

Monte Carlo Particle Transport

Particle Transport

Many important problems in physics and practical applications in technology involve the motion of particles through an object or a medium.

One of the first such applications to use a digital computer was the simulation of neutron transport in matter by von Neumann and Ulam[1]. They used the MANIAC computer at Los Alamos Laboratory and introduced the name “Monte Carlo” for their simulations with random numbers. The algorithms they introduced have been incorporated in sophisticated codes such as MCNP[2].

In high energy particle physics, codes such as GEANT[3] are used to simulate the trajectories of elementary particles produced in high energy collisions through particle detectors. Other important applications include the passage of diagnostic and therapeutic radiation through human tissue, simulation of radiation shielding in nuclear reactor facilities, oil-well logging by the use of neutrons and gamma rays to study rock formations, transport of photons and neutrons in fusion reactors, and many applications in astrophysics and space technology.

Radiation Shielding

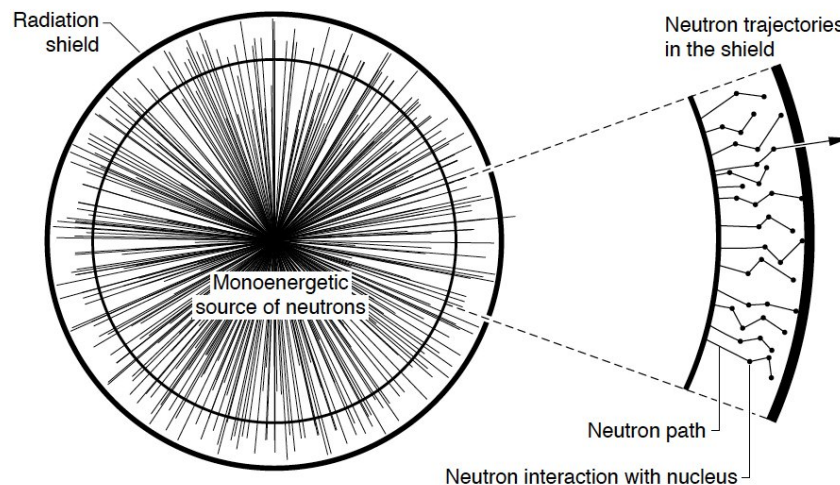


Figure 1: Simplified geometry for the radiation shield surrounding a nuclear reactor, from Ref. [2].

Fig. 1 shows a model of the radiation shield surrounding a nuclear reactor. A real shield has a complex three-dimensional shape and surrounds the reactor core in which neutrons are produced throughout the volume and slowed down by material in control rods. To simplify the simulation, assume that the neutrons emanate from a point source with fixed energy and random directions in a two-dimensional plane. Assume that the shield is a circular ring in the plane of motion of the neutrons. The neutrons undergo four types of interaction with nuclei in the the shield: elastic scattering, inelastic scattering, absorption, and fission. They travel in straight-line paths between interactions. Free neutrons have a finite lifetime and

spontaneous decay must also be taken into account. The object of the simulation is to estimate the fraction of neutrons that make it through the shield.

Simple model for physical effects

A realistic Monte Carlo code takes into account the detailed quantum mechanical probabilities for neutron interactions with all the different nuclear species in the reactor volume and the shield.

We will make some simplifying approximations. Assume that the shield has inner radius a and outer radius b , and that the position of a neutron in two-dimensional polar coordinates is r, θ .

Production in the Reactor: Neutrons are produced one at a time at $r = 0$ with speed v_0 and random polar angle θ chosen uniformly in $[0, 2\pi]$.

Motion in the Reactor: A neutron with $r < a$ moves immediately along its velocity direction to the inner wall of the shield. The reactor volume behaves like a vacuum!

Motion in the Shield: A neutron with $a \leq r \leq b$ has a mean-free path proportional to its speed: $\lambda(v) = v\tau$, where τ is a constant, the mean free time for speed $v = 1$. The neutron is moved a distance ℓ chosen randomly from an exponential probability distribution

$$P(\ell) = \frac{e^{-\ell/\lambda}}{\lambda}, \quad (1)$$

in the direction of its velocity.

Interaction in the Shield: After its free motion the neutron scatters from a shield nucleus, and is absorbed by it or breaks it up possibly producing additional neutrons.

Fission: For simplicity, assume that the nuclei in the shield are very stable (for example ^{56}Fe) and that the probability of fission can be neglected.

Absorption: The neutron is absorbed by the nucleus with probability $0 < p_a < 1$, where p_a is assumed to be a constant independent of speed.

Scattering: If the neutron is not absorbed (probability $1 - p_a$) it scatters and loses a fraction of its kinetic energy $v \rightarrow fv$ where $0 < f \leq 1$ is a constant independent of speed. The scattering is elastic if $f = 1$. Assume that the scattering cross section is isotropic: the velocity direction after scattering is chosen randomly in the interval $[0, 2\pi]$.

Neutron decay: The mean life of a free neutron is approximately 11 min, which is relatively long on the scale of propagation through the reactor and shield. Neutron decay will be neglected.

Intersection of the neutron's trajectory with shield boundaries

Suppose the neutron is at position \mathbf{r} and has velocity \mathbf{v} , and that a and b are the inner and outer radii of the shield. The position of the neutron moving forward from \mathbf{r} in the direction of its velocity is given by the vector $\mathbf{r} + t\mathbf{v}$ where t is the time elapsed. If $r < a$, the intersection of the trajectory with the inner surface of the shield is got by solving

$$(\mathbf{r} + t\mathbf{v})^2 = r^2 + t^2v^2 + 2t\mathbf{r}\cdot\mathbf{v} = a^2 \quad (2)$$

for the time

$$t = \frac{-2\mathbf{r}\cdot\mathbf{v} \pm \sqrt{(2\mathbf{r}\cdot\mathbf{v})^2 - 4v^2(r^2 - a^2)}}{2v^2}, \quad (3)$$

which needs to be positive so the neutron moves forward in the direction of \mathbf{v} :

$$t = \sqrt{\frac{a^2 - r^2}{v^2} + \frac{(\mathbf{r} \cdot \mathbf{v})^2}{v^4}} - \frac{\mathbf{r} \cdot \mathbf{v}}{v^2} . \quad (4)$$

If $a < r < b$ the neutron is in the shield. Its trajectory will first intersect the inner boundary if $\mathbf{r} \cdot \mathbf{v} < 0$ at

$$t = \frac{|\mathbf{r} \cdot \mathbf{v}|}{v^2} - \sqrt{\frac{(\mathbf{r} \cdot \mathbf{v})^2}{v^4} - \frac{r^2 - a^2}{v^2}} , \quad (5)$$

if and only if this solution is real and positive. Otherwise, the trajectory will first intersect the outer boundary of the shield at time

$$t = \sqrt{\frac{b^2 - r^2}{v^2} + \frac{(\mathbf{r} \cdot \mathbf{v})^2}{v^4}} - \frac{\mathbf{r} \cdot \mathbf{v}}{v^2} . \quad (6)$$

C++ Monte Carlo code for neutron shielding

The following code implements the neutron transport Monte Carlo algorithm.

_____ Program 1: <http://www.physics.buffalo.edu/phy410-505/topic2/neutron.cpp> _____

```
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
using namespace std;

#include "random.hpp"
using namespace cpl;

Random rg; // random number generator object

const double pi = 4 * atan(1.0); // value of pi

double a = 1, // inner radius of shield
       b = 1.5, // outer radius of shield
       tau = 0.2, // mean free time for v=1
       p_a = 0.05, // absorption probability
       f = 0.95; // inelasticity parameter

double v_0 = 1; // initial speed

double operator * ( // define * to compute dot product
    const vector<double>& v1,
    const vector<double>& v2
) {
    double sum = 0;
    for (int i = 0; i < v1.size(); i++)
        sum += v1[i] * v2[i];
}
```

```

    return sum;
}

struct Neutron {                                // define C++ Neutron object

    vector<double> r, v;                          // position and velocity
    bool absorbed, escaped;                       // final fate

    Neutron() {                                  // constructor initializes neutron

        r = vector<double>(2, 0);
        double theta = 2 * pi * rg.rand();
        v.push_back(v_0 * cos(theta));
        v.push_back(v_0 * sin(theta));
        absorbed = escaped = false;

    }

    bool step_succeeded() {                      // attempt a Monte Carlo transport step

        if (absorbed || escaped)
            return false;

        double r_mag = sqrt(r*r);

        if (r_mag > b) {                          // outside the shield

            escaped = true;
            return false;

        }

        if (r_mag < a)                            // inside core region
            move_to_a();

        move_free();                              // inside the shield
        interact();

        return true;
    }

    void move_to_a() {                            // move to inner wall of shield

        double r2 = r * r, v2 = v * v, rv = r * v;
        double t = sqrt((a*a - r2)/v2 + rv*rv/v2/v2) - rv/v2;

        r[0] += t * v[0];
        r[1] += t * v[1];

    }
}

```

```

void move_free() {          // free motion in shield

    double r2 = r * r, v2 = v * v, rv = r * v;
    double t_max;
    if (rv < 0 && abs(rv) >= v2 * (r2 - a*a))
        t_max = abs(rv)/v2 - sqrt(rv*rv/v2/v2 - (r2 - a*a)/v2);
    else
        t_max = sqrt((b*b - r2)/v2 + rv*rv/v2/v2) - rv/v2;

    double t = - tau * log(rg.rand());
    if (t > t_max)
        t = t_max;

    r[0] += t * v[0];
    r[1] += t * v[1];
}

void interact() {          // interaction with nucleus in shield

    if (rg.rand() < p_a) { // absorbed by nucleus

        absorbed = true;

    } else {                // scatter from nucleus

        double v_mag = f * sqrt(v*v);
        double theta = 2 * pi * rg.rand();
        v[0] = v_mag * cos(theta);
        v[1] = v_mag * sin(theta);
    }
}

};

int main() {

    cout << " Neutron Reactor Shielding Monte Carlo\n"
         << " -----\n"
         << " Enter number of particles to simulate: ";
    int n;
    cin >> n;

    // set random number generator seed
    rg.set_seed_time();
    cout << " Using " << rg.get_algorithm()
         << " and seed " << rg.get_seed() << endl;

    ofstream file("neutron.data");
    int step_sum = 0, absorbed = 0, escaped = 0;
    for (int i = 0; i < n; i++) {

```

```

Neutron neutron;
int steps = 0;
do {
    file << neutron.r[0] << '\t' << neutron.r[1] << '\n';
    ++steps;
} while (neutron.step_succeeded());
file << '\n';

step_sum += steps;
if (neutron.absorbed)
    ++absorbed;
if (neutron.escaped)
    ++escaped;
}
file.close();

cout << " Number escaped          = " << escaped << '\n'
     << " Number absorbed         = " << absorbed << '\n'
     << " Averde number of steps = " << step_sum/double(n) << '\n'
     << " " << n << " trajectories in file neutron.data" << endl;
}

```

Photon Diffusion in the Sun

The Sun is a gigantic thermonuclear reactor that emits energy at the rate of 4×10^{26} W and delivers 350 W/m^2 to power the Earth.

The transport of gamma rays and X-ray photons, which are produced mainly in the core of the Sun, through its outer layers can be modeled like the transport of neutrons through a reactor shield.

Homework Problem

Adapt the neutron transport code to model the diffusion of photons through the radiation layer of the Sun. You can model the Sun as a two dimensional circle so that code does not need to be generalized to three dimensions. From NASA's Solar website <http://solarscience.msfc.nasa.gov/interior.shtml>, choose reasonable values for the simulation parameters. Run the Monte Carlo to compute the average time it takes for a photon to diffuse through the radiation layer. Hint: It takes a photon a long time to get through the layer, so start with a scaled-down model, figure how the time depends on the scale factor, and then scale up to the real thing.

References

- [1] "Stan Ulam, John von Neumann and the Monte Carlo Method", R. Eckhardt, Los Alamos Science, Special Issue (1987), <http://library.lanl.gov/cgi-bin/getfile?00326867.pdf>.

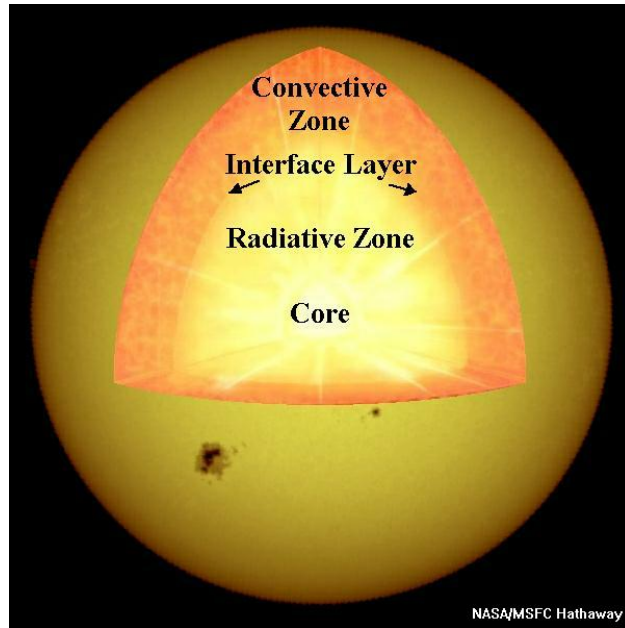


Figure 2: Interior of the Sun from Ref. [4]. Photons produced in thermonuclear burning are transported through the radiative zone into the outer layers of the Sun.

- [2] "A Monte Carlo Code for Particle Transport", J.S. Hendricks, Los Alamos Science (1994),
<http://www.fas.org/sgp/othergov/doe/lanl/pubs/00326727.pdf>.
<http://mcnp-green.lanl.gov/index.html>.
- [3] GEANT4 (GEometry ANd Tracking) Monte Carlo: <http://www.geant4.org/>.
- [4] The Solar Interior, NASA Marshall Solar Physics Website,
<http://solarscience.msfc.nasa.gov/interior.shtml>.