Fast 2D and 3D Lightning Simulation Using Finite Difference Methods and Green’s Function Approximation of Point Charges

It is often desirable in graphics to model physically complex phenomena such as fire, clouds, explosions, and water. The difficulty lies in the fact that these phenomena are not well described by typical representations of objects in a 3D model. They arise from complex interactions between many components and are computationally intensive to model as simple particles or objects.

One such phenomenon is lightning. Lightning is the really a multistep process that contains several bolts that the human eye perceives as flickering of a single bolt. The first stroke is referred to as the “stepped leader,” while the subsequent bolts are referred to as “dart leaders.” The dart leaders do not show the same degree of branching as the stepped ladder and follow roughly the same path carved by the stepped leader through the air. The process is initiated by a large negative potential buildup in a cloud that surpasses the ionization energy of air molecules. A column of air is then ionized into plasma, which is highly conductive, and allows the negative potential to build up at the tip of this new column of air. The process then repeats, branching with a certain frequency (fractal dimension of 1.7) that is also observed in other natural phenomena such as ice crystal formation, lichen, and fracture patterns.\(^1\)

These patterns are classified as Laplacian Growth phenomena and are associated with three common techniques for simulating them. They are Diffusion Limited Aggregation, Hastings-Levitov conformal mapping, and the Dielectric Breakdown Model (DBM).\(^2\) After the stepped leader reaches a potential that is grounded (able to neutralize its charge), the propagation stops and a large positive current traverses the stepped ladder’s path up to the cloud. This is what causes the bright flash that is associated with lightning. The potential at the initial location in the cloud, or a nearby location in the cloud, can be recharged by the surrounding region to a highly negative leader to the ground, repeating the ground-cloud positive current and resulting in another bright flash of light. This can happen multiple times, causing the flickering one sees when observing lightning since the flashes occur too fast for the human eye to distinguish. This same model can apply to any electrical discharge including arcs between two highly charges sources and sinks.

The DBM is able to replicate this subtle nuance of lightning if one adds some positive potential to the path that the stepped ladder took to reach the ground. In this way, the subsequent dart leaders are more statistically likely to traverse the same path. The model divides the 2D or 3D space into a lattice of nodes. Each node has a potential \(\phi_i\). The initial potential is calculated based on the boundary conditions (locations of the

\(^1\) Kim, 2007 p. 2
\(^2\) Ibid.
cloud and ground). The source which is modeled as a single node is then evolved in time by calculating the probability that an adjacent node will be chosen as the next node the bolt passes through. The set of adjacent nodes, N, is updated on each evolution and the probabilities recalculated.

\[ p(n_i) = \begin{cases} 
\frac{(\varphi_i)^\eta}{\sum_{k \in N} (\varphi_k)^\eta}, & \text{if } n_i \in N \\
0, & \text{otherwise}
\end{cases} \]  

The parameter \( \eta \) corresponds to the desired level of branching. Small \( \eta \) corresponds to dense branching whereas high \( \eta \) corresponds to focused progression towards the ground. One can also better direct the lightning with the addition of boundary conditions which correspond to an input electrical potential field to bias the probabilities. As a result, this model is able to control the production of lightning in an intuitive way.\(^3\)

Computing the Laplacian of at each step of the calculation is costly and slows down the computation greatly. The Laplace equation is reproduced below.

\[ \nabla^2 \varphi(x, y, z) = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0 \]  

This is an instance of the more general Poisson Equation reproduced below:

\[ \nabla^2 \varphi(x, y, z) = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = F(x, y, z) \]  

A 2\(^{nd}\) order accurate Finite Difference scheme is described as follows to approximate the 2\(^{nd}\) derivatives of the electric potential.

\[ \frac{\partial^2 \varphi}{\partial x^2} \bigg|_i = \frac{(\varphi_{i+1} - 2\varphi_i + \varphi_{i-1})}{h^2} + O(h^2) \]  

\[ \frac{\partial^2 \varphi}{\partial y^2} \bigg|_j = \frac{(\varphi_{j+1} - 2\varphi_j + \varphi_{j-1})}{h^2} + O(h^2) \]  

\[ \frac{\partial^2 \varphi}{\partial z^2} \bigg|_k = \frac{(\varphi_{k+1} - 2\varphi_k + \varphi_{k-1})}{h^2} + O(h^2) \]

When combined into one equation, this results in the following scheme for updating the potential of a given cell based on its neighbors potentials.

\[ \varphi_{i,j,k}^{t+1} = \frac{1}{6} (\varphi_{i+1,j,k}^t + \varphi_{i-1,j,k}^t + \varphi_{i,j+1,k}^t + \varphi_{i,j-1,k}^t + \varphi_{i,j,k+1}^t + \varphi_{i,j,k-1}^t) - \frac{h^2}{6} F_{i,j,k} \]  

One terminates the iterations when max relative error in a given iteration falls below a user defined threshold:

\[ \max(\varepsilon_{i,j,k}) \leq \text{Threshold}, \quad \text{where} \quad \varepsilon_{i,j,k} = \left| \frac{\varphi_{i,j,k}^{t+1} - \varphi_{i,j,k}^t}{\varphi_{i,j,k}^{t+1}} \right| \]  

To improve the convergence, the method of successive over relaxation was employed. The scheme is described as follows:

\[ \varphi_{i,j,k}^{t+1} = \frac{1}{6} (\varphi_{i+1,j,k}^t + \varphi_{i-1,j,k}^t + \varphi_{i,j+1,k}^t + \varphi_{i,j-1,k}^t + \varphi_{i,j,k+1}^t + \varphi_{i,j,k-1}^t) - \frac{h^2}{6} F_{i,j,k} \]  

\[ \varphi_{i,j,k}^{t+1} = (1 - \omega) \varphi_{i,j,k}^t + \omega \varphi_{i,j,k}^{t+1} \]  

\(^3\) Bickel, 2006. P.3
Where $\omega$ is the relaxation factor. When $\omega$ is less than unity, the scheme is under relaxed. When $\omega$ is greater than unity, the scheme is over relaxed. For moderately high values of $\omega$ for an over relaxed scheme, “the convergence is accelerated” whereas for “high values of $\omega$, the convergence is oscillatory, and for very high value of $\omega$, the oscillations become divergently oscillatory and the iterative scheme becomes unstable. Under relaxed schemes seldom diverge, but the rate of convergence may become painfully slow.” A method for calculating the optimal $\omega$ for a 2D $N \times M$ rectangular grid is given below:

$$\omega = \frac{4}{2 + \sqrt{4 - \left(\cos \left(\frac{\pi}{N-1}\right) + \cos \left(\frac{\pi}{M-1}\right)\right)^2}}$$  \hspace{1cm} (10)

While this scheme has been optimized, computing the Laplacian after a new potential site has been chosen is costly and results in an $O(n^3)$ overall running time where $n$ is the number of grid cells. To reduce the computational complexity to $O(n^2)$, one can simply calculate the Laplacian potential distribution once at the start of the calculation and then approximate the local changes to the potential field. A solution developed by Kim et al. does this by approximating the potential due to a point charge with the solution to the 3D radial Laplacian equation reproduced below:\textsuperscript{4}

$$\nabla^2 \varphi(r, \theta, \beta) = \frac{\partial^2 \varphi}{\partial r^2} + \frac{1}{r^2} \frac{\partial \varphi}{\partial \theta^2} + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 \varphi}{\partial \beta^2} + \frac{2 \partial \varphi}{r \partial r} = 0$$  \hspace{1cm} (11)

$$\varphi(r) = c_1 + \frac{c_2}{r}$$  \hspace{1cm} (12)

$$c_1 = -\left(\frac{R_1}{R_2} - 1\right)^{-1}$$  \hspace{1cm} (13)

$$c_2 = -\left(\frac{1}{R_2} - \frac{1}{R_1}\right)^{-1}$$  \hspace{1cm} (14)

$$\varphi(r) = 1 - \frac{R_1}{r} \quad (as \, R_2 \rightarrow \infty)$$  \hspace{1cm} (15)

Using the 3D Green’s function, Eq. 15, one can treat the potential in each grid cell as the superposition of many point charges plus the original potential due.

$$\varphi_i = \sum_{j=0}^{n} \left(1 - \frac{R_1}{r_{ij}}\right) + \varphi_o$$  \hspace{1cm} (16)

Then one can transform the scheme into an iterative scheme for updating the potentials in the electric field by storing the last computed value and adding the contribution of any new point charges.

$$\varphi_i^{t+1} = \varphi_i^t + \left(1 - \frac{R_1}{r_{i,t+1}}\right)$$  \hspace{1cm} (17)

This allows the initial Laplacian to be applied just once over the boundary conditions for the potential field using a finite difference method and then updated according to Eq. 7 for the addition or removal of charge. Although this is an approximation to the

\textsuperscript{4} Kim et al., 2007 P. 3
true potential, it is close enough to the true potential and can be calculated in \(O(1)\) time. This speeds up the propagation step for each candidate site to \(O(n)\), where \(n\) is the number of grid cells, as opposed to the time it takes to compute the Laplacian. Thus the overall algorithm runs in \(O(n^2 + G(n))\) where \(G\) is the rate of convergence for the Laplacian iteration. The bolt propagation step has \(O(n)\) iterations and performs \(O(n)\) updates so overall it is \(O(n^2)\).

**Summary of Algorithm in 3D**

**Input:**

\(E[\text{Width}][\text{Length}][\text{Height}]\), a multidimensional grid of electric potential values. Initially all values are set to 0. \(E[x][y][z] = 0\).

**Initial Boundary Conditions:**

Along each extreme surface, e.g. \(x = 0\) or \(x = \text{width} - 1\), the potential is forced to be some constant, usually zero. This is true for \(y\) and \(z\) as well. This corresponds to the Dirichlet boundary conditions and the potential can be any constant value as long as the values are specified along the extremes. In the notation used above this corresponds to:

\[
E[0][0][z] = E[0][y][0] = E[x][0][0] = E[\text{Width} - 1][\text{Length} - 1][z] = E[\text{Width} - 1][y][\text{Height} - 1] = E[x][\text{Length} - 1][\text{Height} - 1] = 0
\]

**Additional Boundary Conditions:**

This step is optional. The user specifies certain cells to have certain potentials. At least one cell must be specified to have a positive potential otherwise there will only be a trivial solution to the Laplace equation, \(\varphi(x,y,z) = 0\), and thus no lightning bolt will actually propagate. By choosing the locations of the positive potentials, the lightning bolt will be attracted to them and eventually terminate on one of them. One can also specify negative or zero potentials to repulse or make an area neutral respectively. This can be used to guide the shape of the lightning bolt.

**Initial Potential Based on Initial Boundary Conditions:**

The Laplace Equation in 3D in Cartesian coordinates can be solved iteratively by the scheme described in Eqs. 2 – 10. The successive over relaxation is used to speed the convergence.

**Bolt Propagation:**

The candidate sites are determined in \(O(n)\) time. The chosen site is determined in \(O(n)\) time according to Eq. 1. If the addition of repulsors is allowed, sites with negative potentials are not considered. After the site is determined, it takes time \(O(n)\) to update the entire grid according to Eq. 17 to remove the charge due to the chosen site and to deposit residual positive charge at that location if the bolt is the stepped leader. Once a boundary is reached, the bolt propagation terminates.

**Rendering of the Bolt:**
After the bolt shape is determined, one can choose how to render the bolt. For simplicity each location was rendered as white point (for the 2D images produced) or a sphere in GL (for the 3D application).

**Deliverables**

- 2D Command Line Program that demonstrated the algorithm worked correctly.
  - This program creates a ring of positive charge as the initial charge configuration and propagates a bolt from the center of the ring to the edge of the ring. Different initial charge configurations and starting points were used to create the sample images as well as to change a lot of the user parameters such as \( \omega \) and \( \eta \).
  - Sample Images created with the program demonstrating the following:
    - Laplacian Potential Field Due to Given Boundary Conditions. This calculation converged in only 1836 iterations although it was 1024 by 1024 pixels. This is because the successive over relaxation was used with the optimal \( \omega \).
      - TwoRing10241024_1836Iterations_optso.tiff
    - Bolt (with Fixed Seed) attracted to positive point charge. There are two point charges, one at the top center and one at the right center of the image. The number after seed is the seed used in C’s srand() call at the start of the program. One can see from the images that the bolt goes towards one or the other point charge based on the seed. In the final image in this series, one can see that the second strike happens to go towards the first point charge when the Seed is fixed at 4.
      - Bolt1-Seed0.tiff
      - Bolt2-Seed2.tiff
      - Bolt3-Seed3.tiff
      - Bolt4-Seed4.tiff
      - Bolt5-Seed4-2Strikes.tiff
    - Multiple Bolts follow original path of the stepped leader. The rho parameter changes how strong the dart leaders are attracted to the stepped leader. The number after rho tells the value of rho used to make the image. The underscore is a decimal point; e.g. 0_10 == rho = 0.10.
      - Bolt8-Seed4-2Strikes-Rho0_10.tiff
      - Bolt9-Seed4-2Strikes-Rho1_00.tiff
      - Bolt10-Seed4-2Strikes-Rho0_00.tiff
    - Bolts propagating in a spherical charge initial configuration
      - Bolt0SphereSeed4.tiff
      - Bolt1SphereSeed4.tiff
    - Cloud to Ground Lightning with multiple bolts
      - CloudGround_1Bolt.tiff
- CloudGround_3Bolts.tiff
- CloudGround_3Bolts_2.tiff
- CloudGround_1Bolt_1024x1024.tiff

- Program Name: Efield_test
- Sources: Efield.cpp Efield.h Image.cpp Image.h figio.cpp figio.h Efield_test.cpp
- 3D Command Line Version Name: Efield3D_test
  - Note this only produces an image for the depth/2 slice of the bolt. This was mainly used to test that the 3D bolt was being calculated correctly.

- 3D Application that interactively demonstrates and visualizes the 3D bolt calculated.
  - Program Name: Efield3D_UI2
  - Sources: Efield3D2.cpp Efield3D2.h Image.cpp Image.h figio.cpp figio.h Efield3D_UI.cpp ModelerCamera.cpp ModelerCamera.h glDraw.cpp glDraw.h BoltUI2.cpp BoltUI2.h BoltView2.cpp BoltView2.h
  - The user can change the width, height, and depth of the 3D volume the bolt is being rendered in. The smaller these values are the less memory the application uses and the faster the computation.
  - The user can choose to visualize the bolt in 2D, which views the current zplane value as given by the value slider or in 3D which views the entire volume. In viewing mode, when edit mode is unchecked, the user can also use the mouse left button and motion to rotate the view, the right mouse button and motion to zoom in or out, and the middle mouse button to translate the viewpoint.
  - In edit mode, the user can edit the current zplane in one of two modes. The zero mode, adds boundary conditions of zero charge to the cell the user left clicks. In positive mode, the user adds boundary conditions to cells the user clicks of a given potential that is defined by the value currently in potential. It is initially set to one but can be modified by the user. The user can right mouse click to change the start of the propagating bolt. Note that the user can only edit in 2D mode, otherwise mouse clicks have no effect when edit mode and 3D mode are selected.
  - The Zplane slider affects which zplane is being viewed or edited.
  - The number of bolts parameter indicates how many bolts to use for the next simulation. The residual positive charge model will be used when this parameter is more than 1.
  - Eta, or \( \eta \), is the branching factor to be used in the bolts calculation. Small eta result in high branching and low eta in little branching.
  - Screenshot of Edit Mode
Screenshot of 3D Mode

File Details
- Cell Class
The cell class was the basic grid cell unit that has a specific position in 2D or 3D space. It has a potential value and a Boolean flag indicating whether or not it is a boundary condition.

- **Efield Class:** Efield.cpp Efield.h Efield3D2.cpp Efield3D2.h
  - The Efield class was constructed as a multidimensional array of Cell classes and provides methods for computing the Laplacian for the given initial boundary conditions as well as propagating a bolt of lightning given an initial position. It also provides a method for computing the optimal relaxation parameter to speed up the Laplacian calculation as described in Eqs. 8-10. Additionally, it provides an OpenGL Method for drawing a representation of the boundary conditions for editing the initial boundary conditions or for visualizing the calculated bolt. It stores the last computed bolt as an STL vector of Cell*. It also provides a method for saving the 2D slice of a given zplane (in 3D) or the 2D bolt to an image (as used in Efield_test to create the sample images).

- **Image.cpp Image.h Figio.cpp Figio.h**
  - The latter two files were adapted from code given to the author for a Computer Graphics assignment. It contains methods for loading and saving char buffer data to the ppm and tiff image formats. Image.cpp and Image.h provide another layer of abstraction over that to work with the char buffers and to align the image coordinates with that of the char buffer coordinates.

- **glDraw.h glDraw.cpp**
  - These files contain methods for drawing texture mapped OpenGL primitive objects in local coordinates and uses display lists to speed up the drawing.

- **ModelerCamera.cpp ModelerCamera.h**
  - These files were adapted from code given to the author for a Raytracer program. The camera uses OpenGL calls to allow the user to control the camera with the mouse.

- **BoltUI2.cpp BoltUI2.h**
  - These files contain FLUID generated code, (see LightningUI5.fl) for the user interface for the 3D application. It links the interface interaction with function calls to the BoltView class.

- **BoltView2.cpp BoltView2.h**
  - These files contain the logic of the 3D applications functionality. It calls methods in Efield3D2 to compute the bolt with the desired parameters and to update the Efield class accordingly. Part of the code was generated by FLUID (see BoltView2.fl).

**Design Considerations and Decisions**

- The Efield class was implemented as a multidimensional array to improve speed of access performance. Instead of using multiple arrays of different types for each
of the various parameters, a single array of class Cell was used. This was done to improve the locality of the code since most of the parameters of class Cell would be looked up in close proximity so looking up a number of different parameters in different arrays would be inefficient.

- The final bolt calculated was stored as a STL vector of class Cell* since the bolt would mostly be processed as a whole and storing the pointers rather than the actual cells would reduce memory usage.
- The uniform array was chosen over an adaptive structure like a quadtree or an octree (partially due to time constraints) but also because the initial bottleneck was the calculation not the memory allocation. After the speedups to the calculations, this step of the algorithm is actually the bottleneck but it only occurs when new memory must be allocated, which occurs when the user changes the grid dimensions.
- The negative potential that is the source of lightning bolt is mapped to a zero potential so that in the growth step, cells already part of the growing bolt are not added. The user can add zero boundary conditions to regions that they do not want the bolt to move towards. Positive potentials correspond to the ground in typical ground to cloud to ground descriptions of lightning. The bolt is attracted to these regions and the user can add these positive boundary conditions by selecting positive mode and left clicking with the mouse.
- The user can click the reset field button to reset the initial boundary conditions to just the bounding box of the volume as well as reset the camera.
- The rho parameter, which controls the level of attraction the dart leaders have towards the stepped leader’s initial path, was not added as a user parameter because a good parameter range was not tested for it.
- A simplistic method of rendering the lightning was used, simple spheres or points, as opposed to a more complicated model researched due to unfamiliarity with OpenGL. The bolt is colored based on the distance the bolt is from its starting point. The starting point’s color is blue and the bolt becomes redder the farther away it is from there.

Future Work

- The addition of repulsors by allowing negative values of potential could create even more control of the lightning.
- A more physically accurate model of the lightning can be used to render the computed bolt in OpenGL.
- Due to the speed of the computation, the calculation can be used to generate animation speed lightning barring improvements in the memory allocation.

Resources

- FLTK
- OpenGL
- Figio.h Figio.cpp: CPSC 478 Assignment 1 Code 2012
- ModelerCamera.h ModelerCamera.cpp: CPSC 478 Raytracer Code 2012
Summary
Using Finite Difference methods on a uniform grid and accelerating the convergence of the iterative scheme with successive over relaxation, the solution to the 2D and 3D Laplace equation can be computed in $O(G(n))$ time, where $n$ is the number of cells in the uniform grid and $G(n)$ is the rate of convergence of the iterative scheme given $n$. By approximating local changes in the potential field with the 3D Green’s function, the lightning bolt propagation can be calculated in $O(n^2)$ as opposed to $O(n*G(n))$ since $G(n) >> n$ generally.

Bibliography


Kim, T.; Lin, M.C.; "Physically based animation and rendering of lightning."
DOI: 10.1109/PCCGA.2004.1348357


Glassner, A.; , "The digital ceraunoscope: synthetic thunder and lightning. 1,"
DOI: [http://dx.doi.org/10.1109/38.824552](http://dx.doi.org/10.1109/38.824552)

Glassner, A.; , "The digital ceraunoscope: synthetic thunder and lightning. 2,"
DOI: [http://dx.doi.org/10.1109/38.844376](http://dx.doi.org/10.1109/38.844376)

DOI: 10.1103/PhysRevLett.52.1033

DOI: 10.1007/s00371-006-0061-z

DOI: 10.1109/PCCGA.2001.962896

