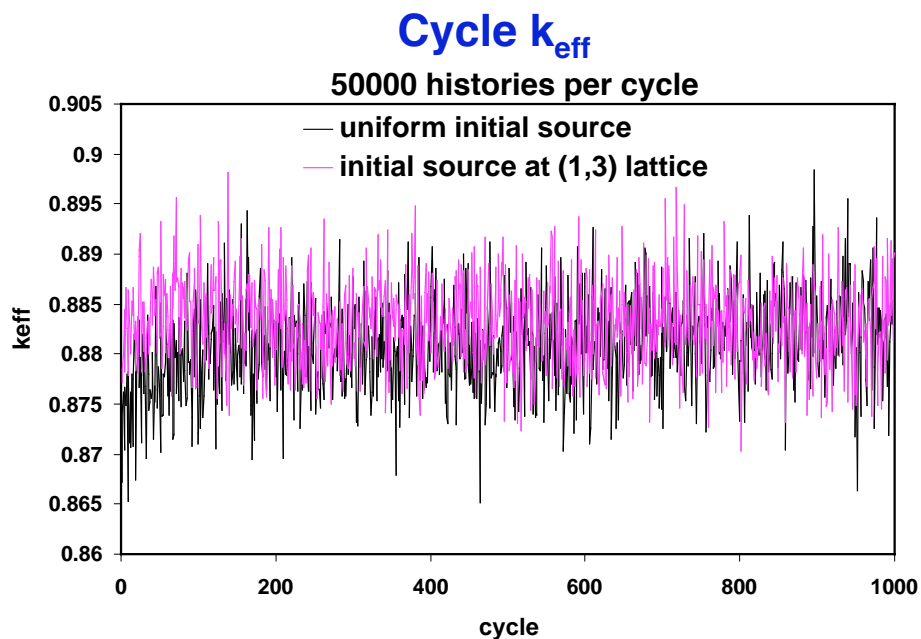
- The MCNP team has been investigating new stationarity tests

Progressive relative entropy

Posterior relative entropy
(500 inactive and 500 active cycles)

Special Topic – Stationarity Tests

**One cycle delay embedding plot of
relative entropy wrt initial source**

Special Topic – Stationarity Tests
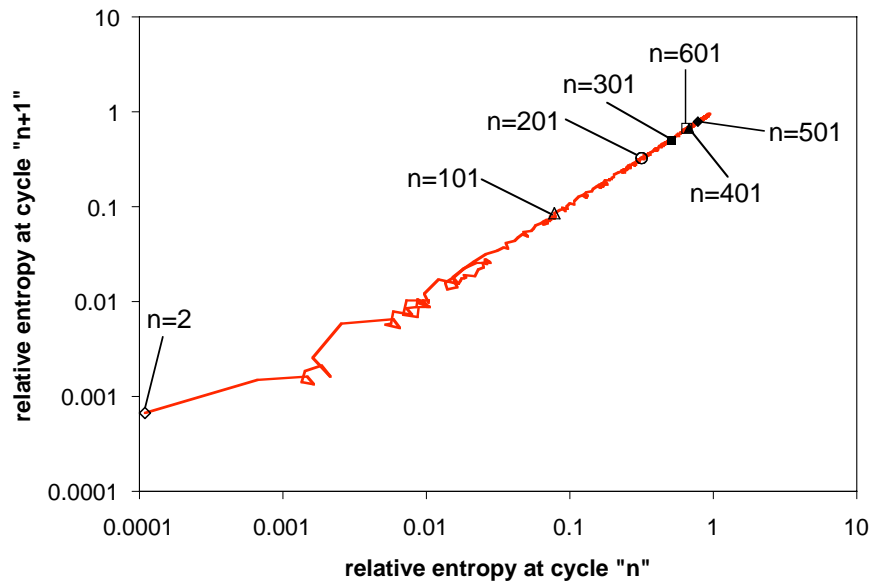
- In a series of related papers, we have significantly extended the theory of Monte Carlo eigenvalue calculations, explicitly accounting for correlation effects.

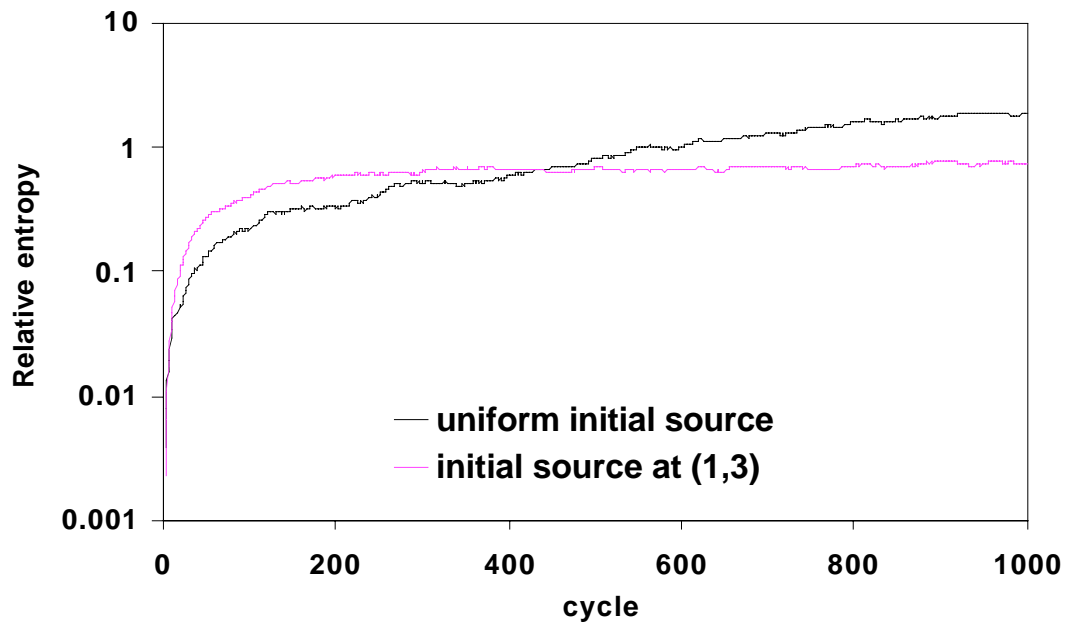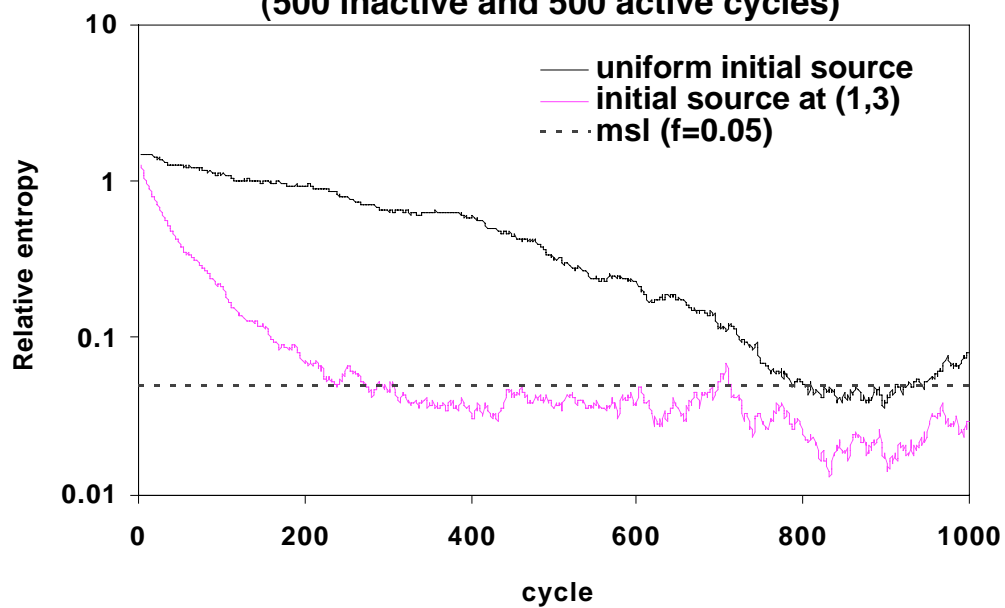LA-UR-02–0190:   T Ueki, "**Intergenerational Correlation in Monte Carlo K-Eigenvalue Calculations**", Nucl. Sci. Eng. (2002)

LA-UR-01–6770:   T Ueki & FB Brown, "**Autoregressive Fitting for Monte Carlo K-effective Confidence Intervals**", ANS Summer Meeting, (June 2002)

LA-UR-02–3783:   T Ueki & FB Brown, "**Stationarity Diagnostics Using Shannon Entropy in Monte Carlo Criticality Calculations I: F Test**", ANS Winter Meeting (Nov 2002)

LA-UR-02–6228:   T Ueki & FB Brown, "**Stationarity and Source Convergence in Monte Carlo Criticality Calculations**", ANS Topical Meeting on Mathematics & Computation, Gatlinburg, TN (April, 2003)

LA-UR-03–0106:   T Ueki, FB Brown, DK Parsons, "**Dominance Ratio Computation via Time Series Analysis of Monte Carlo Fission Sources**" , ANS Annual Meeting (June 2003)

LA-UR-02–5700:   T Ueki, FB Brown, DK Parsons, & DE Komreich, "**Autocorrelation and Dominance Ratio in Monte Carlo Criticality Calculations**", Nucl. Sci. Eng. (Nov 2003)

LA-UR-03–3949:   T Ueki & FB Brown, "**Informatics Approach to Stationarity Diagnostics of the Monte Carlo Fission Source Distribution**", ANS Winter meeting (Nov 2003)

LA-UR-03–5823:   T Ueki, FB Brown, DK Parsons, JS Warsa, "**Time Series Analysis of Monte Carlo Fission Source: I. Dominance Ratio Calculation**", Nucl. Sci. Eng. (Nov 2004)

LA-UR-03-????:   T Ueki & FB Brown, "**Stationarity Modeling and Informatics-Based Diagnostics in Monte Carlo Criticality Calculations,**" submitted to Nucl. Sci. Eng.

Eigenvalue Calculations Part II

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

8 -1

Eigenvalue Calculations – Part II

- **K-eigenvalue equation**

- **Solution by power iteration**

- **Convergence of power iteration**

- **Stationarity Diagnostics**

- **Weilandt acceleration method**

- **Superhistory method**

8 -2

$$\left[\vec{\Omega}\cdot\nabla + \Sigma_T(\vec{r},E)\right]\Psi_k(\vec{r},E,\vec{\Omega}) = \iint \Psi_k(\vec{r},E',\vec{\Omega}')\Sigma_S(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')d\vec{\Omega}'dE'$$

$$+\ \frac{1}{K_{eff}}\cdot\frac{\chi(E)}{4\pi}\iint \nu\Sigma_F(\vec{r},E')\Psi_k(\vec{r},E',\vec{\Omega}')d\vec{\Omega}'dE'$$

where

$$K_{eff} \qquad = \text{k-effective, eigenvalue for fundamental mode}$$

$$\Psi_k(\vec{r},E,\vec{\Omega}) \qquad = \text{angular flux, for fundamental k-eigenmode}$$

$$\vec{\Omega}\cdot\nabla\Psi_k(\vec{r},E,\vec{\Omega}) \qquad = \text{loss term, leakage}$$

$$\Sigma_T(\vec{r},E)\,\Psi_k(\vec{r},E,\vec{\Omega}) \qquad = \text{loss term, collisions}$$

$$\iint \Psi_k(\vec{r},E',\vec{\Omega}')\Sigma_S(\vec{r},E'\to E,\vec{\Omega}\cdot\vec{\Omega}')d\vec{\Omega}'dE' \qquad = \text{gain term, scatter from E',}\Omega'\text{ into E,}\Omega$$

$$\frac{1}{K_{eff}}\cdot\frac{\chi(E)}{4\pi}\iint \nu\Sigma_F(\vec{r},E')\Psi_k(\vec{r},E',\vec{\Omega}')d\vec{\Omega}'dE' \qquad = \text{gain term, production from fission}$$

⇒ **Jointly find K_{eff} and $\Psi_k(r,E,\Omega)$ such that equation balances**

• Use operator (or matrix) form to simplify notation

$$(L + T)\Psi = S\Psi + \tfrac{1}{K_{eff}}M\Psi$$

where

L = leakage operator   S = scatter-in operator

T = collision operator   M = fission multiplication operator

• Rearrange

$$(L + T - S)\Psi = \tfrac{1}{K_{eff}}M\Psi$$

$$\Psi = \tfrac{1}{K_{eff}}\cdot(L + T - S)^{-1}M\Psi$$

$$\Psi = \tfrac{1}{K_{eff}}\cdot F\Psi$$

⇒ **This eigenvalue equation will be solved by __power iteration__**

Eigenvalue equation

$$\Psi = \frac{1}{K_{eff}} \cdot F\Psi$$

1. **Assume that k_{eff} and Ψ on the right side are known for iteration n, solve for Ψ on left side (for iteration n+1)**

$$\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} \cdot F\Psi^{(n)}$$

Note: This requires solving the equation below for $\Psi^{(n+1)}$, with $K_{eff}^{(n)}$ and $\Psi^{(n)}$ fixed

$$(L + T - S)\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} M\Psi^{(n)}$$

2. **Then, compute $K_{eff}^{(n+1)}$**

$$K_{eff}^{(n+1)} = K_{eff}^{(n)} \cdot \frac{\int M\Psi^{(n+1)} d\vec{r}}{\int M\Psi^{(n)} d\vec{r}} \qquad \text{(other norms could be used)}$$

- **Power iteration procedure:**

1. **Initial guess for K_{eff} and Ψ**

    $K_{eff}^{(0)}, \quad \Psi^{(0)}$

    Source points for $\Psi^{(0)}$
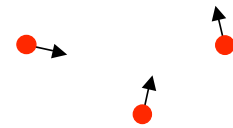
2. **Solve for $\Psi^{(n+1)}$ [Monte Carlo random walk for N particles]**

$$\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} \cdot F\Psi^{(n)}$$

    Source points for $\Psi^{(n+1)}$
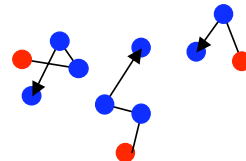
3. **Compute new K_{eff}**

$$K_{eff}^{(n+1)} = K_{eff}^{(n)} \cdot \frac{\int M\Psi^{(n+1)} d\vec{r}}{\int M\Psi^{(n)} d\vec{r}}$$

4. **Repeat 1–3 until both $K_{eff}^{(n+1)}$ and $\Psi^{(n+1)}$ have converged**

Power Iteration

- **Power iteration for Monte Carlo k-effective calculation**
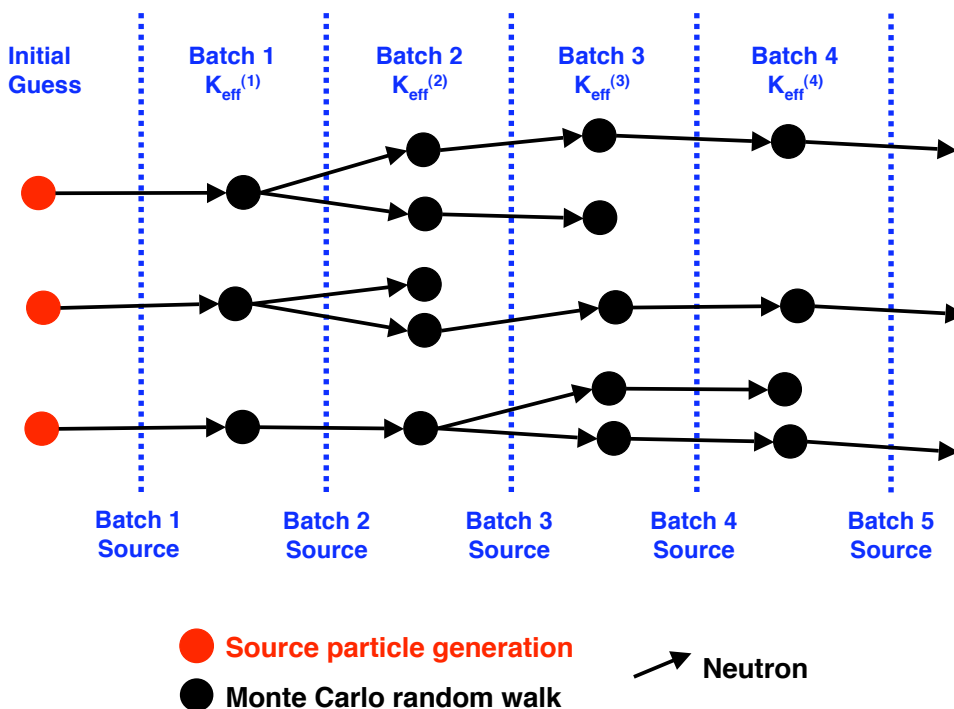
Power Iteration

Diffusion Theory or Discrete-ordinates Transport	**Monte Carlo**

Diffusion Theory or Discrete-ordinates Transport

1. Initial guess for K_{eff} and Ψ

$$K_{eff}^{(0)}, \quad \Psi^{(0)}$$

2. Solve for $\Psi^{(n+1)}$

   Inner iterations over space or space/angle to solve for $\Psi^{(n+1)}$

$$(L + T - S)\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} M\Psi^{(n)}$$

3. Compute new K_{eff}

$$K_{eff}^{(n+1)} = K_{eff}^{(n)} \cdot \frac{1 \cdot M\Psi^{(n+1)}}{1 \cdot M\Psi^{(n)}}$$

4. Repeat 1–3 until both $K_{eff}^{(n+1)}$ and $\Psi^{(n+1)}$ have converged

Monte Carlo

1. Initial guess for K_{eff} and Ψ

$$K_{eff}^{(0)}, \quad \Psi^{(0)}$$

2. Solve for $\Psi^{(n+1)}$

   Follow particle histories to solve for $\Psi^{(n+1)}$

$$(L + T - S)\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} M\Psi^{(n)}$$

   During histories, save fission sites to use for source in next iteration

3. Compute new K_{eff}

   During histories for iteration (n+1), estimate $K_{eff}^{(n+1)}$

$$K_{eff}^{(n+1)} = K_{eff}^{(n)} \cdot \frac{\int M\Psi^{(n+1)} d\vec{r}}{\int M\Psi^{(n)} d\vec{r}}$$

4. Repeat 1–3 until both $K_{eff}^{(n+1)}$ and $\Psi^{(n+1)}$ have converged

5. **Continue iterating, to compute tallies**
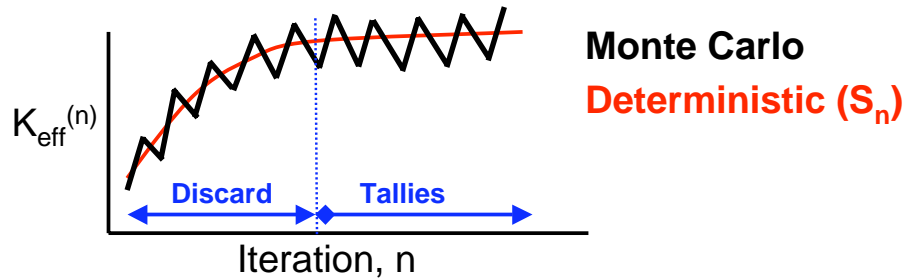
Power Iteration

- **Guess an initial source distribution**
- **Iterate until converged**   **(How do you know ???)**
- **Then**
  - **For S_n code:**   **done, print the results**
  - **For Monte Carlo:**   **start tallies,**
    keep running until uncertainties small enough

- **Convergence? Stationarity? Bias? Statistics?**

## Power Iteration – Convergence

- **Expand Ψ in terms of eigenfunctions $u_j(r, E, \Omega)$**

$$\Psi = \sum_{j=0}^{\infty} a_j \vec{u}_j = a_0 \vec{u}_0 + a_1 \vec{u}_1 + a_2 \vec{u}_2 + a_3 \vec{u}_3 + ...$$

$$\int \vec{u}_j \vec{u}_k dV = \delta_{jk}$$

$$a_j = \int \Psi \cdot \vec{u}_j dV$$

$$\vec{u}_j = \frac{1}{k_j} F \cdot \vec{u}_j \qquad k_0 > k_1 > k_2 > ...$$

$$k_0 \equiv k_{effective}$$

- **Expand the initial guess in terms of the eigenmodes**

$$\Psi^{(0)} = \sum_{j=0} a_j^{(0)} \vec{u}_j$$

- **Substitute the expansion for Ψ into eigenvalue equation**

$$\Psi^{(n+1)} = \frac{1}{K^{(n)}} F \cdot \Psi^{(n)} = \frac{1}{k^{(n)}} \cdot \frac{1}{k^{(n-1)}} \cdots \frac{1}{k^{(0)}} \cdot F^n \cdot \Psi^{(0)}$$

$$= \left[\prod_{m=0}^{n} \frac{k_0}{K^{(m)}} \right] \cdot a_0^{(0)} \cdot \left[\vec{u}_0 + \sum_{j=1} \left(\frac{a_j^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_j}{k_0} \right)^{n+1} \cdot \vec{u}_j \right]$$

$$\approx [\text{constant}] \cdot \left[\vec{u}_0 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot \vec{u}_1 + \left(\frac{a_2^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_2}{k_0} \right)^{n+1} \cdot \vec{u}_2 + \dots \right]$$

Power Iteration – Convergence

$$\Psi^{(n+1)} \approx [\text{constant}] \cdot \left[\vec{u}_0 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot \vec{u}_1 + \left(\frac{a_2^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_2}{k_0} \right)^{n+1} \cdot \vec{u}_2 + \dots \right]$$

$$K^{(n+1)} \approx k_0 \cdot \frac{\left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot G_1 + \left(\frac{a_2^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_2}{k_0} \right)^{n+1} \cdot G_2 + \dots \right]}{\left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n} \cdot G_1 + \left(\frac{a_2^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_2}{k_0} \right)^{n} \cdot G_2 + \dots \right]}$$

where

$$G_m = \frac{\int M \vec{u}_m d\vec{r}}{\int M \vec{u}_0 d\vec{r}}$$

$\Rightarrow$ **Because $k_0 > k_1 > k_2 > \dots$, all of the red terms vanish as $n \rightarrow \infty$,**

thus

$$\Psi^{(n+1)} \rightarrow \text{constant} \cdot u_0$$

$$K^{(n+1)} \rightarrow k_0$$

- After n iterations, the J-th mode error component is reduced by the factor $(k_J / k_0)^n$

- Since $1 > k_1/k_0 > k_2/k_0 > k_3/k_0 > \ldots$,
  after the initial transient, error in $\Psi^{(n)}$ is dominated by first mode:

$$\Psi^{(n)} \approx [\text{cons}\,\text{tan}\,t] \bullet \left[\vec{u}_0 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^n \cdot \vec{u}_1 + \ldots \right]$$
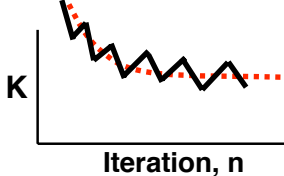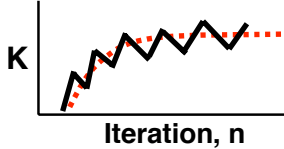
- (k_1 / k_0) is called the <u>dominance ratio</u>, DR or ρ

    – Errors die off as $\sim (DR)^n$

    – To reduce 10% error → .1% error

        DR~.9   →   44 iterations
        DR~.99 →   458 iterations
        DR~.999 → 2301 iterations

Initial guess
Exact solution

8 -13
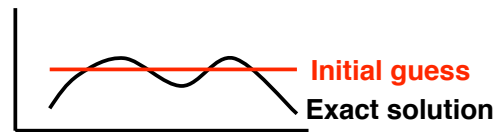
Typical K-effective convergence patterns

- Higher mode error terms die out as $(k_J / k_0)^n$, for n iterations

- When initial guess is concentrated in center of reactor, initial K_{eff} is too high (underestimates leakage)

K

Iteration, n

- When initial guess is uniformly distributed, initial K_{eff} is too low (overestimates leakage)

K

Iteration, n

- The **Sandwich Method** uses 2 K_{eff} calculations - one starting too high & one starting too low.
  Both calculations should converge to the same result.

8 -14

$$\Psi^{(n+1)} \approx [\text{constant}] \cdot \left[\vec{u}_0 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot \vec{u}_1 + \ldots \right]$$

$$K^{(n+1)} \approx k_0 \cdot \frac{\left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot G_1 + \ldots \right]}{\left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n} \cdot G_1 + \ldots \right]} \approx k_0 \cdot \left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n+1} \cdot G_1 \right] \cdot \left[1 - \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n} \cdot G_1 \right]$$

$$\approx k_0 \cdot \left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_1}{k_0} \right)^{n} \cdot \left(\frac{k_1}{k_0} - 1 \right) \cdot G_1 + \ldots \right]$$

- **For problems with a high dominance ratio (e.g., DR ~ .99), the error in K_{eff} may be small, since the factor $(k_1/k_0 - 1)$ is small.**

⇒ **K_{eff} may appear converged, even if the source distribution is <u>not</u> converged**

- **Keff is an integral quantity – converges faster than source shape**

Keff calculation for 2 nearly symmetric slabs, with Dominance Ratio = .9925

Power Iteration – Convergence

Noise (fluctuation)

Exact solution
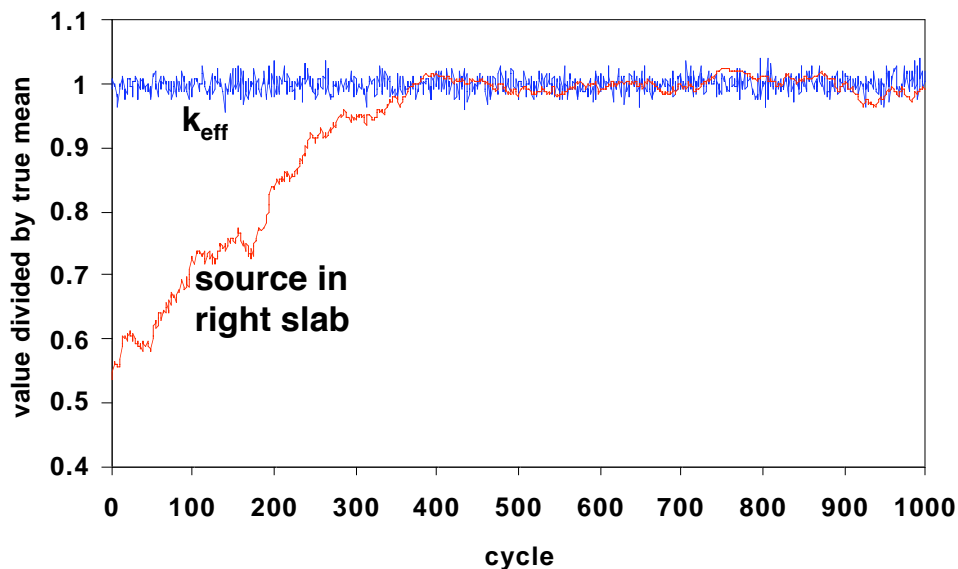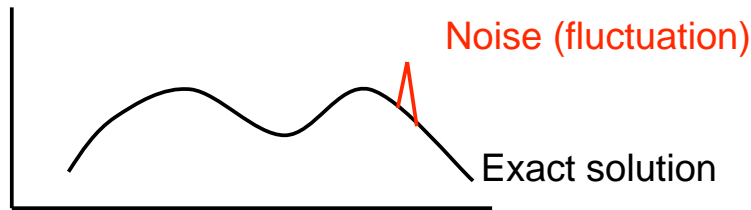
- For Monte Carlo power iteration,
  statistical fluctuations in source shape die out gradually over a number of successive iterations.
  - **Persistence of the noise over successive iterations gives correlation among source distributions in successive iterations. (Positive correlation)**

  - **Correlation directly affects confidence intervals:**
    Serial correlation in the source distribution ➜ larger confidence intervals

$\Rightarrow$ **Most Monte Carlo codes ignore these correlation effects & incorrectly underestimate the confidence intervals**

Power Iteration – Convergence

Summary

- <u>**Local errors in the source distribution**</u> **decay as** $(k_J/k_0)^n$
  - **Higher eigenmodes die out rapidly, convergence dominated by** k_1/k_0
  - **High DR** $\rightarrow$ **slow convergence**
  - **High DR** $\rightarrow$ **large correlation** $\rightarrow$ **large error in computed variances**

- <u>**Errors in K_{eff}**</u> **decay as** $(k_J/k_0 - 1) * (k_J/k_0)^n$
  - **High DR** $\rightarrow$ $k_J/k_0 \sim 1$ $\rightarrow$ **small error**

- **K_{eff} errors die out faster than local source errors**
  - **K_{eff} is an integral quantity – positive & negative fluctuations cancel**

- **High DR is common for**
  - Large reactors, with small leakage
  - Heavy-water moderated or reflected reactors
  - Loosely-coupled systems

$\Rightarrow$ **If <u>local</u> tallies are important (e.g., assembly power, pin power, …), examine their convergence – not just K_{eff} convergence**

Keff Calculations – Stationarity Diagnostics

- **Plots of single-cycle Keff or cumulative Keff are sometimes difficult to interpret when assessing convergence**

Cycle k_{eff}

50000 histories per cycle
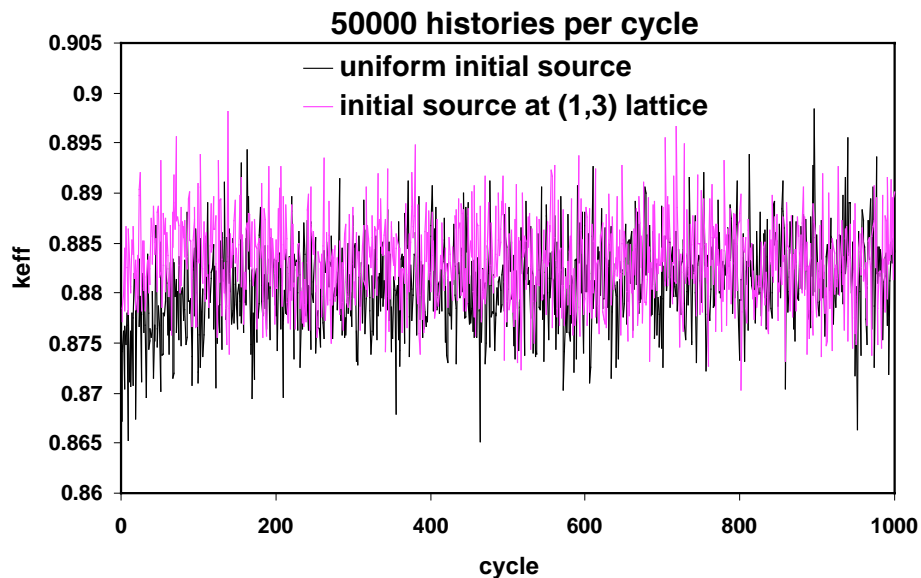
Keff Calculations – Stationarity Diagnostics

- **Initial cycles of a Monte Carlo K-effective calculation should be discarded, to avoid contaminating results with errors from initial guess**
  - **How many cycles should be discarded?**
  - **How do you know if you discarded enough cycles?**
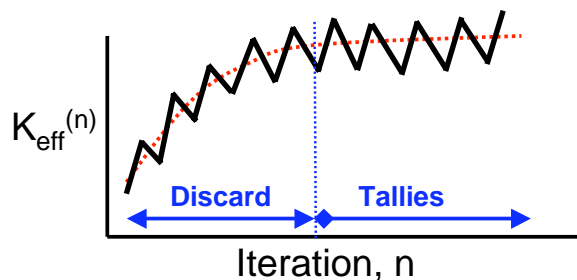
- **Analysis of the power iteration method shows that Keff is not a reliable indicator of convergence** — K_{eff} can converge faster than the source shape

- **Based on concepts from information theory, Shannon entropy of the source distribution is useful for characterizing the convergence of the source distribution**

· Divide the fissionable regions of the problem into **N$_S$ spatial bins**

   – Spatial bins should be consistent with problem symmetry

   – Typical choices: — 1 bin for each assembly

                          — regular grid superimposed on core

   – Rule-of-thumb for number of spatial bins:

       N$_S$ ~ (histories/batch) / 25   or   less

       Why?

        · Would like to have >25 fission source sites per bin to get good statistics

        · If source distribution were uniform, ~25 sites would be in each bin

· **Shannon entropy of the source distribution**

$$H(S) = -\sum_{J=1}^{N_S} p_J \cdot \ln_2(p_J), \quad \text{where} \quad p_J = \frac{(\text{\# source particles in bin } J)}{(\text{total \# source particles in all bins})}$$

· **Shannon entropy of the source distribution**

$$H(S) = -\sum_{J=1}^{N_S} p_J \cdot \ln_2(p_J), \quad \text{where} \quad p_J = \frac{(\text{\# source particles in bin } J)}{(\text{total \# source particles in all bins})}$$

   – **0 ≤ H(S) ≤ ln$_2$(N$_S$)**

   – Note that $p_J \ln_2(p_J) = 0$ if p_J=0

   – **For a uniform source distribution**, $p_1 = p_2 = \ldots = p_{Ns} = 1/N_S$,
     so that   **H(S) = ln$_2$(N$_S$)**

   – **For a point source (in a single bin), H(S) = 0**

· **H(S$^{(n)}$) provides a single number to characterize the source distribution for iteration n**

   – **As the source distribution converges in 3D space,
    a line plot of H(S$^{(n)}$) vs. n (the iteration number) converges**

K$_{eff}$ Calculations – Stationarity Diagnostics
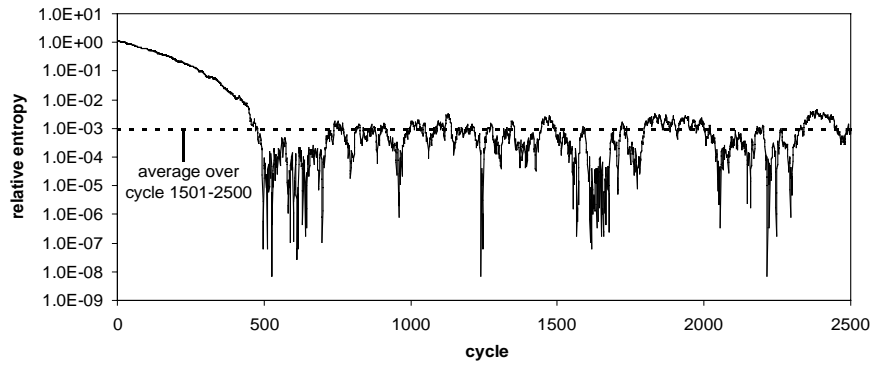
Figure 3: Posterior computation of relative entropy assuming
the true source is the mean source over 1501-2500 cycles (problem 1)
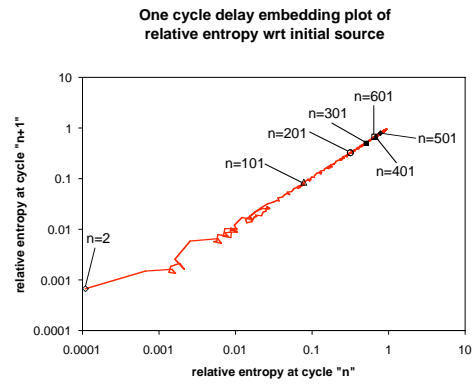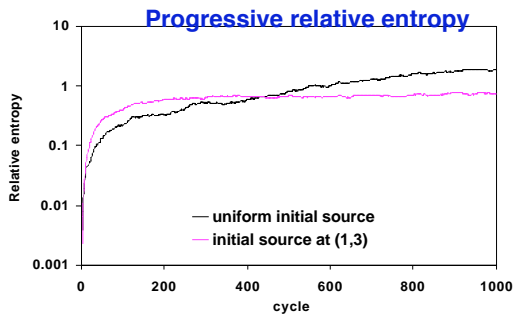
K$_{eff}$ Calculations – Stationarity Diagnostics

- **Example – Reactor core (Problem inp24)**
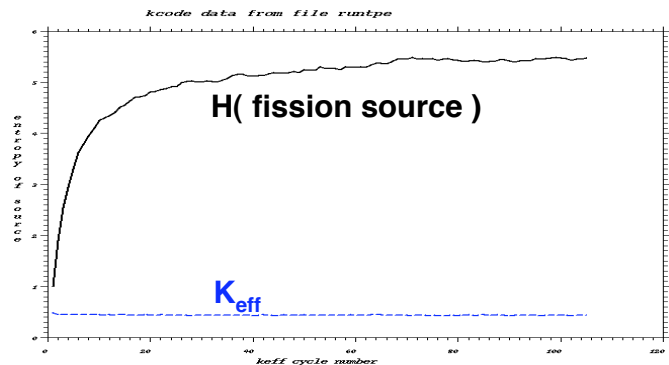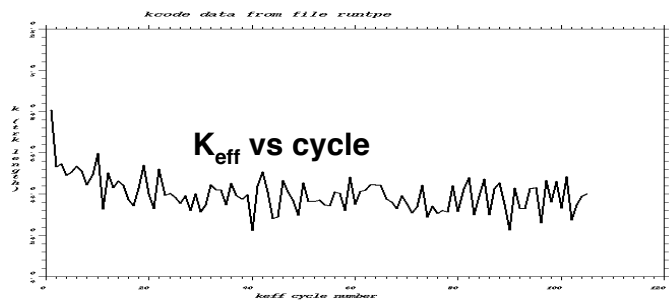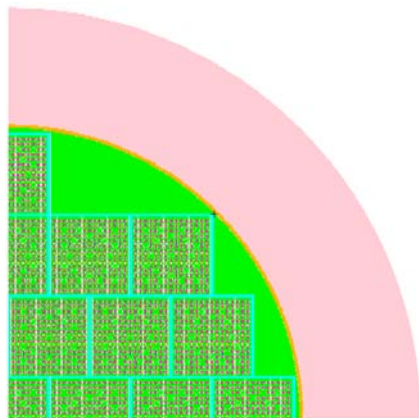
K_{eff} Calculations – Stationarity Diagnostics

- **Example – Loosely-coupled array of spheres (Problem test4s)**

8 -25
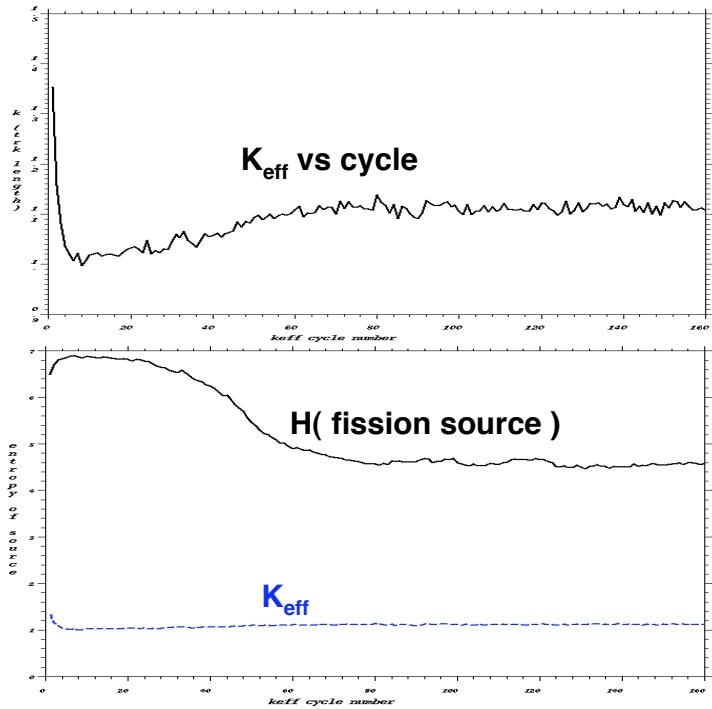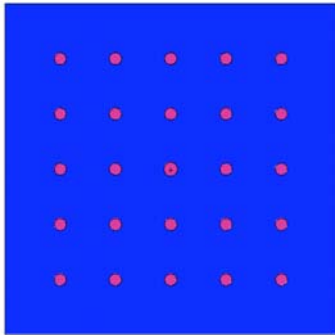
K_{eff} Calculations – Stationarity Diagnostics

- **Example – Fuel Storage Vault (Problem OECD_bench1)**

8 -26
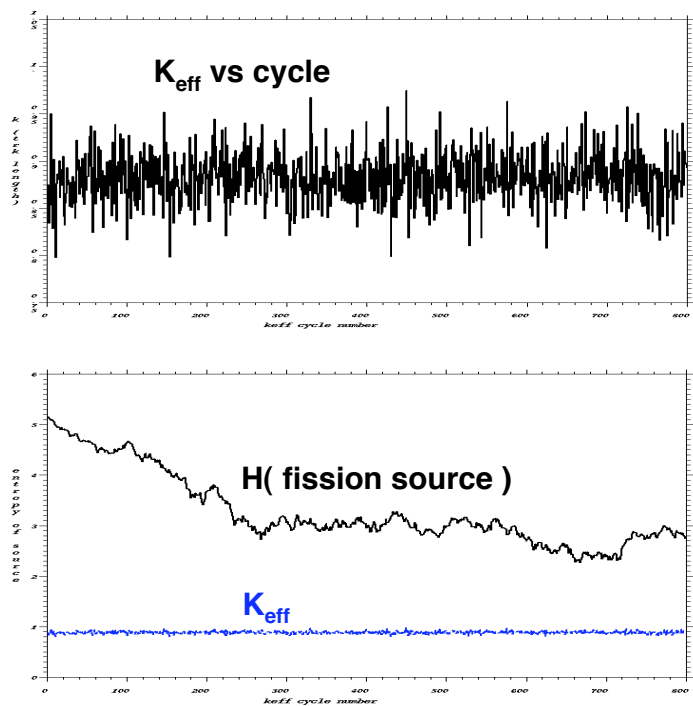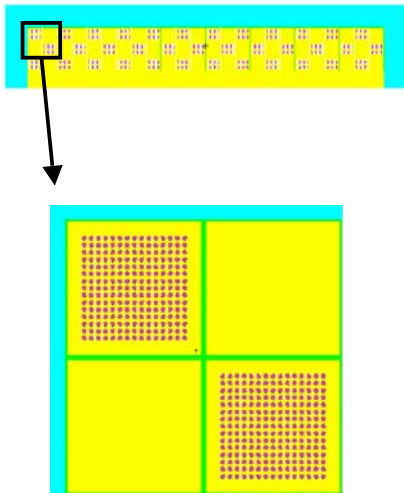
- Basic transport equation for eigenvalue problems

$$(L + T - S)\Psi = \tfrac{1}{K_{eff}}M\Psi$$

  L = loss to leakage   S = gain from scatter-in

  T = loss to collisions   M = gain from fission multiplication

- Define a fixed parameter **k_e** such that **$k_e > k_0$** (k_0 = exact eigenvalue)

- Subtract $\tfrac{1}{k_e}M\Psi$ from each side of the transport equation

$$(L + T - S - \tfrac{1}{k_e}M)\Psi = (\tfrac{1}{K_{eff}} - \tfrac{1}{k_e})M\Psi$$

- Solve the modified transport equation by power iteration

$$(L + T - S - \tfrac{1}{k_e}M)\Psi^{(n+1)} = (\tfrac{1}{K_{eff}^{(n)}} - \tfrac{1}{k_e})M\Psi^{(n)}$$

8 - 27

- **Power iteration for modified transport equation**

$$(L + T - S - \tfrac{1}{k_e}M)\Psi^{(n+1)} = (\tfrac{1}{K_{eff}^{(n)}} - \tfrac{1}{k_e})M\Psi^{(n)}$$

$$\Psi^{(n+1)} = (\tfrac{1}{K_{eff}^{(n)}} - \tfrac{1}{k_e}) \cdot (L + T - S - \tfrac{1}{k_e}M)^{-1}M\Psi^{(n)}$$

$$\Psi^{(n+1)} = \tfrac{1}{\tilde{K}^{(n)}} \cdot \tilde{F}\Psi^{(n)}$$

  where $\tilde{K}^{(n)} = (\tfrac{1}{K_{eff}^{(n)}} - \tfrac{1}{k_e})^{-1}$ or $K_{eff}^{(n)} = (\tfrac{1}{\tilde{K}^{(n)}} + \tfrac{1}{k_e})^{-1}$

- **How to choose k_e**
  - **k_e must be larger than k_0** (but, don't know k_0!)
  - **k_e must be held constant for all of the histories in a batch, but can be adjusted between batches**
    - Typically, guess a large initial value for k_e, such as k_e=5 or k_e=2
    - Run a few batches, keeping k_e fixed, to get an initial estimate of K_{eff}
    - Adjust k_e to a value slightly larger than the estimated K_{eff}
    - Run more batches, possibly adjusting k_e if the estimated K_{eff} changes

8 - 28

Wielandt Method

- **Convergence**
  - Eigenfunctions for the Wielandt method are same as for basic power iteration
  - Eigenvalues are shifted:

$$\tilde{k}_J = \left[\tfrac{1}{k_J} - \tfrac{1}{k_e} \right]^{-1} \qquad k_e > k_0 > k_1 > \ldots$$

  - Expand the initial guess, substitute into Wielandt method, rearrange to:

$$\Psi^{(n+1)} \approx [\text{constant}] \cdot \left[\vec{u}_0 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_e - k_0}{k_e - k_1} \cdot \frac{k_1}{k_0} \right)^{n+1} \cdot \vec{u}_1 + \ldots \right]$$

$$K^{(n+1)} \approx k_0 \cdot \left[1 + \left(\frac{a_1^{(0)}}{a_0^{(0)}} \right) \cdot \left(\frac{k_e - k_0}{k_e - k_1} \cdot \frac{k_1}{k_0} \right)^{n} \cdot \left(\frac{k_e - k_0}{k_e - k_1} \cdot \frac{k_1}{k_0} - 1 \right) \cdot G_1 + \ldots \right]$$

  - **Additional factor $(k_e{-}k_0)/(k_e{-}k_1)$ is less than 1 and positive, so that the red terms die out faster than for standard power iteration**

8 -29

Wielandt Method

- The **dominance ratio** for this modified power iteration is

$$DR' = \frac{\tilde{k}_1}{\tilde{k}_0} = \frac{[\tfrac{1}{k_1} - \tfrac{1}{k_e}]^{-1}}{[\tfrac{1}{k_0} - \tfrac{1}{k_e}]^{-1}} = \frac{k_e - k_0}{k_e - k_1} \cdot \frac{k_1}{k_0} = \frac{k_e - k_0}{k_e - k_1} \cdot DR$$

  - Since $k_e > k_0$ and $k_0 > k_1$, **DR' < DR**
  - DR of Wielandt method is always **smaller** than standard power iteration

- **Wielandt acceleration improves the convergence rate of the power iteration method for solving the k-eigenvalue equation**
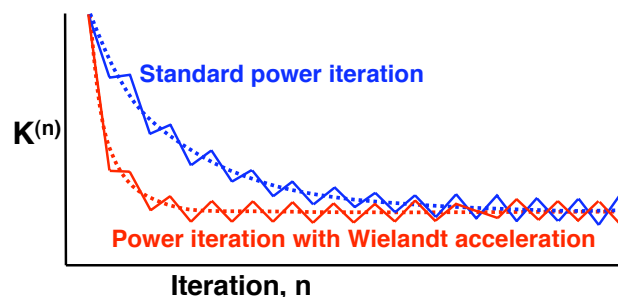
$\Rightarrow$ **Weilandt method converges at a faster rate than power iteration**

8 -30

Wielandt Method

- **Monte Carlo procedure for Wielandt acceleration**

$$(L + T - S - \tfrac{1}{k_e}M)\Psi^{(n+1)} = (\tfrac{1}{K_{eff}^{(n)}} - \tfrac{1}{k_e})M\Psi^{(n)}$$

- **For standard Monte Carlo (power iteration) in generation n+1**
  - When a collision occurs, the expected number of fission neutrons produced is

$$n_F = \left\lfloor wgt \cdot \frac{\nu\Sigma_F}{\Sigma_T} \cdot \frac{1}{K^{(n)}} + \xi \right\rfloor$$

  - **Store n_F copies of particle in the "fission bank"**
  - **Use the fission bank as the source for the next generation (n+2)**
- **For Monte Carlo Wielandt method in generation n+1**
  - When a collision occurs, compute 2 expected numbers of fission neutrons

$$n_F' = \left\lfloor wgt \cdot \frac{\nu\Sigma_F}{\Sigma_T} \cdot \left(\frac{1}{K^{(n)}} - \frac{1}{k_e} \right) + \xi \right\rfloor \qquad n_e' = \left\lfloor wgt \cdot \frac{\nu\Sigma_F}{\Sigma_T} \cdot \frac{1}{k_e} + \xi \right\rfloor$$

  - Note that $E[\, n_F' + n_e' \,] = E[\, n_F \,]$
  - **Store n_F' copies of particle in the "fission bank"**
  - **Follow n_e' copies of the particle in the current generation (n+1)**
  - **Use the fission bank as the source for the next generation (n+2)**

Wielandt Method

- <u>**Power iteration**</u> **for Monte Carlo k-effective calculation**

Wielandt Method

- **Wielandt method** for Monte Carlo k-effective calculation
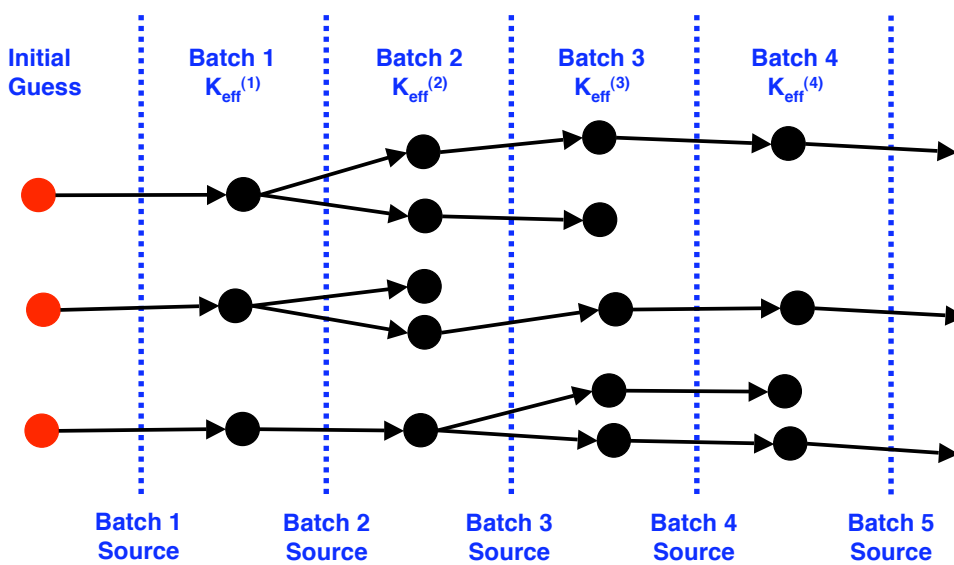
Source particle generation

Monte Carlo random walk

Neutron

Additional Monte Carlo random walks within generation due to Wielandt method

---
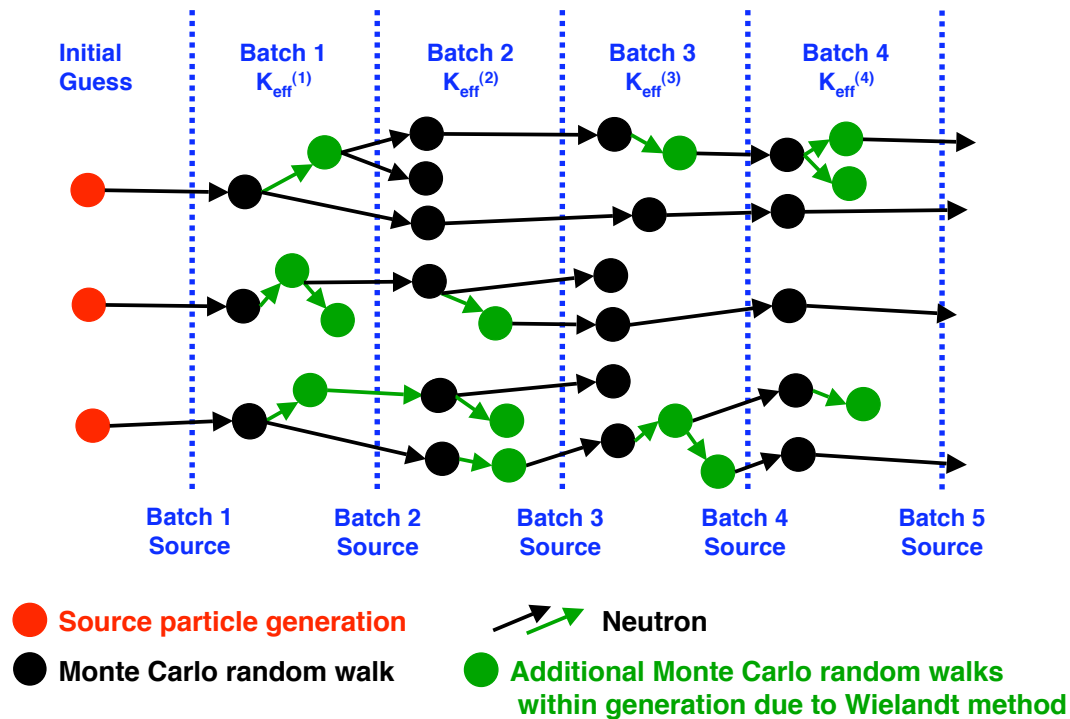
Wielandt Method

Summary

- **Wielandt Method has a lower DR than power iteration**
  - **Faster convergence rate than power iteration ⇒ fewer iterations**

  - **Some of the particle random walks are moved from the next generation into the current generation ⇒ more work per iteration**

  - **Same total number of random walks ⇒ no reduction in CPU time**

- **Advantages**
  - **Reduced chance of false convergence for very slowly converging problems**

  - **Reduced inter-generation correlation effects on variance**

  - **Fission source distribution spreads more widely in a generation (due to the additional particle random walks), which should result in more interactions for loosely-coupled problems**

Superhistory Method

- **Standard generation model, solved by power iteration**

$$\Psi^{(n+1)} = \frac{1}{K_{eff}^{(n)}} \cdot F\Psi^{(n)}$$

- **Superhistory method**
  - **Follow several generations (L) before recomputing K_{eff} and renormalizing**

$$\Psi^{(n+1)} = \frac{1}{\tilde{K}^{(n)}} \cdot \tilde{F}\Psi^{(n)}, \qquad \text{with} \quad \tilde{F} = F^L, \quad \tilde{K}^{(n)} = (K_{eff}^{(n)})^L$$

- **Convergence**
  - Same eigenfunctions as standard power iteration
  - Eigenvalues are k_0^L, k_1^L, k_2^L, ...
  - DR' = DR^L, where DR = dominance ratio for power iteration
  - Fewer iterations, but L generations per iteration $\Rightarrow$ same work as power iteration
  - Same convergence rate as power iteration
- **Advantages**
  - Reduced correlation between iterations
  - Fewer renormalizations

Superhistory Method

- <u>**Superhistory Method**</u> **for Monte Carlo k-effective calculation**

Example with L = 2 generations/batch

References
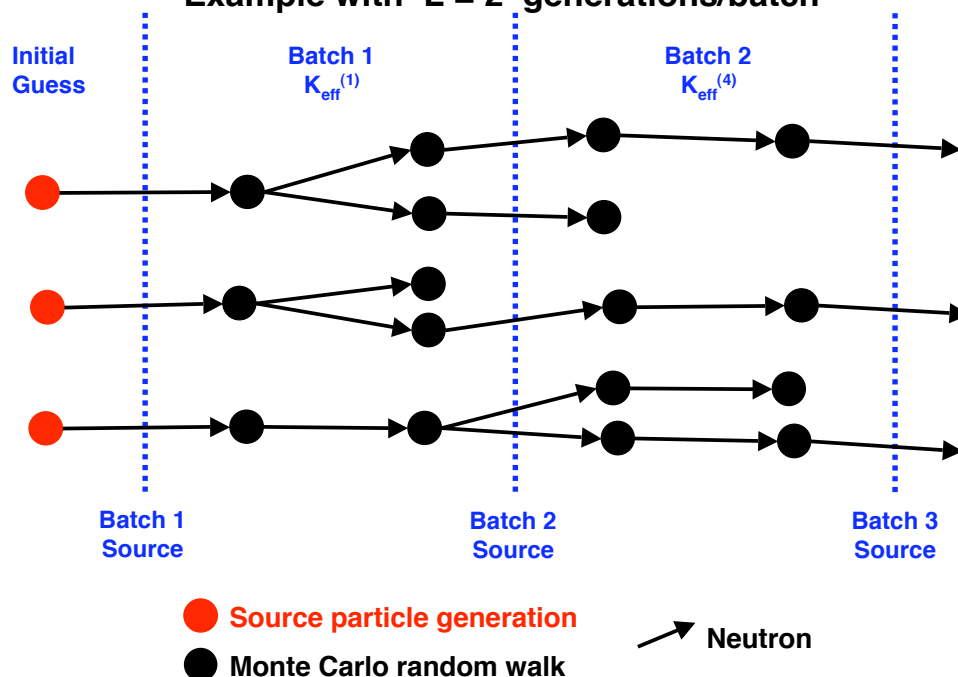
Monte Carlo k-effective Calculations

J. Lieberoth, "A Monte Carlo Technique to Solve the Static Eigenvalue Problem of the Boltzmann Transport Equation," *Nukleonik* **11**,213 (1968).

M. R. Mendelson, "Monte Carlo Criticality Calculations for Thermal Reactors," *Nucl. Sci Eng.* **32**, 319–331 (1968).

H. Rief and H. Kschwendt, "Reactor Analysis by Monte Carlo," *Nucl. Sci. Eng.*, **30**, 395 (1967).

W. Goad and R. Johnston, "A Monte Carlo Method for Criticality Problems," *Nucl. Sci. Eng.* **5**, 371–375 (1959).

Superhistory Method

R.J. Brissenden and A.R. Garlick, "Biases in the Estimation of Keff and Its Error by Monte Carlo Methods," *Ann. Nucl. Energy*, Vol 13, No. 2, 63–83 (1986).

Wielandt Method

T Yamamoto & Y Miyoshi, "Reliable Method for Fission Source Convergence of Monte Carlo Criticality Calculation with Wielandt's Method", *J. Nuc. Sci. Tech.*, **41**, No. 2, 99-107 (Feb 2004).

S Nakamura, Computational Methods in Engineering and Science, R. E. Krieger Pub. Company, Malabar, FL (1986).

Sandwich Method

J Yang & Y. Naito, "The Sandwich Method for Determining Source Convergence in Monte Carlo Calculations", *Proc. 7th Int. Conf. Nuclear Criticality Safety, ICNC2003, Tokai-mura, Iburaki, Japan, Oct 20–24, 2003*, JAERI-Conf 2003–019, 352 (2003).

References

Stationarity Diagnostics & Correlation in Monte Carlo k-effective Calculations

T. Ueki, "Intergenerational Correlation in Monte Carlo k-Eigenvalue Calculation," *Nuc. Sci. Eng.,* **141**, 101 (2002).

T. Ueki and F.B. Brown, "Autoregressive Fitting for Monte Carlo K-effective Confidence Intervals", *Trans. Am. Nuc. Soc.* **86**, 210 (2002).

T. Ueki and F.B. Brown, "Stationarity Diagnostics Using Shannon Entropy in Monte Carlo Criticality Calculation I: F test," *Trans. Am. Nuc,* **87**, 156 (2002).

T. Ueki and F.B. Brown, "Stationarity and Source Convergence Diagnostics in Monte Carlo Criticality Calculation," *proceedings of M&C 2003, ANS Topical Meeting, Gatlinburg, Tennessee* (April, 2003).

T. Ueki, F.B. Brown and D.K. Parsons, "Dominance Ratio Computation via Time Series Analysis of Monte Carlo Fission Sources," *Trans. Am. Nuc,* **88**, 309 (2003).

T. Ueki and F.B. Brown, "Informatics Approach to Stationarity Diagnostics of the Monte Carlo Fission Source Distribution," *Trans. Am. Nuc,* **89**, 458 (2003).

T. Ueki, F.B. Brown, D.K. Parsons, and D.E. Kornreich, "Autocorrelation and Dominance Ratio in Monte Carlo Criticality Calculations," *Nuc. Sci. Eng.,* **145**, 279 (2003).

T. Ueki, F.B. Brown, D.K. Parsons, and J.S. Warsa, "Time Series Analysis of Monte Carlo Fission Sources: I. Dominance Ratio Computation," *Nuc. Sci. Eng.,* **148**, 374 (2004).

T. Ueki, "Entropy and Undersampling in Monte Carlo Criticality Calculations," *Trans. Am. Nuc,* **91**, 119 (2004).

T. Ueki, "Time Series Modeling and MacMillan's Formula for Monte Carlo Iterated-Source Methods," *Trans. Am. Nuc*, **90**, 449 (2004).

T. Ueki, "Principal Component Analysis for Monte Carlo Criticality/Eigenvalue Calculations," *Trans. Am. Nuc*, **90**, 461 (2004).

T. Ueki and F.B. Brown, "Stationarity Modeling and Informatics-Based Diagnostics in Monte Carlo Criticality Calculations," *Nuc. Sci. Eng.,* **149**, 38 (2005).

T. Ueki, "Asymptotic Equipartition Property and Undersampling Diagnostics in Monte Carlo Criticality Calculations," *proceedings of MC 2005, ANS Topical Meeting in Monte Carlo, Chattanooga, TN* (2005).

T. Ueki, "Information Theory and Undersampling Diagnostics for Monte Carlo Simulation of Nuclear Criticality," *Nuc. Sci. Eng.,* in press.

T. Ueki, "Time Series Analysis of Monte Carlo Fission Sources: II. k-effective Confidence Interval," *Nuc. Sci. Eng.,* submitted

Variance Reduction

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

Monte Carlo Calculations

Geometry
• Which cell is particle in?
• What will it hit next?
• How far to boundary?
• What's on other side?
• Survival?

Physics
• How far to collision?
• Which nuclide?
• New E, direction?
• Secondaries?
• Survival?

Tallies
• Tally events of interest
• Compute results
• Compute statistics
• Balances
• Performance stats
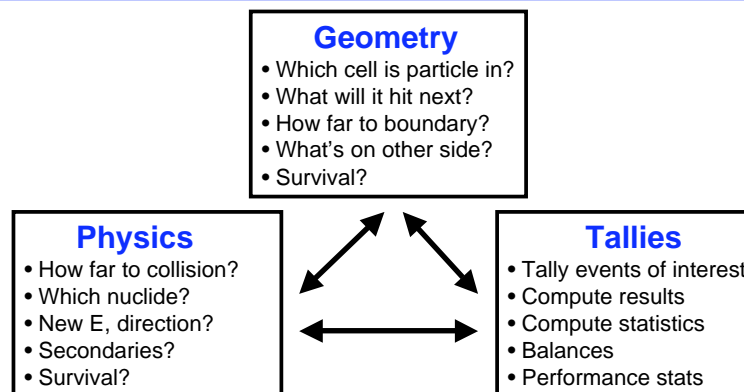
mcnp, rcp, vim, racer, sam-ce, tart, morse, keno, tripoli, mcbend, monk, o5r, recap, andy,…..

- **Variance reduction**
  - **Modify the PDFs for physics interactions to favor events of interest**
  - **Use splitting/rouletting to increase particles in certain geometric regions**
  - **Kill particles in uninteresting parts of problem**
- **May be necessary in order to sample rare events**
- **More samples (with less weight each) —> smaller variance in tallies**

Monte Carlo Estimates of Integrals

Given a function R(x), where x is a random variable with PDF f(x),

- Expected value of R(x) is
$$\mu = \int R(x)\, f(x)\, dx$$

- Variance of R(x) is
$$\sigma^2 = \int R^2(x)\, f(x)\, dx - \mu^2$$

Monte Carlo method for estimating μ
- make N random samples $\hat{x}_j$ from f(x)
- Then

$$\overline{R} \approx \frac{1}{N}\sum_{j=1}^{N} R(\hat{x}_j)$$

$$\sigma_{\overline{R}}^2 \approx \frac{1}{N-1}\cdot\left(\frac{1}{N}\sum_{j=1}^{N} R^2(\hat{x}_j) - \overline{R}^2\right)$$

Variance Reduction – Basic Idea

•Expected mean score is not changed by variance reduction

$$\mu = \int R(x)\, f(x)\, dx = \int R(x)\left(\frac{f(x)}{g(x)}\right)\cdot g(x)\, dx$$

•Sample x' from f(x)
•Tally R(x')

•Sample x' from g(x)
•Tally R(x') • f(x')/g(x')

•Variance is changed due to altered sampling scheme

$$\sigma^2 = \int [R(x)]^2 f(x)\, dx - \mu^2 \qquad\qquad \sigma^2 = \int \left[R(x)\frac{f(x)}{g(x)}\right]^2 g(x)\, dx - \mu^2$$

Goal: Choose g(x) such that variance is reduced

- Given a set of random samples, $x_1, x_2, \ldots, x_N,$
    - Mean

$$\overline{x} = \frac{1}{N}\sum_{j=1}^{N}x_j$$

    - Variance of the mean

$$\sigma_{\overline{x}}^2 = \frac{1}{N-1}\cdot\left(\frac{1}{N}\sum_{j=1}^{N}x_j^2 - \overline{x}^2\right)$$

    - Relative Error

$$RE = \frac{\sigma_{\overline{x}}}{\overline{x}}$$

    - Figure of Merit

$$FOM = \frac{1}{RE^2\cdot T}$$

- Variance reduction: Reduce RE or T, to increase FOM

- **Analog Monte Carlo**
    - Faithful simulation of particle histories
    - No alteration of PDFs (i.e., no biasing or variance reduction)
    - Particle is born with weight = 1.0
    - Weight unchanged throughout history until particle is killed
    - Scores are weighted by **1.0** when tallying events of interest

- **Weighted Monte Carlo (non-analog)**
    - Alter the PDFs to favor events of interest
    - Particle is born with weight = 1.0
    - Weight, **wgt**, is altered if biased PDF is used
    - Weight can also be changed by Russian roulette/splitting & other variance reduction techniques
    - Scores are weighted by **wgt** when tallying events of interest

Variance Reduction – General Approaches

- **Truncation**
  - Remove particles from parts of phase space that do not contribute significantly to the tallies

- **Population control**
  - Use particle splitting and Russian rouletting to control the number of samples taken in various regions of phase space

- **Modified sampling**
  - Modify the PDFs representing problem physics, to favor tallies of interest

- **Deterministic methods**
  - Replace portions of a particle random walk by the expected results obtained from a deterministic calculation

Typical Variance Reduction Techniques

- **MCNP has 14 variance reduction techniques**
  1. Time and energy cutoffs
  2. Geometry splitting & roulette
  3. Weight windows
  4. Exponential transform
  5. Forced collisions
  6. Energy splitting & roulette
  7. Time splitting & roulette
  8. Point and ring detectors
  9. DXTRAN
  10. Implicit capture
  11. Weight cutoff
  12. General source biasing
  13. Secondary particle biasing
  14. Bremsstrahlung energy biasing

Survival Biasing

- Also called implicit absorption or non-absorption weighting

- **Modify collision process** according to expected outcome

- Particle always survives collision
    - Tally expected absorption, $\qquad$ wgt $\cdot$ (σ_A/σ_T)
    - Reduce weight of surviving particle, $\qquad$ wgt' = wgt $\cdot$ $(1 - \sigma_A/\sigma_T)$

- Extends particle history so that more particles reach events which occur after many collisions

- Most effective for thermal reactor problems, but doesn't hurt in other types of problems

- Must also use some form of low-weight cutoff to eliminate particles with very low weight

Geometry Splitting & Russian Roulette

- Increase the number of particles in "important" regions, decrease the number of particles in "unimportant" regions

- Assign each cell an importance, I_{cell}
    - Arbitrary, use best guess or adjoint fluxes from deterministic calculation
    - Could use one value for all energies or separate values for different energy ranges
    - Higher value —> more important
    - $I_{cell} > 0$
    - $I_{cell}=0$ is a way to declare regions as not in physical problem
    - Values of I_{cell} must not change during Monte Carlo calculation

- Modify random walk simulation at surface crossings:
    - **If $(I_{enter}/I_{leave}) > 1$,   perform splitting**
    - **If $(I_{enter}/I_{leave}) < 1$,   perform Russian roulette**

Geometry Splitting & Russian Roulette

- Let $r = I_B / I_A$

  $n = \lfloor r \rfloor$

Cell B
I_B

Cell A
I_A

$I_B > I_A$
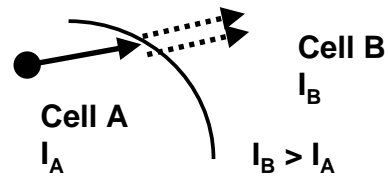
- If $n > 1$, **split** into n particles with weight (wgt/n)
  - All of the **n** particles emerging from splitting have identical attributes (e.g., x,y,z, u,v,w, E) including **wgt' = wgt/n**
  - All of the **n** particles from a splitting are part of the **same** history, and their tallies must be combined
  - Typically, (n-1) particles are banked, 1 particle is followed until its death, then a particle is removed from the bank & followed, etc.
- Avoid over-splitting
  - Splitting into a large number of particles can increase CPU-time & lead to (apparent) bias in results
  - Typically, choose cell importances to split 2-for-1 or 3-for-1
  - Typically, can limit the splitting to n-for-1 or less
- Total particle weight is exactly conserved in splitting

Geometry Splitting & Russian Roulette

- Let $r = I_B / I_A$

?

Cell B
I_B

Cell A
I_A

$I_B < I_A$
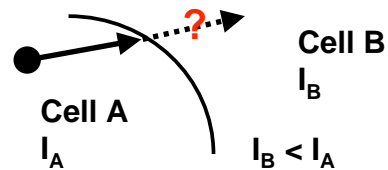
- If $r < 1$, play **Russian roulette**
  - With probability **r**, keep the particle & alter its weight to **(wgt/r)**
  - With probability **(1-r),** kill the particle (set its weight to 0)

  if $\xi < r$,
      wgt' = wgt/r
  else
      wgt' = 0

- Russian roulette effectively merges a number of low-weight particles into one with higher weight

- Total particle weight is only conserved statistically (expected value)

Weight Cutoff

- Specify a cutoff weight, W_{low}, and a survival weight, W_{ave}

- If particle weight drops below W_{low}, play Russian roulette with weight of W_{ave} for survivors
  - Probability of surviving RR = wgt/W_{ave}
  - Probability of being killed = $1 - wgt/W_{ave}$

  If   wgt < W_{low},

        if ξ < wgt/W_{ave},

            wgt' = W_{ave}

        else

            wgt' = 0

Particle Weight   ——— W_{ave}

           W_{low}

Set wgt to W_{ave}
Or kill   ??
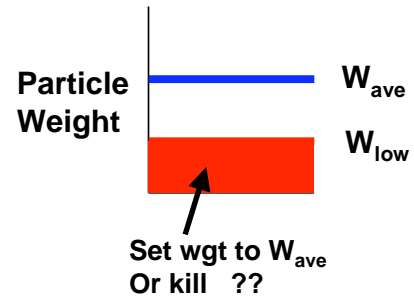
- Expected value of surviving weight is conserved, $(wgt/W_{ave}) \bullet W_{ave}$

Weight Cutoff

- In some codes (e.g., MCNP), the weight cutoff parameters are functions of cell importance

  - Let R_j = (importance of **source cell**) / (importance of **cell j**)

  - Then, $W_{ave}(j) = W_{ave} \bullet R_j$
    $W_{low}(j) = W_{low} \bullet R_j$

- Weight cutoffs reduce computing time, not variance

- Weight cutoffs can be applied anytime the particle weight changes – after collisions, after boundary crossings, …

Weight Windows
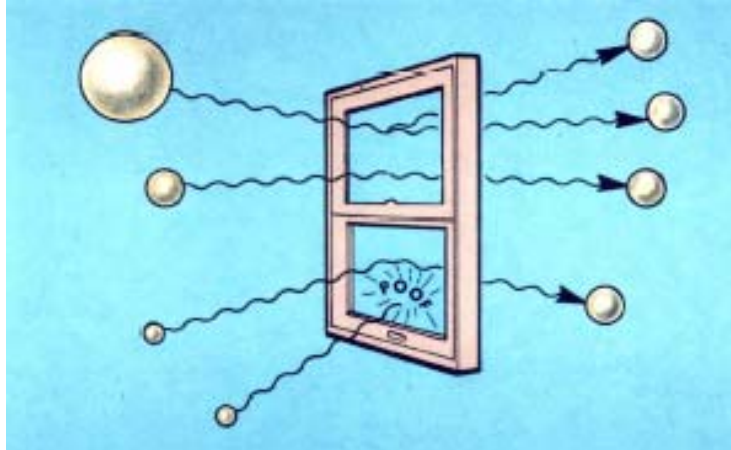
- Prevent particle weights from getting too large or too small
  - Weight too large —> splitting
  - Weight too small —> Russian Roulette

9 -15

Weight Windows

- Large fluctuations in particle weights contributing to a tally lead to larger variance

- Weight windows eliminate large or small weights (outside the window) by creating or destroying particles

- Weight windows can be applied any time – after collisions, after surface crossings, …

  If   wgt > W_{hi}
        splitting
  Elseif   wgt < W_{low}
        roulette

9 -16

Weight Windows

- MCNP weight window scheme

  Input: W_{low} for each cell (can be energy or time dependent),
                 $[W_{ave}/W_{low}]$, $[W_{hi}/W_{low}]$,   mxspln

  If   wgt > W_{hi}
        n = min(mxspln, 1 + wgt/W_{hi})   <— max splitting is mxspln-to-1
        wgt = wgt/n
        bank n-1 copies of particle   <— n-to-1 splitting

  Elseif   wgt < W_{low}
        P = max(1/mxspln, wgt/W_{ave})   <— limits survivor to mxspln*wgt
        if ξ < P
          wgt = wgt/P   <— particle survives
        else
          wgt = 0   <— particle killed

Source Biasing

- Bias the PDFs used to select the angle, energy, or position or source particles
  - Produce more source particles (with lower weights) in desired parts of phase space

  True source: $f(R,E,\Omega)$

  Sample (R',E',Ω') from $g(R,E,\Omega)$

     & assign weight $f(R',E',\Omega')/g(R',E',\Omega')$ to source particle

  Choose $g(R,E,\Omega)$ to favor directions more important to tallies

Forced Collisions

- Particles entering specified cells are split into collided & uncollided parts
  - For distance-to-boundary d

    Prob(no collision) = exp($-\Sigma_T$d)

    Prob(collision) = 1 – exp($-\Sigma_T$d)

$$\mathbf{wgt} \cdot \left(1 - e^{-\Sigma_T d}\right)$$

$$\mathbf{wgt} \cdot e^{-\Sigma_T d}$$

d
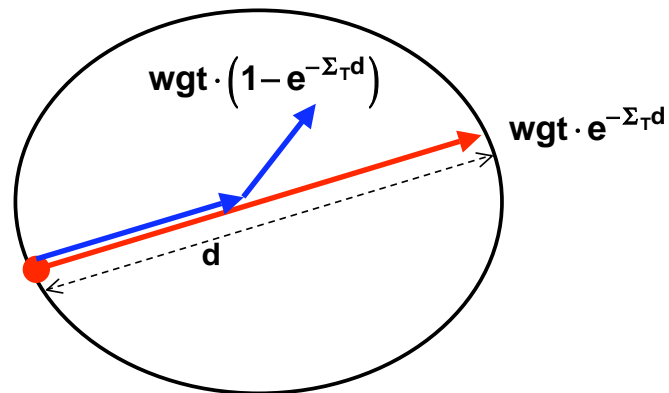
Forced Collisions

- Sampling the flight distance **s** for a forced collision with max flight distance **d**

  Sampling from a truncated exponential PDF:

$$\mathbf{f(s)} = \Sigma_T \cdot \frac{e^{-\Sigma_T s}}{1 - e^{-\Sigma_T d}}, \qquad \mathbf{0 \le s \le d}$$

$$\mathbf{F(s)} = \frac{1 - e^{-\Sigma_T s}}{1 - e^{-\Sigma_T d}}$$

Solve for s : $\quad \xi = \mathbf{F(s)}$

$$\mathbf{s} = \frac{-\ln\left[1 - (1 - e^{-\Sigma_T d})\xi\right]}{\Sigma_T}$$

Exponential Transform

- Encourage particles to head in a certain preferred direction, Ω_0

Ω

Source or Collision point - - - - - - - - - - - - - → **Detector** Ω_0
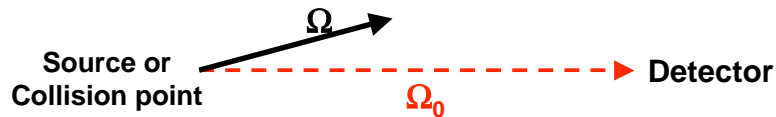
- Replace Σ_T by $\Sigma^* = \Sigma_T [1 - p\,\Omega\cdot\Omega_0]$
  - p = a parameter, $0<p<1$
  - Ω_0 = unit vector from particle position to detector
  - Ω = actual particle direction

- Sample flight distance **s'** from $g(s) = \Sigma^* \exp(-\Sigma^* s)$
- Adjust weight by factor:

$$f(s')/g(s') = \exp(-p\,\Omega\cdot\Omega_0\,\Sigma_T s')/[1 - p\,\Omega\cdot\Omega_0]$$

- **Paths toward detector are stretched** ($\Sigma^* < \Sigma_T$)
- **Paths away from detector are shortened** ($\Sigma^* > \Sigma_T$)

9 - 21

Variance Reduction Goals & Cautions

- Maximize FOM – either reduce RE or T

- Keep the number of particles per cell roughly constant from source to detector

- Reduce the number of particles in unimportant regions

- Achieve adequate sampling of all portions of phase space

- Avoid over-biasing (e.g., over-splitting)

- Ensure that tallies pass statistical checks

9 - 22

References

– T.E. Booth, "A Sample Problem for Variance Reduction in MCNP",
    LA-10363-MS, LANL Report (Oct 1985)

– X-5 Monte Carlo Team, "MCNP – A General Monte Carlo N-Particle Transport
    Code, Version 5", LA-UR-03–1987 (April 2003)

– L.L. Carter & E.D. Cashwell, "Particle Transport Simulation with the Monte
    Carlo Method", TID-26607, National Technical Information
    Service (1975)

Parallel

Monte Carlo

Forrest B. Brown
Diagnostics Applications Group (X-5)
Los Alamos National Laboratory

10 -1

Parallel Monte Carlo

- **Parallel Computing**
  - Parallel Computers
  - Message Passing
  - Threads
  - Amdahl's Law
- **Parallel Monte Carlo**
  - Parallel Algorithms
  - Histories, Random Numbers, Tallies
  - Load Balancing, Fault Tolerance, …
- **Parallel Monte Carlo Performance**
  - Performance Measures & Limits
  - Parallel Scaling
- **MCNP5 Parallel Processing**
  - MCNP5 parallelism
  - MPI or PVM + Threads
  - Run Commands & Input Options
  - Performance on ASCI Tera-scale systems
  - Parallel Processing for Large-scale Calculations

10 -2

Parallel Computing

Perspective

- **Fast desktop computers**

1980s super:	200 MHz	16 MB	10 GB	$ 20 M
Today, PC:	2000 MHz	1000 MB	100 GB	$ 2 K

- **Linux clusters + MPI**
    - **Cheap parallel computing**
    - **Everyone can do parallel computing, not just national labs**

- **Mature Monte Carlo codes**
    - **MCNP, VIM, KENO, MCBEND, MONK, COG, TART, RACER, RCP, …**

- **New generation of engineers/scientists**
    - **Less patience for esoteric theory & tedious computing procedures**
    - **Computers are tools, not to be worshipped**
    - **What's a slide rule ???**

➜ **More calculations with Monte Carlo codes**

Trends in Computing Technology

- **Commodity chips**
  - Microprocessor speed → ~2x gain / 18 months
  - Memory size → ~2x gain / 18 months
  - Memory latency → ~ no change (getting worse)

- **High-end scientific computing**
  - Key driver (or limit) → **economics:** mass production of desktop PCs & commercial servers

  - Architecture → **clusters:** with small/moderate number of commodity microprocessors on each node

- **Operating systems**
  - Desktop & server → Windows, Linux
  - Supercomputers → Unix, Linux

CPU performance on supercomputer → same as desktop PC

High-performance scientific computing → parallel computing

Parallel Computers

1 GFLOP = 10^9 FLOP
1 TFLOP = 10^{12} FLOP
1 PFLOP = 10^{15} FLOP

Teraflop computers → 1000's of processors

Petaflop computers → 10^5 processors ?

Parallel Computers
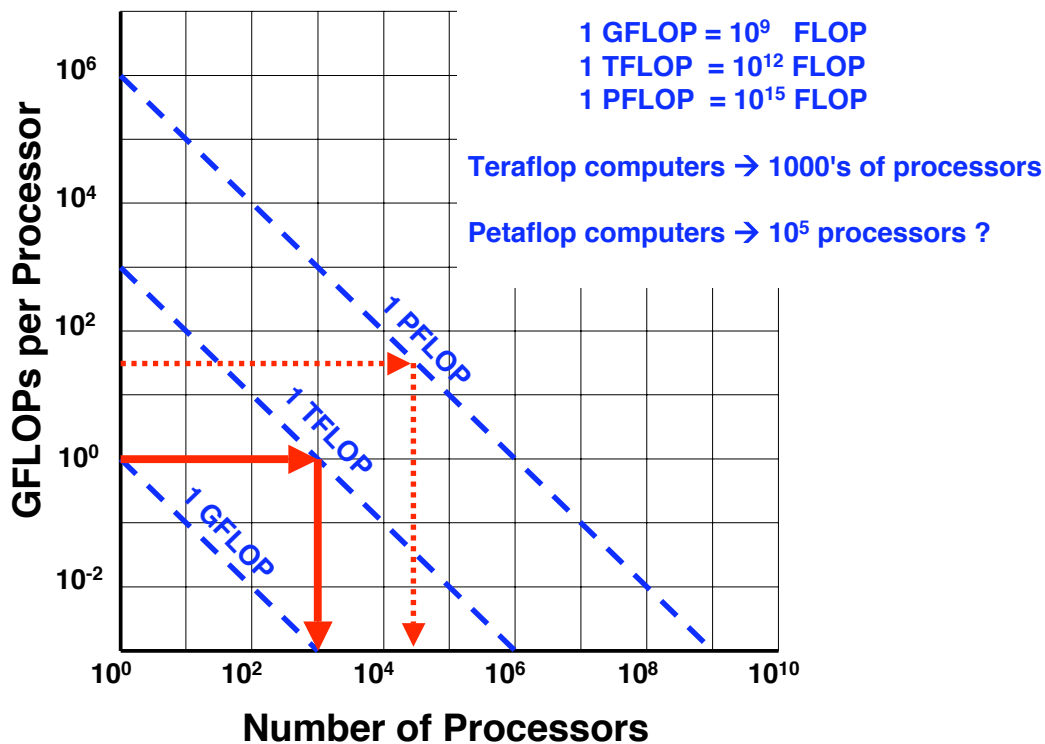
- **Characterize computers by:**
  - CPU: scalar, vector, superscalar, RISC, …..
  - Memory: shared, distributed, cache, banks, bandwidth, …..
  - Interconnects: bus, switch, ring, grid, …..

- **Basic types:**

Traditional

CPU — Mem

Shared Memory Parallel

CPU CPU ….. CPU — Mem

Distributed Memory Parallel

CPU — Mem, CPU — Mem, ….. CPU — Mem

Clustered Shared Memory

CPU CPU CPU CPU — Mem, CPU CPU CPU CPU — Mem, ….. CPU CPU CPU CPU — Mem

---
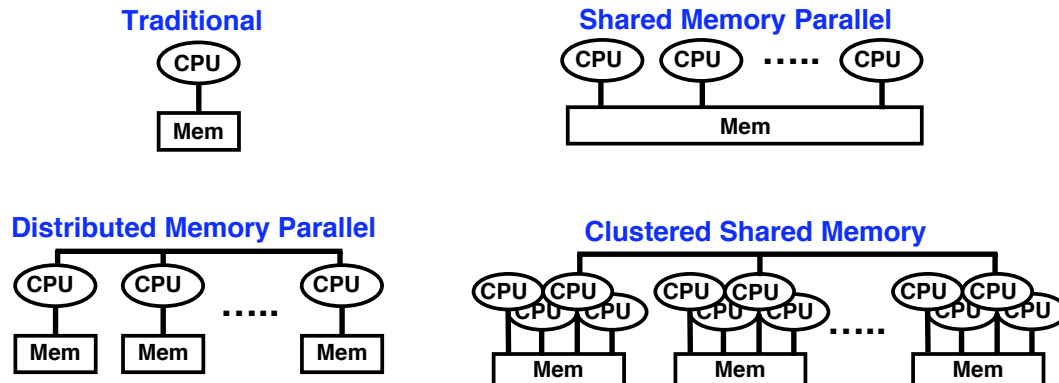
Approaches to Parallel Processing

High-level
- Independent programs + **message-passing**
- Distribute work among processors
- Loosely-coupled
- Programmer must modify high-level algorithms

Mid-level
- **Threads** (task-level)
- Independent tasks (subprograms) + **shared memory**
- For shared memory access, use locks on critical regions
- Compiler directives by programmers

Low-level
- **Threads** (loop-level)
- Split DO-loop into pieces, compute, synchronize
- Compiler directives by programmers

Low-level
- Pipelining or vectorization
- Pipelined execution of DO-loops
- Automatic vectorization by compilers &/or hardware,
      or compiler directives by programmers

Message-passing

Program B ← Lots of computation

Program A

Program B ← Interchange data via messages
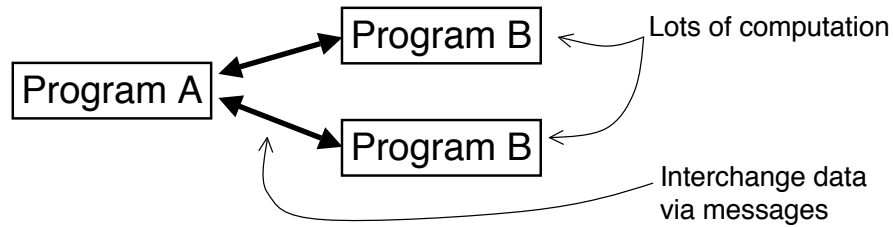
- – Independent programs
- – Separate memory address space for each program (private memory)
- – All control information & data must be passed between programs by explicit messages (SENDs & RECEIVEs)
- – Can run on distributed or shared memory systems
- – Efficient only when $T_{computation} \gg T_{messages}$
- – Standard message-passing:
  - • MPI
  - • PVM

Threading (task-level)

```
program A
  …..
  !$omp parallel
  call trnspt
  !$omp end parallel
  …..
end program A
```

```
subroutine trnspt
  …..
  return
end subroutine trnspt
```

```
subroutine trnspt
  …..
  return
end subroutine trnspt
```

Shared Data

Address space for Program A
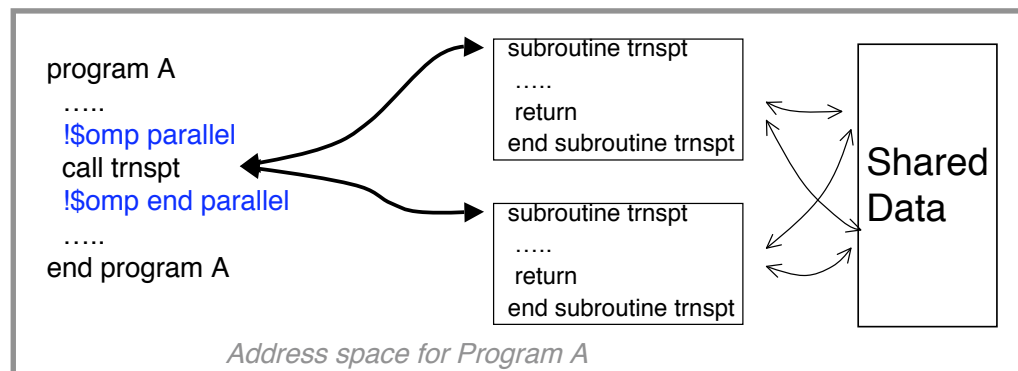
- – Single program, independent sections or subprograms
- – Each thread executes a portion of the program
- – Common address space, must distinguish private & shared data
- – Critical sections must be "locked"
- – Can run only on shared memory systems, not distributed memory
- – Thread control by means of compiler directives
- – Standard threading:
  - • OpenMP

Threading (loop-level)

```
!$omp do parallel
do k=1,n
   c(k) = a(k)+b(k)
enddo
```

```
do k=1,n,2
   c(k) = a(k)+b(k)
enddo
```

```
do k=2,n,2
   c(k) = a(k)+b(k)
enddo
```
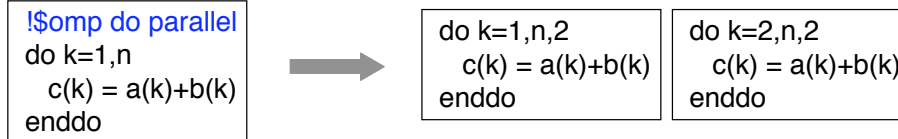
– Single DO-loop within program

– Each loop iteration must be independent

– Each thread executes different portion of DO-loop

– Invoked via compiler directives

– Standard threading:

  • OpenMP

Domain Decomposition

Decompose
Computational
Problem

Analyze
Subdomains
In parallel

**Collect
Problem
Results**
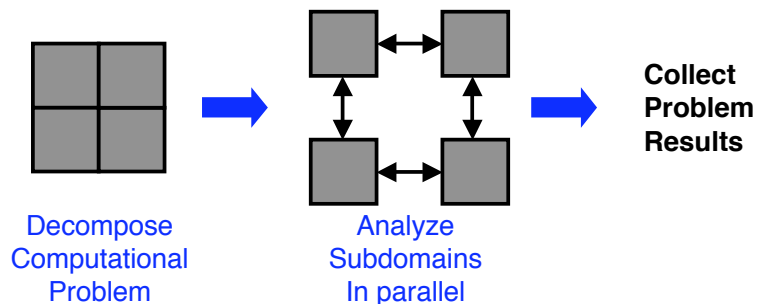
– Coarse-grained parallelism, high-level

– For mesh-based programs:
  1. Partition physical problem into blocks (domains)
  2. Solve blocks separately (in parallel)
  3. Exchange boundary values as needed
  4. Iterate on global solution

– Revised iteration scheme may affect convergence rates

– Domain decomposition is often used when the entire problem will not fit in the memory of a single SMP node

Amdahl's Law

If a computation has fast (parallel) and slow (scalar) components, the overall calculation time will be dominated by the slower component

Overall System   =   Single CPU   *   _____1_____
Performance   Performance   1-F + F/N

where   **F** = fraction of work performed in parallel
           N = number of parallel processors

Speedup = 1 / (1-F + F/N)

For N=10

F	S	F	S
20%	1.2	90%	5.3
40%	1.6	95%	6.9
60%	2.2	99%	9.2
80%	3.6	99.5%	9.6

For N=infinity

F	S	F	S
20%	1.3	90%	10
40%	1.7	95%	20
60%	2.5	99%	100
80%	5	99.5%	200

Amdahl's Law

My favorite example …..

Which system is faster?

System A: (16 processors)•(1 GFLOP each)   = **16 GFLOP total**

System B: (10,000 procs)•(100 MFLOP each)   = **1,000 GFLOP total**

Apply Amdahl's law, solve for F:

1 / (1-F + F/16) = .1 / (1-F + F/10000)

➔ **System A is faster, unless >99.3% of work is parallel**

- In general, a smaller number of fatter nodes is better
- For effective parallel speedups, must parallelize everything

Parallel
Monte Carlo

Parallel Algorithms

- **Possible parallel schemes:**

    - **Jobs**   run many sequential MC calculations, combine results

    - **Functional**   sources, tallies, geometry, collisions, …..

    - **Phase space**   space, angle, energy

    - **Histories**   Divide total number of histories among processors

- **All successful parallel Monte Carlo algorithms to date have been history-based.**

    - Parallel jobs always works, variation on parallel histories
    - Some limited success with spatial domain decomposition

Master / Slave Algorithm (Simple)

- **Master task**: control + combine tallies from each slave
- **Slave tasks**: Run histories, tallies in private memory

  - **Initialize**:
    - Master sends problem description to each slave
    - (geometry, tally specs, material definitions, …)
  - **Compute**, on each of N slaves:
    - Each slave task runs 1/N of total histories.
    - Tallies in private memory.
    - Send tally results back to Master.
  - **Combine tallies:**
    - Master receives tallies from each slave &
      - combines them into overall results.
- Concerns:
  - Random number usage
  - Load-balancing
  - Fault tolerance (rendezvous for checkpoint)
  - Scaling

10 -17

Master / Slave Algorithm (Simple)

Control + Bookkeeping   Computation

Master

```
! initialize
do n=1,nslaves
   send_info( n )

! Compute
nchunk = nhistories / nslaves
do n=1,nslaves
   k1 = 1 + (n-1)*nchunk
   k2 = min( k1+nchunk, nhistories)
   send_control( n, k1,k2 )

! Collect & combine results
totals(:) = 0
do n=1,nslaves
   recv_tallies( n )
   add_tallies_to_totals()

! Done
print_results()
save_files()
```

Slave 3

Slave 2

Slave 1

```
! Initialize
recv_info()                    )

                              2 )
! Compute
recv_control( k1, k2 )
do k=k1,k2                          ster
   run_history( k )
                                    ster
! Send tallies to master
send_tallies()

! Done
stop
```

10 -18

Random Number Usage

- **Linear Congruential RN Generator**

$$S_{k+1} = g\, S_k + C \mod 2^M$$

- **RN Sequence & Particle Histories**

••••••••••••••• ••••••••••••••• •••••••••••••

　　　　　1　　　　　　　2　　　　　　　3　　　　　　　etc.

MCNP stride for new history: 152,917

- **To skip ahead k steps in the RN sequence:**

$$S_k = g\, S_{k-1} + C \mod 2^M = g^k S_0 + C\,(g^k-1)/(g-1) \mod 2^M$$

- **Initial seed for n-th history**

$$S_0^{(n)} = g^{n*152917} S_0 + C\,(g^{n*152917}-1)/(g-1) \mod 2^M$$

This is easy to compute quickly using exact integer arithmetic

- **Each history has a unique number**
  - Initial problem seed $\rightarrow$ initial seed for n^{th} particle on m^{th} processor
  - If slave knows initial problem seed & unique history number, can initialize RN generator for that history

---

Fault Tolerance

- On parallel systems with complex system software & many CPUs, interconnects, disks, memory, MTBF for system is a major concern.

- Simplest approach to fault tolerance:
  - Dump checkpoint files every M histories (or XX minutes)
  - If system crashes, restart problem from last checkpoint

- Algorithm considerations
  - Rendezvous every M histories.
  - Slaves send current state to master, master saves checkpoint files

  - Parallel efficiency affected by M.

Fault Tolerance
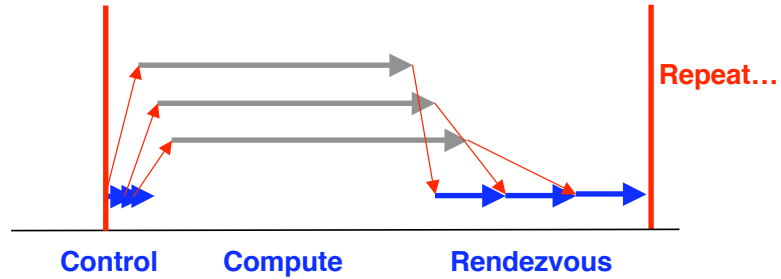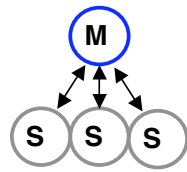
- For efficiency, want   (compute time) >> (rendezvous time)

    – Compute time: Proportional to #histories/task

    – Rendezvous time: Depends on amount of tally data &
                          latency+bandwidth for message-passing

---

Master / Slave Algorithm, with Rendezvous

- **Initialize**:

    Master sends problem description to each slave
    (geometry, tally specs, material definitions, …)

- For   rendezvous = 1, L

    - **Compute**, on each of N slaves:
        Each slave task runs 1/N of (total histories)/L.
        Tallies in private memory.
        Send tally results back to Master.

    - **Combine tallies:**
        Master receives tallies from each slave &
            combines them into overall results.

    - **Checkpoint:**
        Master saves current tallies & restart info in file(s)

Load Balancing

- **Time per history may vary significantly**
  - For problems using **variance reduction**:
    - Particles headed in "wrong" direction may be killed quickly, leading to a short history.
    - Particles headed in "right" direction may be split repeatedly. Since the split particles created are part of the same history, may give a very long history.

  - For problems run on a workstation **cluster**:
    - Workstation nodes in the cluster may have different CPU speeds
    - Workstations in the cluster may be simultaneously used for interactive work, with highly variable CPU usage on that node.
    - Node performance effectively varies continuously over time.

- **Naïve solution**
  - Monitor performance per node (e.g., histories/minute)
  - Periodically adjust number of histories assigned to each node, according to node performance

    # histories assigned to node n ∼ measured speed of node n

- **Better solution:**   **self-scheduling**

Load Balancing – Self-Scheduling

- For a problem with N slave processors, divide histories into **more than** N chunks.

  - Let   L = number of chunks,   L > N
  - Typically,   L ∼ 20 N   or   L ∼ 30 N

  - Histories/chunk = (total histories) / L

  - Slave: If idle, ask master for work. Repeat until no more work.
  - Master: Send chunk of work to idle slave. Repeat until no more work.

  - On average, imbalance in workload should be < 1/L

- Additional gains:
  - Naïve master/slave algorithm is **synchronous**
  - Self-scheduling master/slave algorithm is **asynchronous.** More overlap of communication & computation ➜ reduced wait times & better performance

Load Balancing – Self-Scheduling
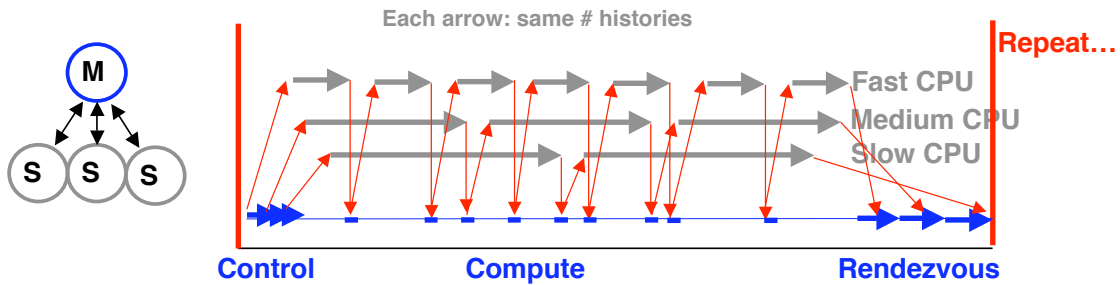
- Much more communication with Master, but only minimal amount of control info needed (1st & last history in chunk)

- Need to handle stopping condition carefully – avoid "dangling" messages

---

Load Balancing – Self-Scheduling

Master

```
! initialize
do n=1,nslaves
   send_info( n )

! Compute
nchunks = nslaves*20
nh = nhistories / nchunks
do n=1,nchunks
   k1 = 1 + (n-1)*nh
   k2 = min( k1+nh, nhistories)
   recv_idle_proc( M )
   send_control( M, k1,k2 )
enddo

! Collect & combine results
totals(:) = 0
do n=1,nslaves
   recv_idle_proc( M )
   send_control( M, 0, -1 )
   recv_tallies( M )
   add_tallies_to_totals()
enddo

! Done
print_results()
save_files()
```

Slave 3

Slave 2

Slave 1

```
! Initialize
recv_info()

! Compute
do
   send_idle_proc()
   recv_control( k1, k2 )
   if( k1>k2 )  exit
   do k=k1,k2
      run_history( k )
   enddo
enddo

! Send tallies to master
send_tallies()

! Done
stop
```

Hierarchical Parallelism

- For clustered SMPs,
  - Use message-passing to distribute work among slaves ("boxes")
  - Use threading to distribute histories among individual processors on box
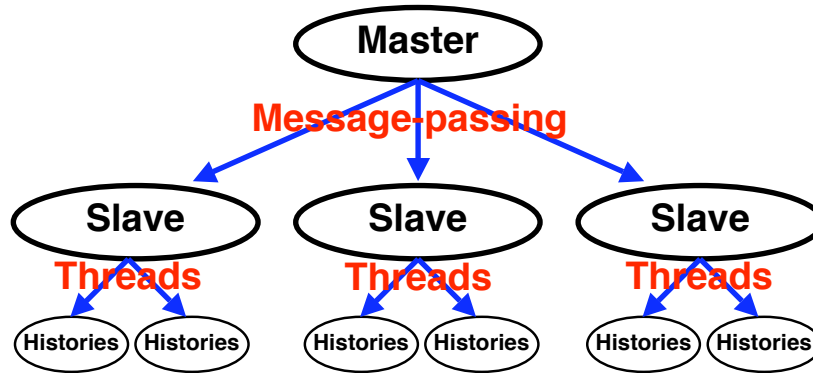
- Only the master thread (thread 0) on each slave uses MPI send/recv's

---

Master / Slave Algorithm, threaded & self-scheduling

Master

```
! initialize
do n=1,nslaves
   send_info( n )

! Compute
nchunks = nslaves*20
nh = nhistories / nchunks
do n=1,nchunks
   k1 = 1 + (n-1)*nh
   k2 = min( k1+nh, nhistories)
   recv_idle_proc( M )
   send_control( M, k1,k2 )
enddo

! Collect & combine results
totals(:) = 0
do n=1,nslaves
   recv_idle_proc( M )
   send_control( M, 0, -1 )
   recv_tallies( M )
   add_tallies_to_totals()
enddo

! Done
print_results()
save_files()
```

Slave 3

Slave 2

Slave 1

```
! Initialize
recv_info()                    )

! Compute
do
   send_idle_proc()                    ster
   recv_control( k1, k2 )
   if( k1>k2 )  exit
   !$OMP PARALLEL DO
   do k=k1,k2
      run_history( k )
   enddo
enddo

!$OMP PARALLEL
!$  combine_thread_tallies()

! Send tallies to master
send_tallies()

! Done
stop
```

Parallel Monte Carlo Performance

Parallel MC Computational Characteristics

- For master/slave algorithms (with self-scheduling, fault tolerance, & threads):

    – **No** communication among slave tasks

    – Occasional communication between master & slaves (rendezvous)

    – Slave tasks are compute-intensive
        - Few DO-loops
        - 40% of ops are test+branch (IF… GOTO…)
        - Irregular memory access, no repetitive patterns

    – For fixed-source problems:
        - Only 1 rendezvous is strictly necessary, at end of calculation
        - More rendezvous used in practice, for fault tolerance

    – For eigenvalue problems (K-effective):
        - Must have a rendezvous every cycle   (cycle = batch = generation)
        - Master controls iteration & source sampling

- Common-sense approach to performance:
        Fewer rendezvous ➔ better parallel performance

Parallel MC Performance Measures

- Metrics
  - Speedup $\quad\quad\quad S_N = T_1 / T_N$ $\quad\quad\quad\quad\quad$ N = # processors
  - Efficiency $\quad\quad\quad E_N = S_N / N$

- Fixed overall work $\quad\quad\quad\quad\quad\quad$ (fixed problem size)
  - Efficiency decreases with N
  - Speedup (eventually) drops as N increases
  - Why?

    As N increases, same communication/processor, but less work/processor (fewer histories/processor) ➔ (computation/communication) decreases

- Fixed work per processor $\quad\quad\quad\quad\quad$ (scaled problem size)
  - Efficiency approx. constant with N
  - Speedup approx. linear with N
  - Why?

    As N increases, same communication/processor, same work/processor
    (# histories ~ N) ➔ (computation/communication) stays approx. same
  - Called **scaled speedup**

Parallel MC Performance Limits

- Another way to determine efficiency

  Parallel Efficiency $\quad = \quad\quad\quad T_C / (T_C + T_M)$

  $\quad\quad\quad T_C$ = computing time
  $\quad\quad\quad T_M$ = time for messages, not overlapped with computing

- Slaves can send messages in parallel
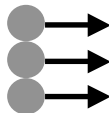
- Master receives & processes messages serially

  If enough messages are sent to master, extra wait time will limit performance

Parallel MC Performance Scaling

T_S T_C T_R

Repeat…

Control   **Compute**   **Rendezvous**
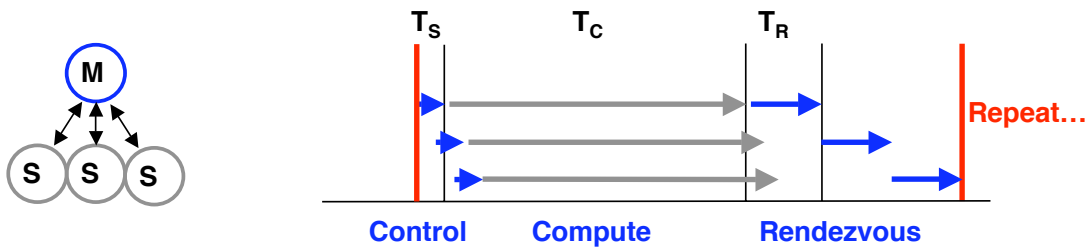
N = # processors

T_1 = CPU time for M histories using 1 processor

        (Depends on physics, geometry, compiler, CPU speed, memory, etc.)

L   = amount of data sent from 1 slave each rendezvous

$T_S = 0$ negligible, time to distribute control info

$T_R = s + L/r$ **s** = latency for message, **r** = streaming rate

T_C^{fix} $= T_1 / N$ fixed problem size, **M** histories/rendezvous
$T_C^{scale} = T_1$ scaled problem size, **NM** histories/rendezvous

---

Parallel MC Performance Scaling

• **Scaling models, for master/slave with serial rendezvous**

– "fixed"   = constant number of histories/rendezvous, M (constant work)
– "scaled"   = M histories/slave per rendezvous, NM total   (constant time)

Histories/rendezvous   **Speedup**

fixed $S = N / (1 + cN^2)$

scaled $S = N / (1 + cN)$

N = number of slaves
c = (s + L/r) / T_1

$T_1 \sim M$,   more histories/rendezvous ➜ larger T_1 , smaller c
S+L/r,   fixed, determined by number of tallies, ….

As M➔infinity, c➔0,   S➔N   (limit for 1 rendezvous)

Fixed size, serial messages

$$S = N / (1 + cN^2)$$

$$\frac{1}{2\sqrt{c}}$$

$$\frac{1}{\sqrt{c}}$$

Scaled size, serial messages

$$S = N / (1 + cN)$$

$1/c$

$1/2c$

$1/c$

N = number of slaves
$$c = (s + L/r) / (M_1 t_h)$$

10 - 35

VIM Monte Carlo — Example of Estimating Performance

- TREAT reactor, continuous-energy neutron transport
- Per slave: M_1 = 400 histories,
  L = .341 MB of tally data
- Communications:

Workstation network,	P4 + ethernet:	$s \approx .001$ sec, $r \approx .8$ MB/sec
IBM-SP1,	P4 + ethernet:	$s \approx .001$ sec, $r \approx .8$ MB/sec
IBM-SP1,	P4 + EUI-H:	$s \approx 50$ μsec, $r \approx 8.5$ MB/sec

	t_h	$T_1 = 400\, t_h$	T_r	c
Sun Sparc2	.25 sec	100 sec	.43 sec	.004
rs6000/350	.10 sec	40 sec	.43 sec	.011
SP1 - ethernet	.075 sec	30 sec	.43 sec	.014
SP1 - EUIH	.075 sec	30 sec	.04 sec	.001

Note: T_1 — very repeatable & predictable.
  T_r — difficult to measure on busy machine

10 - 36

Parallel MC Performance Scaling

VIM Monte Carlo — Measured Performance on SP1 — TREAT reactor

Scaled Speedup (y-axis), **Number of Slaves** (x-axis)
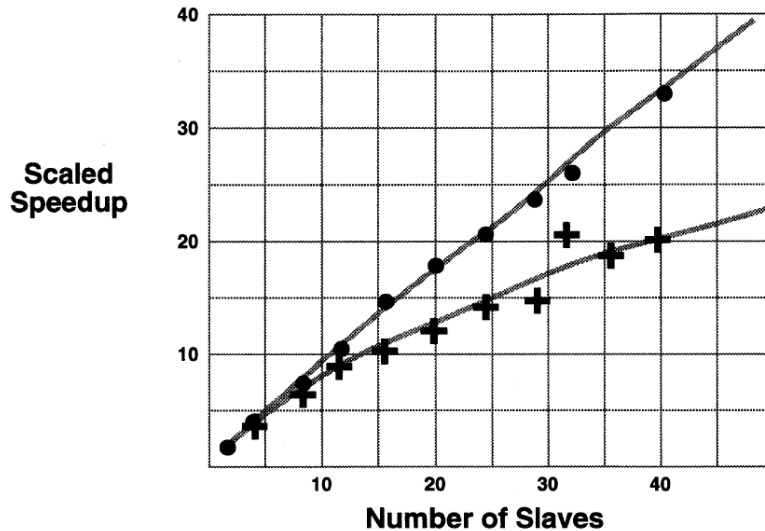
● SP1 — P4 + EUIH $\quad$ $S = N / (1 + .0056 N)$

✚ SP1 — P4 + ethernet $\quad$ $S = N / (1 + .028 N)$

**Measured message passing on SP1 is 2-3 times slower than specs
(busy machine; experimental software; flaky hardware)**

Parallel MC Performance Scaling

Parallel Eigenvalue Calculations — scaled size, serial messages

$$S \quad = \quad N / (1 + cN)$$

$$S_{max} = \quad 1 / c$$

$$N_{1/2} = \quad 1 / c$$

> N = number of slaves
>
> $c = \quad (s + L/r) / (M_1 t_h)$

Examples:

- **VIM, TREAT problem**

Sun Sparc2 workstation cluster	c=.0043	S_{max} = 233
rs6000/350 workstation cluster	c=.011	S_{max} = 93
SP1, using ethernet	c=.014	S_{max} = 70
SP1, using EUIH comm.	c=.00134	S_{max} = 748

- **RACER, "typical" large problem**

    100 K histories/min, 20 K histories/slave
    32 MB tally data, r ~ 1800 MB/sec

Cray-C90, using SSD for messages (16 processors, max)	c~.001	S_{max} ~ 1000

- **Master/slave algorithms work well**
  - Load-balancing: Self-scheduling
  - Fault-tolerance: Periodic rendezvous
  - Random numbers: Easy, with LCG & fast skip-ahead algorithm
  - Tallies: Use OpenMP "critical sections"
  - Scaling: Simple model, more histories/slave + fewer rendezvous
  - Hierarchical: Master/slave MPI, OpenMP threaded slaves
  - Portability: MPI/OpenMP, clusters of anything

- **Remaining difficulties**
  - Memory size: Entire problem must fit on each slave

    - Domain-decomposition has had limited success
      - Should be OK for reactor problems
      - May not scale well for shielding or time-dependent problems
      - For general 3D geometry, effective domain-decomposition is unsolved problem

    - Random access to memory distributed across nodes gives huge slowdown
      - May need functional parallelism with "data servers"

MCNP5
Parallel
Calculations

DOE Advanced Simulation & Computing – ASC

Blue Pacific – 3 TeraOps

Red – 3 TeraOps

Blue Mountain – 3 TeraOps
(R.I.P.)

Lightning
Red Storm
Blue Gene/L

White – 12 TeraOps

Q – 20 TeraOps

Hierarchical Parallelism

- Use **message-passing** to distribute work among slaves ("boxes")
- Use **threading** to distribute histories among individual cpus on box

Master

Message-passing   **MPI**

Slave   **Slave**   **Slave**

Threads   **Threads**   **Threads**   **OpenMP**

Histories Histories   Histories Histories   Histories Histories
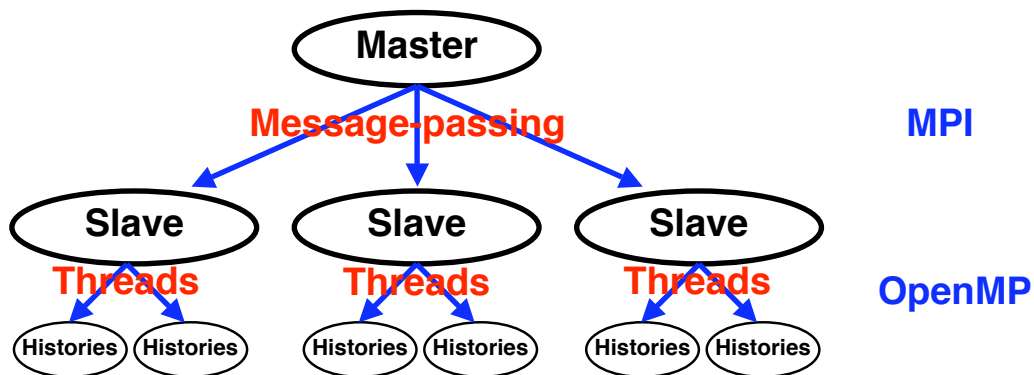
- **We routinely test MCNP5 on:**
    – **ASCI Bluemountain** –   SGI,   48 boxes x 128 cpus/box
    – **ASCI White** –   IBM,   512 boxes x   16 cpus/box
    – **ASCI Q** –   HP, 2 x 512 boxes x   4 cpus/box
    – **Linux clusters**
    – **Windows PC cluster**
- **1,000 processor jobs are "routine"**

Parallelism in MCNP5

- Threading
  - Individual histories are handled by separate threads
  - No thread synchronization is needed during a history
  - Implemented by OpenMP compiler directives

  - Tallies, RN data, & some temporary variables for history are in thread-private memory

    Example:

    common **/RN_THREAD/** RN_SEED, RN_COUNT, RN_NPS
    !$OMP THREADPRIVATE (**/RN_THREAD/**)
    save /RN_THREAD/

  - OpenMP **critical sections** are used for some tallies or variable updates

    Example:

    !$OMP CRITICAL (RN_STATS)
    RN_COUNT_TOTAL = RN_COUNT_TOTAL + RN_COUNT
    $!OMP END CRITICAL (RN_STATS)

  - Message-passing & file I/O are executed only from thread-0 (master thread) for each MPI task

---

Parallelism in MCNP5

- Message-passing
  - In MCNP5, all message-passing is handled by calls to the **dotcomm** package, a communications layer which contains an interface to either MPI or PVM
  - Recommend using MPI – PVM is obsolete & won't be supported in future

MCNP5	
dotcomm	
MPI	**PVM**
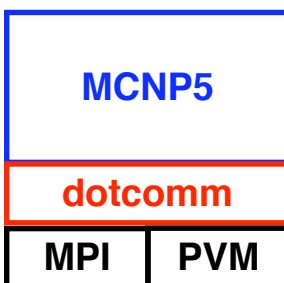
  - Either MPI or PVM message-passing is selected in **dotcomm** <u>at compile-time</u>

  - Using the **dotcomm** package & either MPI or PVM, MCNP5 can run in parallel without source code changes on

    - Parallel supercomputers (e.g., ASCI tera-scale computers)
    - COWs (clusters of workstations)
    - Linux clusters
    - PC clusters

MCNP5 Parallel Calculations

N = total number of MPI tasks,   master + (N-1) slaves

M = number of OpenMP threads/slave

- Running on parallel systems with MPI only

    mpirun -np **N** mcnp5.mpi   i=inp01 …..

- Running with threads only

    mcnp5   tasks **M** i=inp01 …..

- Running on parallel systems with MPI & threads

    ASCI Bluemountain (SGI)

    mpirun -np **N** mcnp5.mpi tasks **M** i=inp01 …..

    ASCI Q (HP/Compaq)

    prun –n **N** -c **M**   mcnp5.mpi tasks **M**   i=…

        If submitting jobs through a batch system (e.g., LSF),
        N & M must be consistent with LSF requested resources

MCNP5 Parallel Calculations

- How many threads ?
    - Max number of threads = # CPUs per node

        - ASCI Bluemountain: 128 cpus / node
        - ASCI Q: 4 cpus /node
        - Laptop PC cluster: 1 cpu / node

    - Experience on many systems has shown that a moderate number of threads per slave is efficient; using too many degrades performance

        - ASCI Bluemountain: 4–12 threads/slave usually effective
                                        >16 threads/slave usually has bad performance
        - ASCI Q: 4 threads/slave is effective

    - Rules-of-thumb vary for each system
        - Thread efficiency is strongly affected by operating system design
        - Scheduling algorithm for threads used by operating system is generally designed to be efficient for small number of threads (<16)
        - For large number of threads, context-switching & cache management may take excessive time, giving poor performance
        - Other jobs on system (& their priority) affect thread performance
        - No definite rules – need to experiment with different numbers of threads

MCNP5 Parallel Calculations

- Parallel performance is sensitive to number of rendezvous
    - Can't control number of rendezvous directly
    - The following things cause a rendezvous:
        - Printing tallies
        - Dumping to the RUNTPE file
        - Tally Fluctuation Chart (TFC) entries
        - Each cycle of eigenvalue problem

- Use PRDMP card to minimize print/dump/TFC

    PRDMP   ndp   ndm   mct   ndmp   dmmp

    ndp   = increment for printing tallies   ⬅ **use large number**
    ndm   = increment for dump to RUNTPE   ⬅ **use large number**
    mct   = flag to suppress time/date info in MCTAL
    ndmp = max number of dumps in RUNTPE
    dmmp = increment for TFC & rendezvous   ⬅ **use large number**

    For fixed-source problems,   increments are in particles
    For eigenvalue problems,   increments are in cycles

MCNP5 Parallel Calculations

- Keff calculations: Use KCODE card for hist/cycle

    - Want to reduce the number of cycles
    - More histories in each cycle
    - Should run hundreds of cycles or more for good results

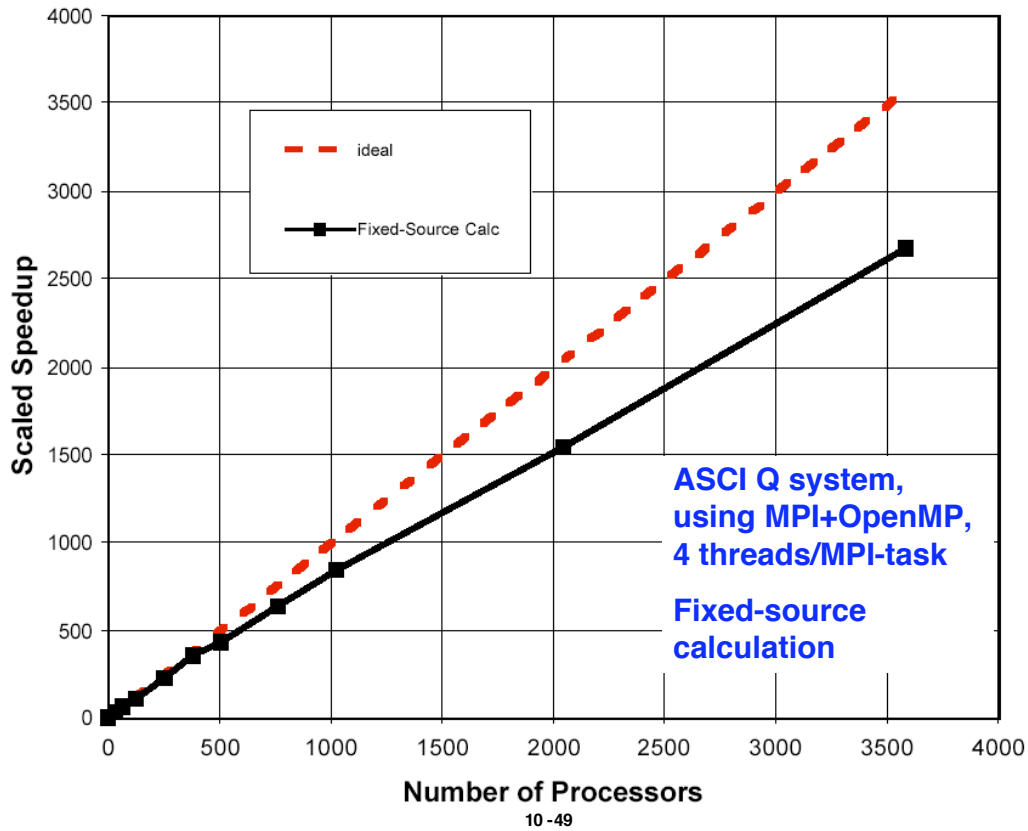   **KCODE   nsrck   rkk   ikz   kct   …..**

    nsrck = histories / cycle   ⬅ **use a large number**
    rkk   = initial guess for Keff
    ikz   = number of initial cycles to discard
    kct   = total number of cycles to run

    Suggested: nsrck ~   (thousands) x (number of processors)

MCNP5 Parallel Scaled Speedup

ASCI Q system,
using MPI+OpenMP,
4 threads/MPI-task

Fixed-source
calculation

MCNP5 Parallel Calculations

MCNP Speed vs. Number of Processors
BNCT Model w/ NPS=100,000 on a Linux Cluster w/ MPICH

Scaled Parallel Speedup – Eigenvalue Problem
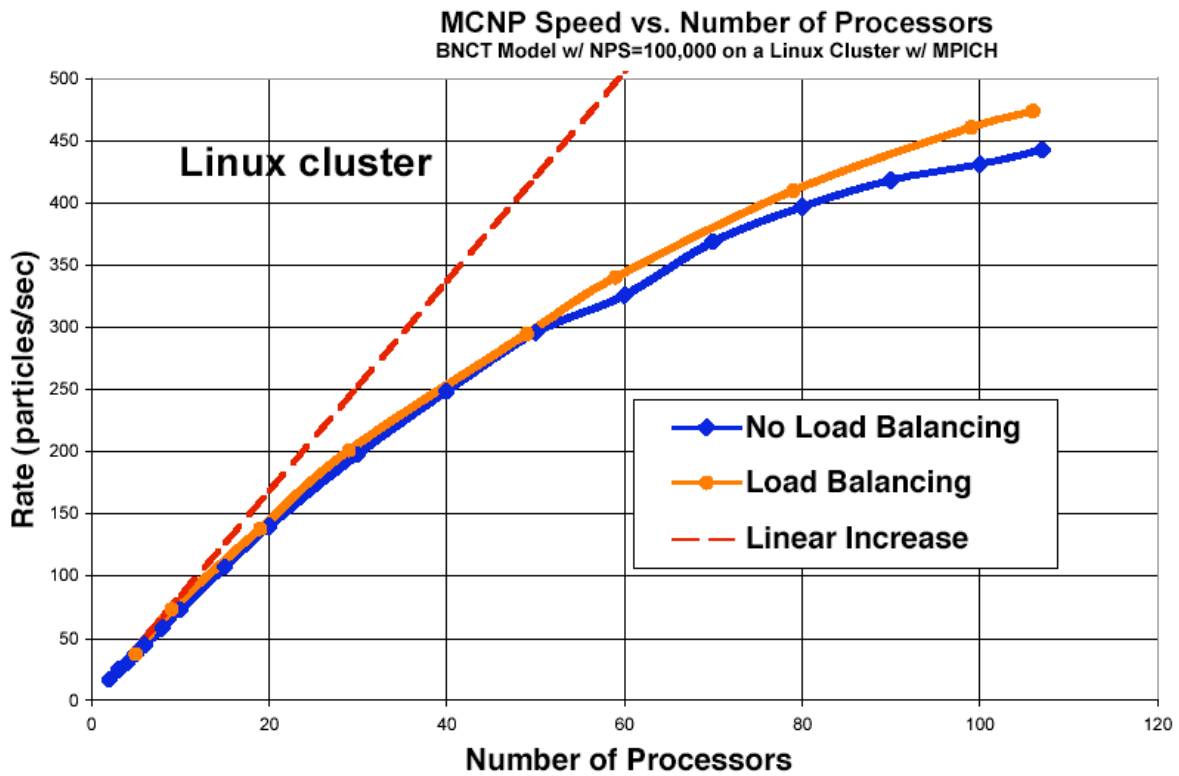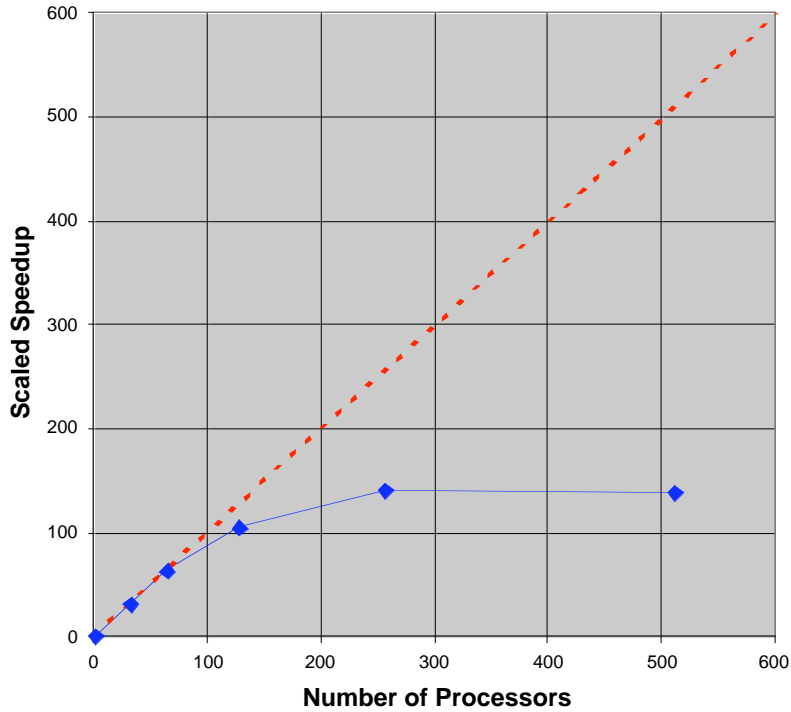
ASCI Qsc
- MCNP5
- MPI + OpenMP
- 4 threads/node
- 1000 cycles

Parallel Processing For Large Monte Carlo Calculations

Domain Decomposition

If a Monte Carlo problem is too large to fit into memory of a single processor

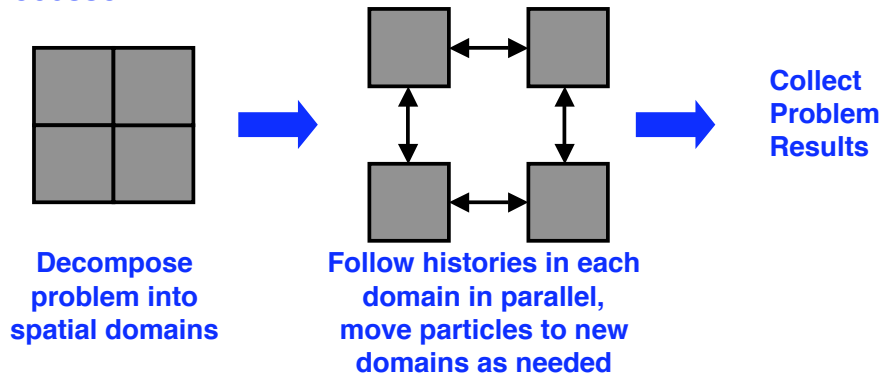**Decompose problem into spatial domains**

Follow histories in each domain in parallel, move particles to new domains as needed

Collect Problem Results

- Need periodic synchronization to interchange particles among nodes
- Use message-passing (MPI) to interchange particles

➡ **Domain decomposition is often used when the entire problem will not fit in the memory of a single SMP node**

Parallel Monte Carlo

- **Inherent parallelism is on <u>particles</u>**
  - Scales well for all problems

- **Domain decomposition**
  - Spatial domains on different processors
  - Scales OK for Keff or α calculations, where particle distribution among domains is roughly uniform
  - Does **not** scale for time-dependent problems due to severe load imbalances among domains

- **Domain decomposition – scaling with N processors**
  - **Best:** performance ~ N **(uniform distribution of particles)**
 - **Worst:** performance ~ 1 **(localized distribution of particles)**

Parallel Monte Carlo

Los Alamos
NATIONAL LABORATORY
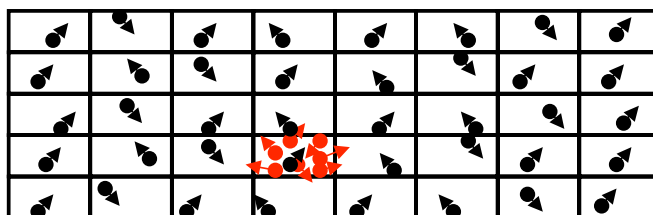
- **Data is distributed by domain decomposition,
  but parallelism is on particles**

- **Solution ?**

  Parallel on particles + distributed data

- **Particle parallelism + Data Decomposition**
  - **Existing parallel algorithm for particles**
  - **Distribute data among processor nodes**
  - **Fetch the data to the particles as needed (dynamic)**

  - **Essentially same approach as used many years ago for CDC (LCM) or CRAY (SSD) machines**

  - **Scales well for all problems (but slower)**

Parallel Monte Carlo

Los Alamos
NATIONAL LABORATORY

- **Particle parallelism + data decomposition — <u>logical</u> view:**
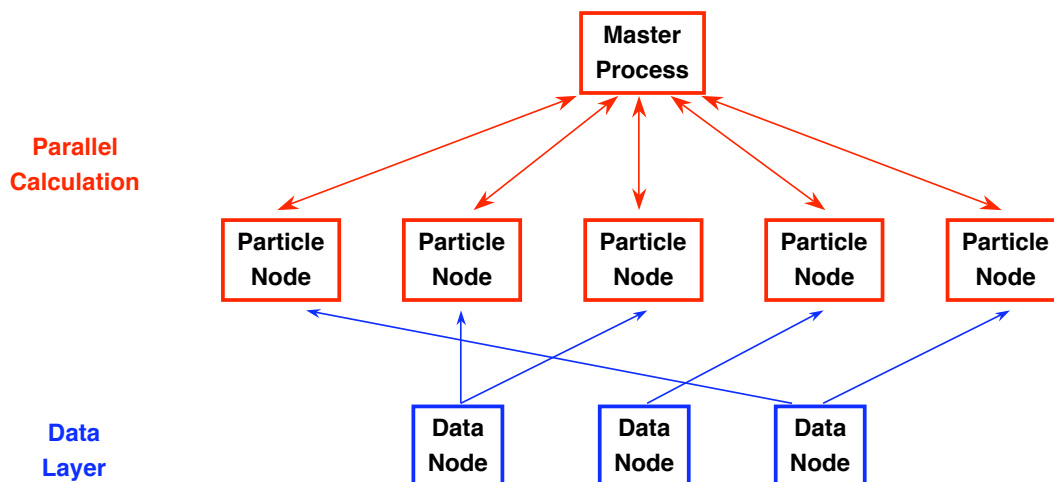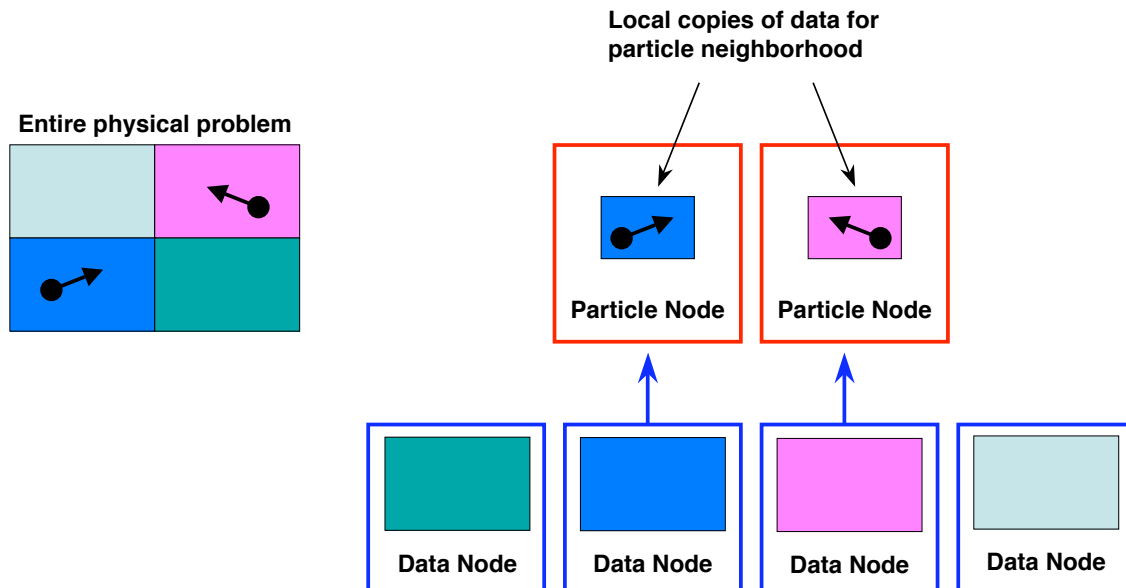

- **Mapping of logical processes onto compute nodes is flexible:**
  - Could map particle & data processes to **different** compute nodes
  - Could map particle & data processes to **same** compute nodes
- **Can replicate data nodes if contention arises**

Parallel Monte Carlo

- **Particle parallelism + data decomposition**

Local copies of data for
particle neighborhood

Entire physical problem

Particle Node   Particle Node

Data Node   Data Node   Data Node   Data Node

Parallel Monte Carlo

- **History modifications for data decomposition**

    source

    while   wgt > cutoff

    .   compute distances & keep minimum:
    .   dist-to-boundary
    .   dist-to-time-cutoff
    .   dist-to-collision
    .   **dist-to-data-domain-boundary**

    .   move particle
    .   pathlength tallies

    .   **if   distance == dist-to-data-domain-boundary**
    .   **fetch new data**

    .   collision physics
    .   roulette & split
    . . .

Parallel Monte Carlo

- **Data windows & algorithm tuning**
  - Defining the "particle neighborhood" is an art
  - Anticipating the flight path can guide the pre-fetching of blocks of data
  - Tuning parameters:
    - How much data to fetch ?
    - Data extent vs. particle direction ?

Entire physical problem

- **Examples**

Parallel Monte Carlo
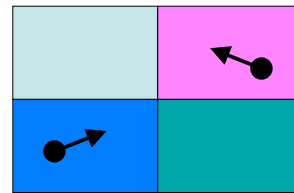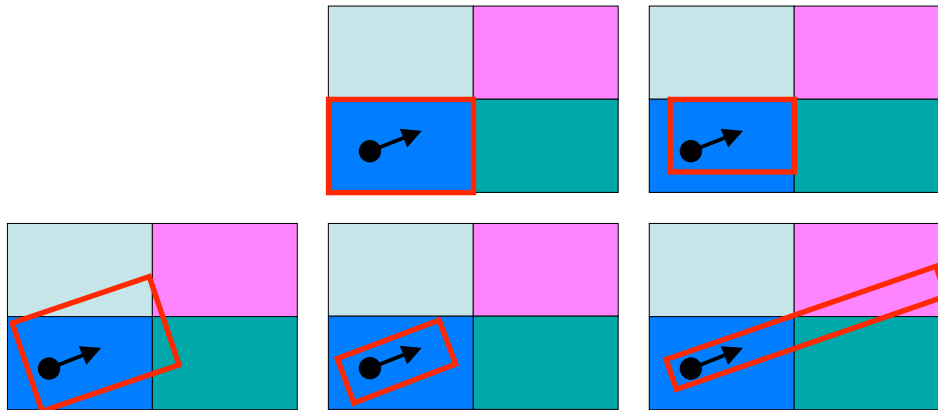
- **Point detector tallies are non-local**
  - Every collision contributes an expected score
  - At every collision, "pseudo-particles" are tracked along the path from collision to detector
  - Scores from all "pseudo-particles" (including from all split particles) must be tallied together into a single score for the history

Entire physical problem

Conclusions

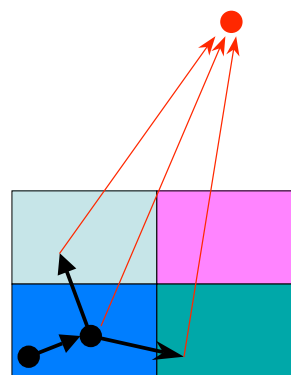**For Monte Carlo problems which <u>can fit in memory</u>:**

- **Concurrent scalar jobs – ideal for Linux clusters**

- **Master/slave parallel algorithm (replication) works well**
    - Load-balancing: Self-scheduling
    - Fault-tolerance: Periodic rendezvous
    - Random numbers: Easy, with LCG & fast skip-ahead algorithm
    - Tallies: Use OpenMP "critical sections"
    - Scaling: Simple model, more histories/slave + fewer rendezvous
    - Hierarchical: Master/slave MPI, OpenMP threaded slaves
    - Portability: MPI/OpenMP, clusters of anything

For Monte Carlo problems <u>too large to fit in memory</u>:

- **Spatial domain decomposition (with some replication) can work for some problems**

- **Particle parallelism + data decomposition is a promising approach which should scale for all problems**

References

1. S. MATSURA, F. B. BROWN, R. N. BLOMQUIST, "Parallel Monte Carlo Eigenvalue Calculations," *Trans. Am. Nucl. Soc.* (Nov. 1994).
2. F.B. BROWN, "Random Number Generation with Arbitrary Strides," *Trans. Am. Nucl. Soc.* (Nov., 1994)
3. F.B. BROWN & Y. NAGAYA, "The MCNP5 Random Number Generator", *Trans. Am. Nucl. Soc.* (Nov., 2002)
4. X-5 MONTE CARLO TEAM, "MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume I: Overview and Theory," LA-UR-03–1987, Los Alamos National Laboratory (April, 2003).
5. F. B. BROWN, J. E. SWEEZY, J. T. GOORLEY, "MCNP5 Parallel Processing Workshop," LA-UR-03-2228, Los Alamos National Laboratory (2003).
6. A. MAJUMDAR and W. R. MARTIN, "Performance Measurement of Monte Carlo Photon Transport on Parallel Machines," *PHYSOR 2000: ANS Int. Topical Meeting*, Pittsburgh (May 2000).
7. S. R. LEE, S. D. NOLEN, F. B. BROWN, "MC++ Monte Carlo Code for ASCI," Los Alamos National Laboratory report (1998).
8. R. PROCASSINI, et al., "Design, Implementation and Testing of MERCURY, a Parallel Monte Carlo Transport Code," *Proc. ANS Math. & Comp. Topical*, Gatlinburg, TN, April 6–11 (2003)
9. H.J. ALME, G.H. RODRIGUE, G.B. ZIMMERMAN, "Domain Decomposition Models for Parallel Monte Carlo Transport", J. Supercomputing, 18, 5–23 (2001).
10. "MPI: A Message Passing Interface", http://www-unix.mcs.anl.gov/mpi/index.html
11. "PVM: Parallel Virtual Machine", http://www.epm.ornl.gov/pvm
12. "OpenMP Fortran Application Program Interface", http://www.openmp.org

References

I. Monte Carlo Methods for Particle Transport Problems

A. First Known Reference for Using Monte Carlo Methods to Solve Particle Transport Problems

- J. von Neumann, letter to R.D. Richtmyer, in "Statistical Methods in Neutron Diffusion," LANL report LAMS-551 (1947).

B. Highly Recommended

- L. L. Carter and E. D. Cashwell, Particle Transport Simulation with the Monte Carlo Method, ERDA Critical Review Series, TID-26607, National Technical Information Service, Springfield MA (1975).

C. Introductory References

- M. H. Kalos and P. A. Whitlock, Monte Carlo Methods. Volume I: Basics, John Wiley & Sons, NY (1986).

- E. E. Lewis and W. F. Miller, Jr., Computational Methods of Neutron Transport, American Nuclear Society, Inc., LaGrange Park, IL (1993).

- S. Nakamura, Computational Methods in Engineering and Science, R. E. Krieger Pub. Company, Malabar, FL (1986).

- S.A. Dupree & S.K. Fraley, A Monte Carlo Primer – A Practical Approach to Radiation Transport, Kluwer Academic, NY (2002).

- S.A. Dupree & S.K. Fraley, A Monte Carlo Primer – Volume 2, Kluwer Academic, NY (2004).

- J. Wood, Computational Methods in Reactor Shielding, Pergamon Press, Oxford (1982).

D. Advanced References

- Lux & L. Koblinger, Monte Carlo Particle Transport Methods: Neutron and Photon Calculations, CRC Press, Boston (1991).

- J. J. Duderstadt and W. R. Martin, Transport Theory, John Wiley & Sons, NY (1979).

- G. Goertzel and M. H. Kalos, "Monte Carlo Methods in Transport Problems," in *Progress in Nuclear Energy*, Series I, Physics and Mathematics, Vol. 2, (1958).

- E. D. Cashwell and C.J. Everett, A Practical Manual on the Monte Carlo Method for Random Walk Problems, Pergamon Press, London (1959).

- J.M. Hammersley & D.C. Handscomb, Monte Carlo Methods, John Wiley & Sons, NY (1964).

- Y. A. Schreider, The Monte Carlo Method, Pergamon Press, NY (1966).

- J. Spanier and E. M. Gelbard, <u>Monte Carlo Principles and Neutron Transport Problems</u>, Addison-Wesley, Reading, MA (1969).

- X-5 Monte Carlo Team, "MCNP – A General N-Particle Transport Code, Version 5 – Volume I: Overview and Theory", LA-UR-03-1987 (April, 2003).

- X-5 Monte Carlo Team, "MCNP – A General N-Particle Transport Code, Version 5 – Volume II: User's Guide", LA-CP-03-0245 (April, 2003).

- X-5 Monte Carlo Team, "MCNP – A General N-Particle Transport Code, Version 5 – Volume III: Developer's Guide", LA-CP-03-0284 (April, 2003).

- F.B. Brown, "Fundamentals of Monte Carlo Particle Transport," LA-UR-04-8817 (Dec, 2004).

II. General References on the Monte Carlo Method

- G.S. Fishman, <u>Monte Carlo Concepts, Algorithms, and Applications</u>, Springer-Verlag, NY (1996).

- R. Y. Rubinstein, <u>Simulation and the Monte Carlo Method</u>, John Wiley & Sons, NY (1981).

- J. H. Halton, "A Retrospective and Prospective Survey of the Monte Carlo Method," *SIAM Rev* 12,1, (1970).

- C.P. Robert & G. Casella, <u>Monte Carlo Statistical Methods</u>, Springer (2004).

III. Random Number Generation and Random Sampling Methods

- L. Devroye, <u>Non-Uniform Random Variate Generation</u>, Springer-Verlag, NY (1986).

- D. E. Knuth, <u>The Art of Computer Programming, Vol. 2: Semi-numerical Algorithms</u>, 3rd Edition, Addison-Wesley, Reading, MA (1998).

- J. von Neumann, "Various Techniques Used in Conjunction with Random Digits," *J. Res. Nat. Bur. Stand. Appl. Math Series* **3**, 36-38 (1951).

- C. J. Everett and E. D. Cashwell, "A Third Monte Carlo Sampler," LA9721-MS, Los Alamos National Laboratory, Los Alamos, NM (1983).

- H. Kahn, "Applications of Monte Carlo," AECU-3259, Rand Corporation, Santa Monica, CA (1954).

- E. J. McGrath and D. C. Irving, "Techniques for Efficient Monte Carlo Simulation," ORNL-RSIC-38, Vols. I-III, Oak Ridge National Laboratory, Oak Ridge, TN (1975).

- J.E. Gentle, <u>Random Number Generation and Monte Carlo Methods</u>, Springer, NY (2003).

- F.B. Brown & Y. Nagaya, "The MCNP5 Random Number Generator", *Trans. Am. Nucl. Soc.* [also, LA-UR-02-3782] (November, 2002).

- F. B. Brown, "Random Number Generation with Arbitrary Strides," *Trans. Am. Nucl. Soc*. **71**,202 (1994).

- F.B. Brown & W.R. Martin, "Direct Sampling of Monte Carlo Flight Paths in Media with Continuously Varying Cross-sections", *ANS Topical Meeting on Mathematics & Computation*, Gatlinburg, TN April 6-11, 2003 [also, LA-UR-02-6530] (September, 2002).

- F. B. Brown & W. R. Martin, "Monte Carlo Particle Transport in Media with Exponentially Varying Time-dependent Cross-sections", *Trans. Am. Nucl. Soc* [also, LA-UR-01-675] (June 2001).

- F. B. Brown and J. L. Vujic, "Comparison of Direct and Rejection Sampling Methods," *Trans.Am. Nucl. Soc.* **69**, 223 (Nov. 1993).

IV. Monte Carlo Methods for the Solution of Reactor Eigenvalue Problems

- J. Lieberoth, "A Monte Carlo Technique to Solve the Static Eigenvalue Problem of the Boltzmann Transport Equation," *Nukleonik* **11**,213 (1968).

- M. R. Mendelson, "Monte Carlo Criticality Calculations for Thermal Reactors," *Nucl. Sci Eng.* **32**, 319-331 (1968).

- H. Rief and H. Kschwendt, "Reactor Analysis by Monte Carlo," *Nucl. Sci. Eng.*, **30**, 395 (1967).

- R. C. Gast and N. R. Candelore, "Monte Carlo Eigenfunction Strategies and Uncertainties," in Proc. NEACRP Meeting of a Monte Carlo Study Group, ANL-75-2, Argonne National Laboratory, Argonne, IL (1974).

- M. H. Kalos, F. R. Nakache, and J. Celnik, "Monte Carlo Methods in Reactor Computations," in Computing Methods in Reactor Physics, Gordon and Breach, NY (1968).

- D. C. living, R. M. Freestone, and F. B. K. Kam, "O5R, A General Purpose Neutron Monte Carlo Code," ORNL-3622, Oak Ridge National Laboratory (1965).

- E. R. Woodcock, et al, "Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry," in Proc. Conf Applications of Computing Methods to Reactor Problems, ANL-7050, Argonne National Laboratory, Argonne, IL (1965).

- W. Goad and R. Johnston, "A Monte Carlo Method for Criticality Problems," *Nucl. Sci. Eng.* **5**, 371-375 (1959).

- R.J. Brissenden and A.R. Garlick, "Biases in the Estimation of Keff and Its Error by Monte Carlo Methods," *Ann. Nucl. Energy*, Vol 13, No. 2, 63-83 (1986).

- T. Yamamoto and Y. Miyoshi, "Reliable Method for Fission Source Convergence of Monte Carlo Criticality Calculation with Wielandt's Method," *J. Nucl. Sci. Technol.* **41**, No. 2, 99-107 (2004).

- F.B. Brown, W.R. Martin, W. Ji, J.L. Conlin, & J.C. Lee, "Stochastic Geometry and HTGR Modeling for MCNP5", submitted to *ANS Monte Carlo 2005 Topical Meeting,* Chattanooga TN, April 17-21, 2005, [also LA-UR-04-8668] (Dec, 2004).

- F.B. Brown & W.R. Martin, "Stochastic Geometry Capability in MCNP5 for the Analysis of Particle Fuel", *Annals of Nuclear Energy*, Vol 31, Issue 17, pp 2039-2047 [also LA-UR-04-5362] (Nov, 2004).

- F.B. Brown & R.D. Mosteller, "MCNP5 Workshop – PHYSOR-2004", LA-UR-04-2647 (April, 2004).

- T. Ueki & F.B. Brown, "Stationarity Modeling and Informatics-Based Diagnostics in Monte Carlo Criticality Calculations", *Nucl. Sci. Eng.* **149**, No. 1, 38-50 [also LA-UR-03-8343] (Jan, 2005).

- T. Ueki, F.B. Brown, D.K. Parsons, & J.S. Warsa, "Time Series Analysis of Monte Carlo Fission Sources: I. Dominance Ratio Computation", *Nucl. Sci. Eng.* **148**, No. 3, 374-390 [also LA-UR-03-5823] (Nov, 2004).

- T. Ueki & F.B. Brown, "Informatics Approach to Stationarity Diagnostics of the Monte Carlo Fission Source Distribution," *Trans. Am. Nucl. Soc.* [also LA-UR-03-3949] (Nov, 2003).

- T. Ueki, F.B. Brown, & D.K. Parsons, "Dominance Ratio Computation via Time Series Analysis of Monte Carlo Fission Sources", *Trans. Am. Nucl. Soc.* [also LA-UR-03-0106] (June, 2003).

- Y. Nagaya & F.B. Brown, "Implementation of a Method to Estimate Change in Eigenvalue Due to Perturbed Fission Source Distribution Into MCNP", LA-UR-03-1387 (Feb. 2003).

- Y. Nagaya & F.B. Brown, "Estimation of Change in Keff Due to Perturbed Fission Source Distribution in MCNP", *ANS Topical Meeting on Mathematics & Computation*, Gatlinburg, TN April 6-11, 2003 [also, LA-UR-02-6879] (September, 2002).

- T. Ueki & F.B. Brown, "Stationarity and Source Convergence in Monte Carlo Criticality Calculations", *ANS Topical Meeting on Mathematics & Computation*, Gatlinburg, TN April 6-11, 2003 [also, LA-UR-02-6228] (September, 2002).

- T. Ueki, F.B. Brown, D.K. Parsons, & D.E. Kornreich, "Autocorrelation and Dominance Ratio in Monte Carlo Criticality Calculations", *Nucl. Sci. Eng.* [also, LA-UR-02-5700] (September, 2002).

- T. Ueki & F.B. Brown, "Stationarity Diagnostics Using Shannon Entropy in Monte Carlo Criticality Calculations I: F Test", *Trans. Am. Nucl. Soc.* [also, LA-UR-02-3783] (November, 2002).

- F.B. Brown, R.C. Little, A. Sood, & D.K. Parsons, "MCNP Calculations for the OECD/NEA Source Convergence Benchmarks", *Trans. Am. Nucl. Soc.* [also, LA-UR-02-3987] (November, 2002).

- T. Ueki & F.B. Brown, "Autoregressive Fitting for Monte Carlo K-effective Confidence Intervals", *Trans. Am. Nucl. Soc.* [also, LA-UR-01-6770] (June, 2002).

- F.B. Brown, R.C. Little, A. Sood, D.K. Parsons, T.A. Wareing, "MCNP Calculations for the OECD/NEA Source Convergence Benchmarks for Criticality Safety Analysis", LA-UR-01-5181 (Sept. 2001)

- D. J. Kelly, "Depletion of a BWR Lattice Using the RACER Continuous-Energy Monte Carlo Code," *Proceedings of the ANS Intl. Conf on Mathematics & Computation, Reactor Physics, & Environ. Analyses*, April 30-May 4, Portland, Oregon (1995).

- T. M. Sutton and F. B. Brown, "Parallel Monte Carlo for Reactor Calculations," *Proceedings of the ANS Topical Meeting on Advances in Reactor Physics,* April 11-15, 1994, Knoxville, TN (April 1994).

- F. B. Brown, K. L. Derstine, and R. N. Blomquist, "Distributed Computing and Nuclear Reactor Analysis," *Proceedings of the ANS Topical Meeting on Advances in Reactor Physics*, April 11-15, 1994, Knoxville, TN (April 1994).

- R. N. Blomquist and F.B. Brown, "Parallel Monte Carlo Reactor Neutronics," *Proceedings of the Society for Computer Simulation Meeting on High Performance Computing '94*, April 11-15, 1994, La Jolla, CA (April 1994).

- E.M. Gelbard, F. B. Brown, and A. G. Gu, "Estimation of Fission Source Bias in Monte Carlo Eigenvalue Calculations," *Trans. Am. Nucl. Soc.* **69**,201 (Nov. 1993).

- F. B. Brown and T. M. Sutton, "Reproducibility and Monte Carlo Eigenvalue Calculations," *Trans. Am. Nucl. Soc.* **65**,235 (June 1992).

- F. B. Brown and F. G. Bischoff, "Computational Geometry for Reactor Applications," *Trans.Am. Nucl. Soc.* **57**, 112 (1988).

- F. B. Brown and M. R. Mendelson, "Vectorized Monte Carlo Applications in Reactor Physics Analysis," *Trans. Am. Nucl. Soc.* **46**, 727 (June 1984).

- F. B. Brown, "Vectorized Monte Carlo Methods for Reactor Lattice Analysis," *Proceedings ANS Topical Meeting on Advances in Reactor Computations*, Salt Lake City, Utah, 108-123 (March 1983).

- F. B. Brown, "Vectorized Monte Carlo Methods for Reactor Lattice Analysis," KAPL-4163, Knolls Atomic Power Laboratory, Schenectady, NY (March 1983).

- F. B. Brown, "Development of Vectorized Monte Carlo Algorithms for Reactor Lattice Analysis," *Trans. Am. Nucl. Soc.* **43**, 377 (1982).

- J.E. Sweezy, et al., "Automated Variance Reduction for MCNP5 using Deterministic Methods", *Proc. Int. Conf. on Radiation Shielding (ICRS-10) and the ANS Radiation Protection and Shielding Division Topical Meeting*, Madeira, Portugal [also LA-UR-04-2956] (April, 2004).

- F.B. Brown, J.E. Sweezy, R. Hayes, "Monte Carlo Parameter Studies and Uncertainty Analysis with MCNP5", *PHYSOR-2004, ANS Reactor Physics Topical Meeting*, Chicago, April 25-29 [also LA-UR-04-0499] (April, 2004).

- F.B. Brown, D. Griesheimer, W.R. Martin, "Continuously Varying Material Properties and Tallies for Monte Carlo Calculations", *PHYSOR-2004, ANS Reactor Physics Topical Meeting,* Chicago, April 25-29 [also LA-UR-04-0732] (April, 2004).

- T. M. Sutton & F. B. Brown, "Unresolved Resonance Treatment for the RACER Monte Carlo Code", *Proceedings of Intl. Conf. on Physics of Nuclear Sci. & Tech.*, Long Island, NY, Oct. 5-8, 1998 (1998).

- C. T. Ballinger, "The Direct S(a,b) Method for Thermal Neutron Scattering," *Proceedings of the ANS Intl. Conf on Mathematics & Computation, Reactor Physics, & Environ. Analyses,* April 30-May 4, Portland, Oregon (1995).

- R. G. Gamino, F. B. Brown, and M. R. Mendelson, "A Monte Carlo Green's Function Method for 3-D Neutron Transport," *Trans. Am. Nucl. Soc* **65**,237 (June 1992).

V. Vector and Parallel Monte Carlo

- S. Matsuura, F. B. Brown, and R. N. Blomquist, "Parallel Monte Carlo Eigenvalue Calculations," *Trans. Am. Nucl. Soc.* **71**, 199 (1994).

- F.B. Brown & W.R. Martin, "High Performance Computing and Monte Carlo", *Trans. Am. Nucl. Soc.* **91** [also LA-UR-04-4532] (Nov, 2004).

- F.B. Brown, J.T. Goorley, & J.E. Sweezy, "MCNP5 Parallel Processing Workshop", LA-UR-03-2228 (April, 2003).

- H.J. Alme, G.H. Rodrigue, G.B. Zimmerman, "Domain Decomposition Methods for Parallel Monte Carlo Transport," *J. Supercomputing*, **18**, 5-23 (2001).

- W. R. Martin, J. A. Rathkopf, and F. B. Brown, "The Impact of Advances in Computer Technology on Particle Transport Monte Carlo," *Proceedings of the ANS Topical Meeting on New Horizons in Radiation Protection and Shielding,* Richland WA (April 1992).

- F. B. Brown, W. R. Martin, and J. A. Rathkopf, "Particle Transport Monte Carlo and Parallel Computers," *Proceedings of the Argonne Theory Institute on Parallel Monte Carlo Simulations,* Argonne National Laboratory (August 1991).

- W. R. Martin and F. B. Brown, "Monte Carlo Methods for Particle Transport," *Trans. Am. Nucl. Soc.* **60**, 336 (1989).

- F. B. Brown, "Present Status of Vectorized Monte Carlo," *Trans. Am. Nucl. Soc.* **55**, 323 (1987).

- W. R. Martin and F. B. Brown, "Present Status of Vectorized Monte Carlo for Particle Transport Analysis," *International Journal of Supercomputer Applications,* Vol. 1, No. 2, 11-32 (June 1987).

- F. B. Brown, "Vectorization of 3-D General-Geometry Monte Carlo," *Trans. Am. Nucl. Soc.* **53**, 283 (1986).

- F. B. Brown and W. R. Martin, "Monte Carlo Methods for Vector Computers," *J. Progress in Nuclear Energy,* Vol. 14, No. 3,269-299 (1984).

- F. B. Brown, "Vectorized Monte Carlo," Ph. D. dissertation, University of Michigan, Ann Arbor, Michigan (1981).

- F. B. Brown, W. R. Martin, and D. A. Calahan, "Investigation of Vectorized Monte Carlo Algorithms," *Trans. Am. Nucl. Soc.* **39**, 755 (1981).

- F. B. Brown, W. R. Martin, and D. A. Calahan, "A Discrete Sampling Method for Vectorized Monte Carlo Algorithms," *Trans. Am. Nucl. Soc.* **38**, 354 (1981).

- F. B. Brown, D. A. Calahan, W. R. Martin, et al, "Investigation of Vectorized Monte Carlo Algorithms" working paper presented at the DOE Conference on High Speed Computing, Gleneden Beach, Oregon (April 1981).