PHY 420 Report: Attenuation of Solar Energy Through a Tree

Aysha Rahman

May 9, 2018

Abstract

Solar panels in urban areas or in home and business settings are often surrounded by trees so that sunlight is blocked at certain angles during the day. This greatly reduces the output energy of the solar panels. To assess the attenuation of light through a dormant tree, Monte Carlo simulations of a randomly branching tree are designed. Tree branching is simulated under the following assumptions: that the tree branches in ten iterations, that it branches into two each time, that each new branch is a fraction of the length of the previous one, that the total thickness or volume of all branches in an iteration is equal to the thickness of the trunk, and that the angle of branching is no more than 90 degrees. After simulating the structure of the tree, a path of sunlight is integrated through the tree's branches to determine the fraction of light blocked for a given range of angles.

1 Introduction

The Bradley Observatory at Agnes Scott College has solar panels on its roof. However, as it is located in an area with many trees, one of the panels is blocked by a tree. This greatly reduces the energy output of the solar panels, even in the winter when the tree is dormant. To investigate how much this tree affects energy output, a Monte Carlo simulation of light passing through a randomly generated dormant tree is designed to measure the amount of light attenuated by the branches. The program is not yet finished; this report documents progress thus far.

2 Assumptions

Some initial assumptions and observations are as follows: Every branch diverges into two each time the tree splits. When the split occurs, the two new branches are no more than 90 degrees apart from one another. The tree branches 10 times; by observation, the trees nearby the area generally split between 7 and 14 times, and therefore 10 is a good assumption to set in the code. The overall thickness of the tree does not change; if the entire tree were to be collimated, the thickness would remain constant. The length of each of the new branches is shorter than the length of the previous one.

3 Creating the Tree

To create the tree, we consider four variables: the length of each branch, its thickness, its horizontal or azimuthal angle φ, and its vertical angle θ.

We consider r to be a random number between 0 and 1, L to be the length of a branch, t to be the thickness, φ to be the horizontal or azimuthal angle, and θ to be the vertical angle. Then we generate each property by these rules:

1. For each new branch L_n, its length is equal to the length of the previous branch L_{n-1} times the random number r_L between zero and one, given by the equation

$$L_n = L_{n-1} * r_L. \tag{1}$$

2. For the thickness, we consider the thickness of a left branch $T_{n,left}$ vs the thickness of a right branch $T_{n,right}$ and multiply by a new random number r_T to get the first branch thickness. The remaining branch thickness is calculated from the first thickness. This is described by the following equations:

$$T_{n,left} = T_{n-1} * r_T \tag{2}$$

$$T_{n,right} = T_{n-1}(1 - r_T). \tag{3}$$

3. For the vertical angle θ, which comes down from the z axis onto the x-y plane, we assume that the branch splits symmetrically along its own axis and say that the θ for the left and the right are the same

$$\theta_{left} = \theta_{right}. \tag{4}$$

4. For the azimuthal angle φ, which rotates the new branches along the x-y plane, we assume that the branches split along this angle exactly opposite each other, so we say

$$\varphi_{left} = \varphi_{right} - \pi. \tag{5}$$

As mentioned previously, the tree is a collection of points inside a 3-dimensional grid. These points are stored in the code as arrays, with one each for the x, y, and z coordinates, as well as one for length and one for thickness. Each array has 2047 points, where we store the beginning and endpoints of each branch. This number comes from the number of tree branches. From our assumptions, we say there are 10 iterations and 2 new branches at every split, so the total number of branches is

$$\sum_{n=0}^{10} 2^n = 2047. \tag{6}$$

4 Coordinate Systems

We use two different kinds of coordinate systems here: a global one and a local one. We think of our global coordinate system being a cube, a 3-dimensional grid with the points where the tree exists filled in. We can say the "lines" of the grid use a step size that we'll define in the code, which we choose to be as small as the computer can handle. The global coordinate system has its origin at the base of the tree. The point where the trunk splits is 0, 0, L_0, with L_0 being the length of the trunk.

Each branch has its own local coordinate system. This is used to generate the new branches, and the origin is located at wherever the parent branch splits into two, with the z axis being along the parent branch. In order for the new branches to be useful, they must be converted back into the global coordinate system. We use the following rotation matrix to do this,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} cos\varphi cos\theta & -sin\varphi & cos\varphi sin\theta \\ sin\varphi cos\theta & cos\varphi & sin\varphi sin\theta \\ -sin\theta & 0 & cos\theta \end{pmatrix} \tag{7}$$

where θ is the azimuthal angle that goes from 0 to 2π and φ is the vertical angle that goes from 0 to π.

5 Code

The following is the code created to generate a random tree.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>

/************************************************************
* Program to calculate the branches of a tree.
*
*       Units:
*               Length = m
*               Time   = sec
*
************************************************************/

int main(int argc, char *argv[])
{
        double tpi = 6.283185308;
        double pi = 3.141592654;
        int i,j,k=0,m,n,p,q,t=0,s,l=0,Nlocal, Lsteps, rsteps;
        int MAXPTS=64000,MAXLOCAL=1000;
        FILE *outf;
        double stepsize=0.1, L0=10.0, r1, r2,r3,r4,t0=1;
        double A00, A01, A02, A10, A11, A12, A20, A21, A22;
        double x0[2047], y0[2047], z0[2047], L[2047], thickness[2047];
        double cost[2047],sint[2047],phi[2047],cosp[2047],sinp[2047];
        double xg[MAXPTS], yg[MAXPTS], zg[MAXPTS];
        double xl[MAXLOCAL], yl[MAXLOCAL], zl[MAXLOCAL];
        char outfile[128];

        /* initialize */
        srand(time((time_t*)NULL));
        for(i=0; i<2048; i++)
        {
                L[i] = 0.0;
                thickness[i] = 0.0;
                x0[i] = 0.0;
                y0[i] = 0.0;
                z0[i] = 0.0;
        }
        for(i=0; i<MAXPTS; i++)
        {
                xg[i] = -1.0;
                yg[i] = -1.0;
                zg[i] = -1.0;
        }
        L[0] = L0; /* this could be user-specified */
```

```
thickness[0] = t0;
k=1;
while(k < 2048)
{
/* generate a new branch */

/* branch length */
         r1 = 1.0*rand()/RAND_MAX;
         L[k]=fabs(L[k-1]*r1);
         L[k+1] = L[k];
         r2 = 1.0*rand()/RAND_MAX;
         thickness[k]=fabs(thickness[k-1]*r2);
         thickness[k+1] = thickness[k-1]*(1-r2);
         Lsteps = L[k]/stepsize+1;
         rsteps = thickness[k]/stepsize+1;
                  for(m=0;m<rsteps;m++)
                  {
                  for(p=0;p<rsteps;p++)
                  {
                  for(q=0;q<Lsteps;q++)
                  {
                  xl[l] = -thickness[k]+m*stepsize;
                  yl[l] = -thickness[k]+p*stepsize;
                  zl[l] = q*stepsize;
                  l++;
                  }}}
         Nlocal = Lsteps*rsteps*rsteps +1;

         /* compute branch direction */
                  r3 = 1.0*rand()/RAND_MAX;
                  /* direction cosine of z coordinate 0-pi/2 */
                  cost[k]=r3;
                  sint[k]=sqrt(1.0-cost[k]*cost[k]);
                  cost[k+1] = cost[k];
                  sint[k+1] = sint[k];
                  /* one branch is mirror image of the other -- off by pi */
                  r4 = 1.0*rand()/RAND_MAX;
                  phi[k]=tpi*r4;
                  phi[k+1] = phi[k]-pi;
                  cosp[k] = cos(phi[k]);
                  sinp[k] = sin(phi[k]);

         /*  Rotation matrix */
         A00 = cosp[k]*cost[k];
         A01 = -sinp[k];
         A02 = cosp[k]*sint[k];
         A10 = sinp[k]*cost[k];
         A11 = cosp[k];
         A12 = sinp[k]*sint[k];
         A20 = -sint[k];
```

```
                    A21 = 0.0;
                    A22 = cost[k];
                    for(l=0;l<Nlocal;l++)
                    {
                    xg[t] = A00*xl[l] + A01*yl[l] + A02*zl[l] + x0[k];
                    yg[t] = A10*xl[l] + A11*yl[l] + A12*zl[l] + y0[k];
                    zg[t] = A20*xl[l] + A21*yl[l] + A22*zl[l] + z0[k];
                    t++;
                    }

                    x0[k+1] = A02*L[k] + x0[k];
                    y0[k+1] = A12*L[k] + y0[k];
                    z0[k+1] = A22*L[k] + z0[k];
                    k++;
                    }
}
```

6 Results and Future Work

The code did not run initially, though it ran on a different machine. The existing code must still be run.

However, this program only generates a tree; future work would involve integrating a beam of light at various angles through the tree and measuring the amount of light blocked. The angle of the beam can be changed to consider the location of the sun at different times of day and different points in the year.