Justin Burzachiello

AMS 595: Final Project Report

1. Information required to run the code:

Running this code simply requires the user to have access to software which can run iPython (aka Jupyter) notebooks. The codebase includes a file called *ams595_finalProject.ipynb* which runs the code relevant to this project. While this iPython file does include all necessary lines of code for importing software dependencies, a Python file called *ams595_finalProject.py* is also included. This Python file is simply a copy of the aforementioned .ipynb file, but in a .py format. Running this .py file requires the user to have access to Python 3 and Pip since the Python code in this project relies on the Numpy and Matplotlib libraries from PyPI. The author of this software recommends that users simply download the provided .ipynb file and run it in Google Colab.

2. Inputs / outputs of the code, and how to use it:

Running this code is best done by downloading the provided .ipynb file and running it in Google Colab without any changes. However, while user inputs are not explicitly required for the code to work, a user would likely want to reinstantiate certain variables in the code. This is because a user can specify the name of the output animation .gif files, the path to where these .gif files are downloaded to, and various parameters which govern the nature of the dielectric breakdown simulation.

At the time of writing this report the last cell in the .ipynb file is the “user interface” – previous cells represent the source code of the project. Therefore, the easiest way to use the code is to start off by running all cells except for the last one, specifying the parameters of the computational dielectric breakdown model in the last cell, then running the last cell. The name of the generated .gif files is instead currently specified in the run_dbm() function.

Possible variables that the user can adjust to alter the simulation are:

a. *num_tree_cells*: the number of cells that the resulting electrical tree will have;
b. *N*: the width of the square grid upon which the DBM model is confined;
c. *threshold*: the convergence threshold for numerically solving the electrostatic Laplace equation on the aforementioned square grid;
d. *boundary_val*: the fixed electrostatic scalar potential field value upon the outermost cells of the grid (the potential of the cells of the electrical tree are also fixed by default, but to 0.00 since it is a conductor).

3. Project objectives:

The goal of this project was to implement Python code to accurately replicate the results of a research paper published in 1984 into *Physical Review Letters* called “Fractal Dimension of Dielectric Breakdown” by L. Niemeyer, L. Pietronero, and H. J. Wiesmann. In particular, I sought to write Python code to complete 5 main tasks:

a. initialize a 2-dimensional grid and numerically solve the electrostatic Laplace equation upon it in accordance with boundary conditions;
b. use the electrostatic scalar potential field (produced from task 1 in this list) to compute a 2-dimensional probability distribution to quantify the probability that a

certain cell of the grid will become part of the electrical tree at the beginning of the next iteration of the simulation, then use this to stochastically construct the electrical tree;

c. compute the gyration tensor to analyze the mechanical structure of the electrical tree;
d. visualize the evolution of the electrical tree, the electrostatic scalar potential field, the aforementioned probability distribution, and a contour plot of the electrostatic scalar potential field;
e. and finally compute the Hausdorff dimension of the electrical tree fractal.

Each of these 5 goals were accomplished except for the last one regarding the computation of the Hausdorff dimension. Future work will likely occur to accomplish this.

4. Techniques, tools, and outline of your code:

This code uses basic scientific Python libraries to implement the dielectric breakdown model in an easy-to-use manner. The first part of the code imports the necessary libraries, Numpy (for vectorization / array programming) and Matplotlib (for creating plots / animations).

Next, the following source code functions are written out (information about the exact implementation is omitted here for brevity; please see the code for that information):

a. *intialize_grid(N, boundary_val)*: initializes then returns an NxN grid, where each cell is a dictionary with a value for 'potential' and 'is_tree'; convolve_2d(grid), which takes in a 2-dimensional grid as input and uses convolution to solve the Laplace equation upon it before returning the convolved grid;
b. *solve_laplace(grid, threshold)*: iteratively solves the electrostatic Laplace equation on a 2-dimensional grid through the convolve_2d function until convergence;
c. *choose_coordinate(probabilities)*: a helper function that randomly chooses a cell in a 2-dimensional grid in accordance with a corresponding 2-dimensional probability distribution;
d. *grow_tree(grid, eta, frames_prob, i)*: takes in a grid as input, computes the probability distribution for cell tree growth in accordance with the equation derived by the authors of the cited paper, then uses the choose_coordinate function to get a coordinate for cell tree growth before turning the grid with a new cell;
e. *visualize_grid(grid, field)*: plots a grid in accordance with the scalar field it represents;
f. *get_gyration_tensor(grid)*: computes and returns the gyration tensor of the electrical tree;
g. *run_dbm(num_tree_cells, N, threshold, boundary_val, eta)*: the function that the user calls to run the DBM in accordance with user-prescribed parameters – it runs the aforementioned functions to iteratively create the grid, compute values from it, and generation .gif files to animate the evolution of the DBM dynamics.

5. Description of collaboration:

Justin Burzachiello worked alone on this project. However, ChatGPT was used to help write this report and fix occasional bugs which arose during the development of the code. As for writing the report, I, Justin Burzachiello, used ChatGPT by writing a paragraph for each requirement of the report before feeding that information to ChatGPT and telling it to proofread the text and make it more readable.

6. Plots / graphics:

The plots on the following pages were made with the following parameters:

num_tree_cells = 75,
N = 40,
threshold = 1e-5,
boundary_val = 8.

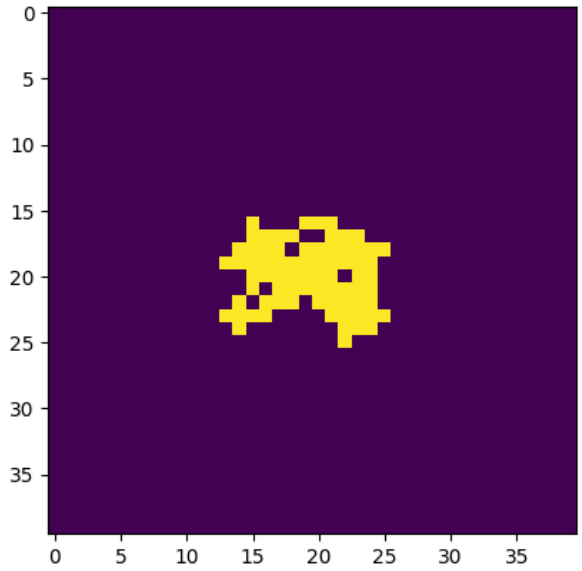
Access the Google Colab notebook via your SBU email at: https://colab.research.google.com/drive/1GkxdDDDZmD8h6b4e7LJzkjCReM9YLMEj?usp=sharing. Click on “Revision history" in the “File” tab to see the history of the notebook during its development.

Eta = 0.0:

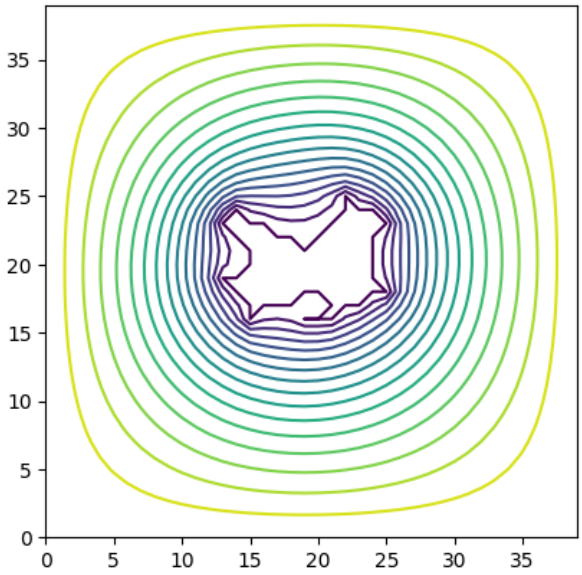
Radius of gyration: 4.12

Acylindricity: 6.50

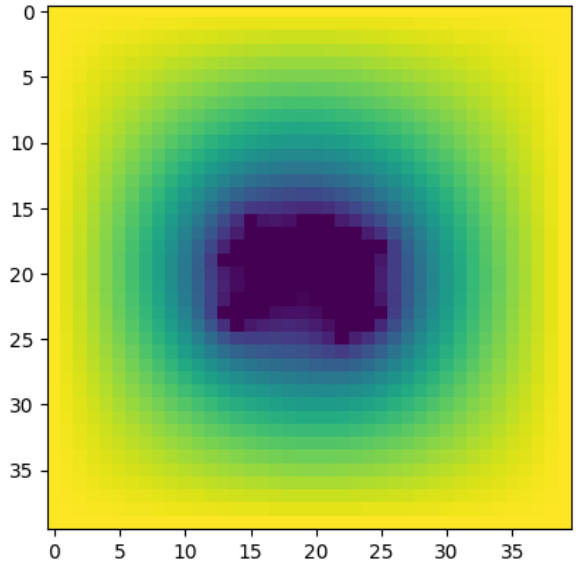
Electrical tree:



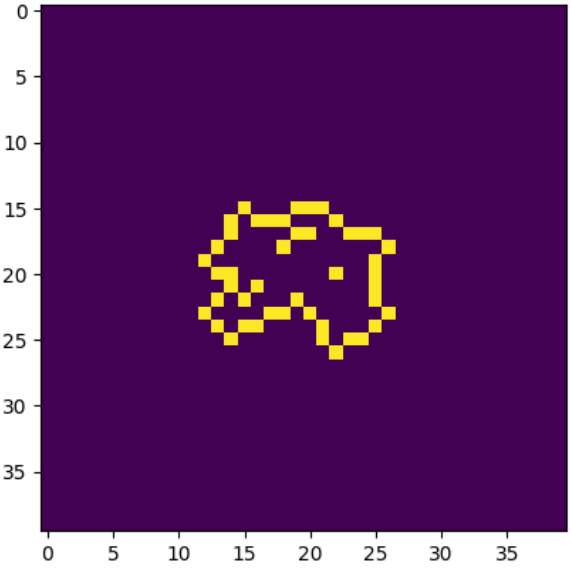
Contour plot:



Potential field:



Probability distribution:

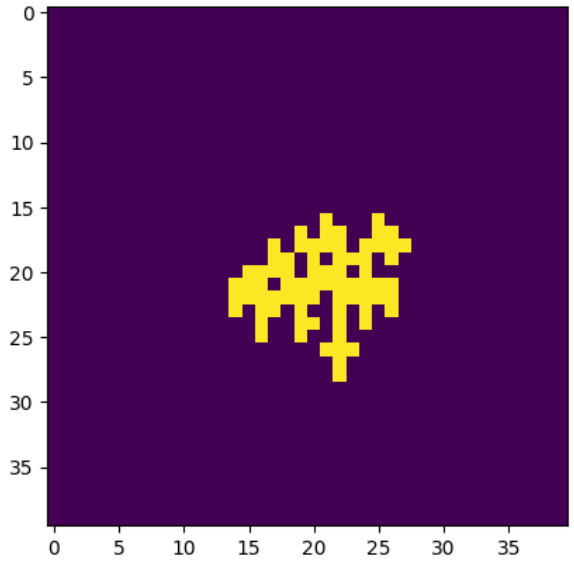


Eta = 0.5:

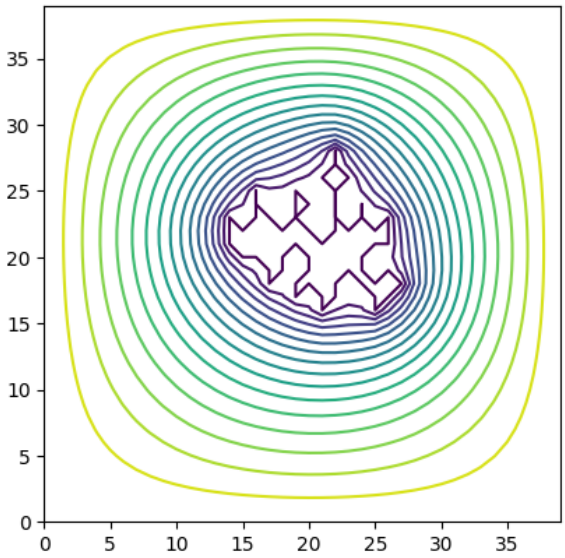
Radius of gyration: 4.39

Acylindricity: 5.78

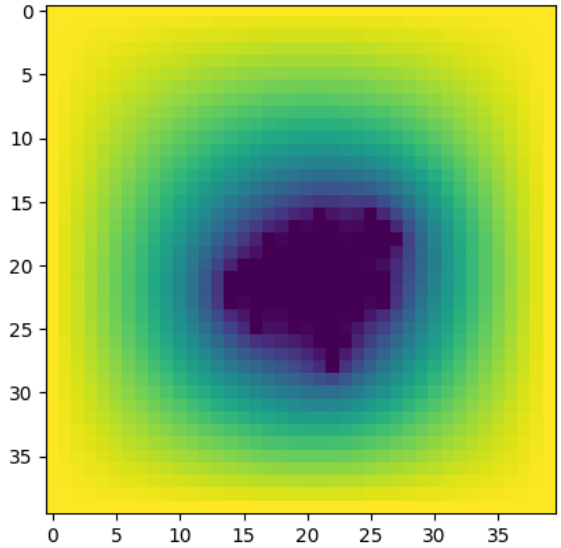
Electrical tree:



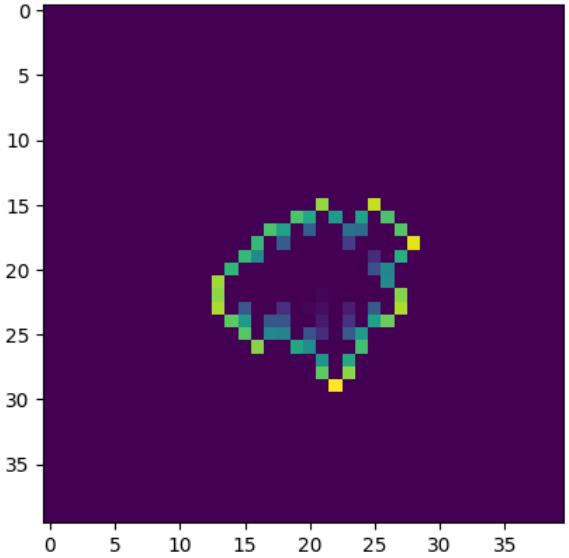
Contour plot:



Potential field:



Probability distribution:

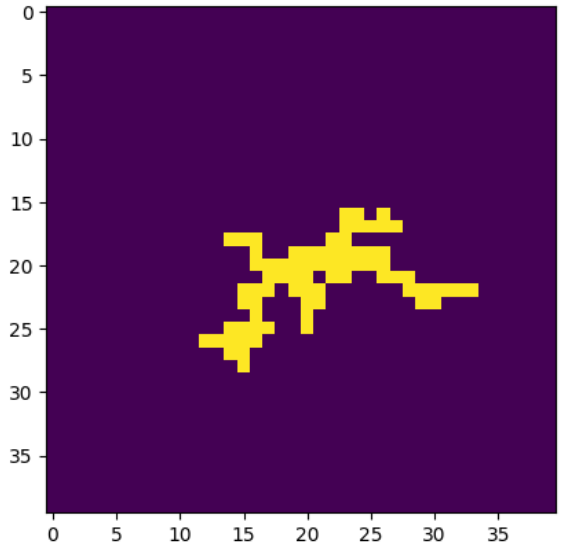


Eta = 1.0

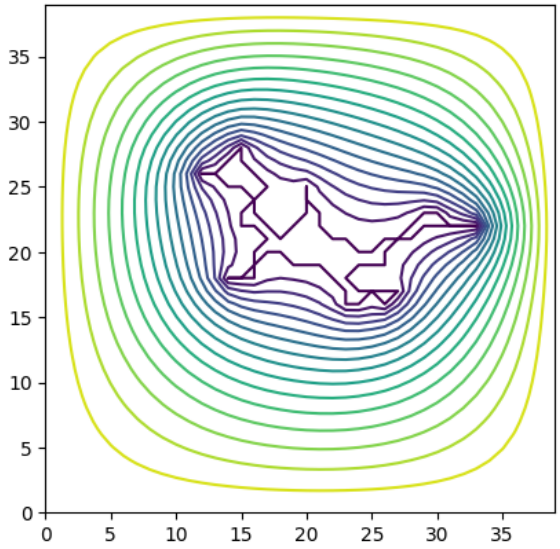
Radius of gyration: 5.87

acylindricity: 21.53

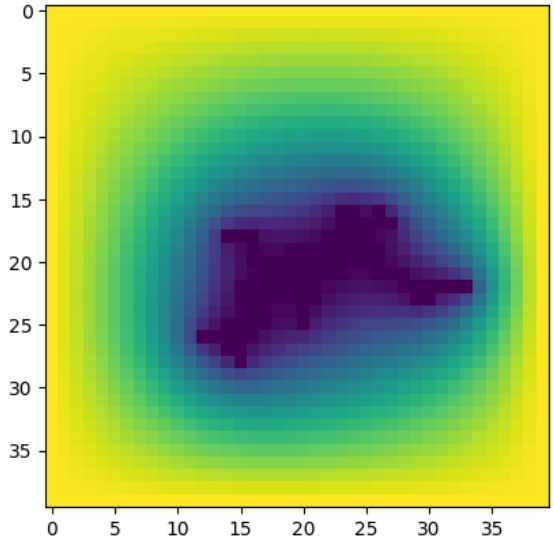
Electrical tree:



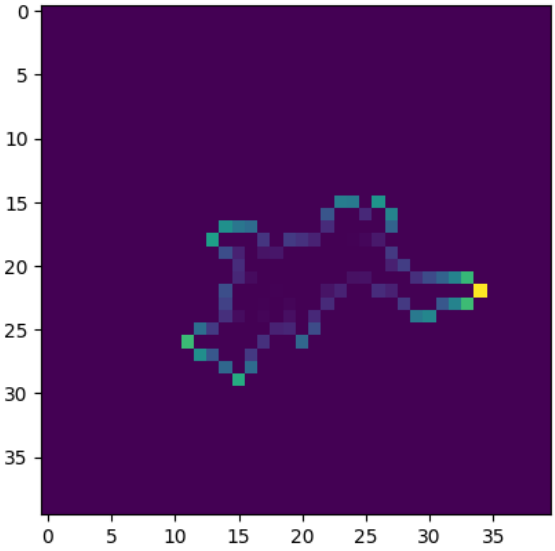
Contour plot:



Potential field:



Probability distribution:

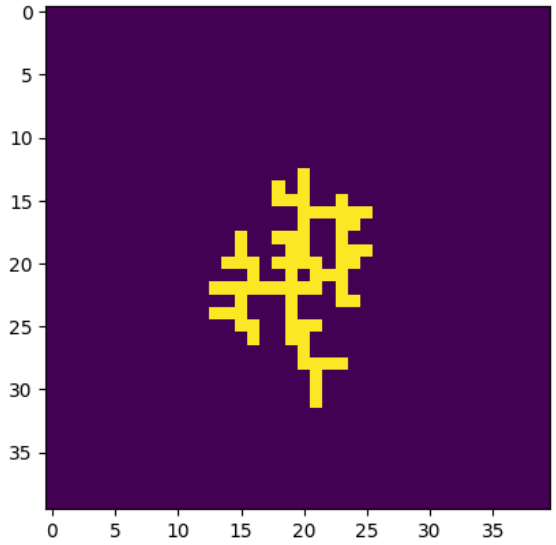


Eta = 2.0:

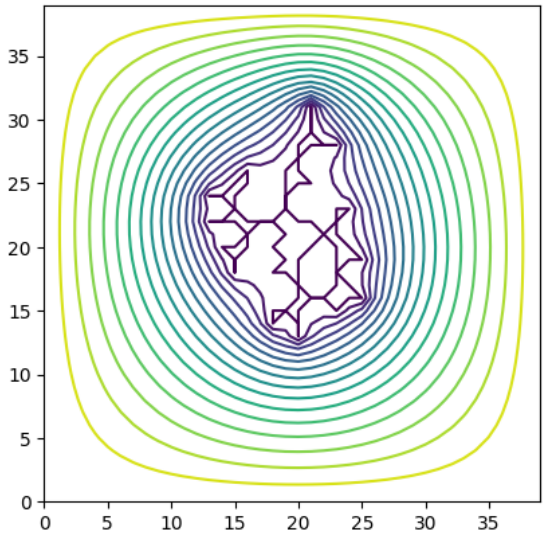
Radius of gyration: 5.16

Acylindricity: 8.32

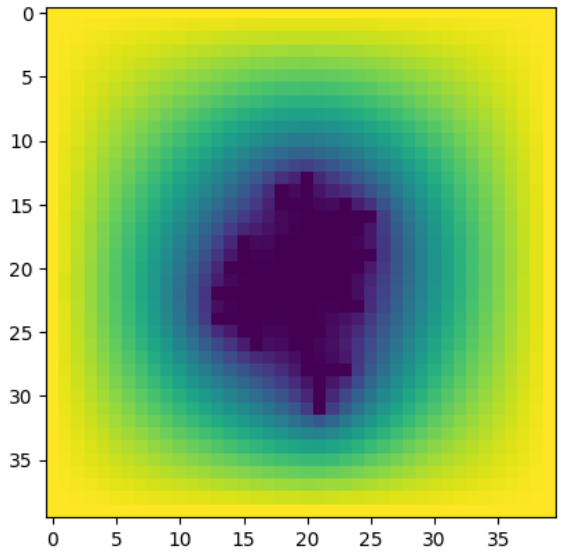
Electrical tree:



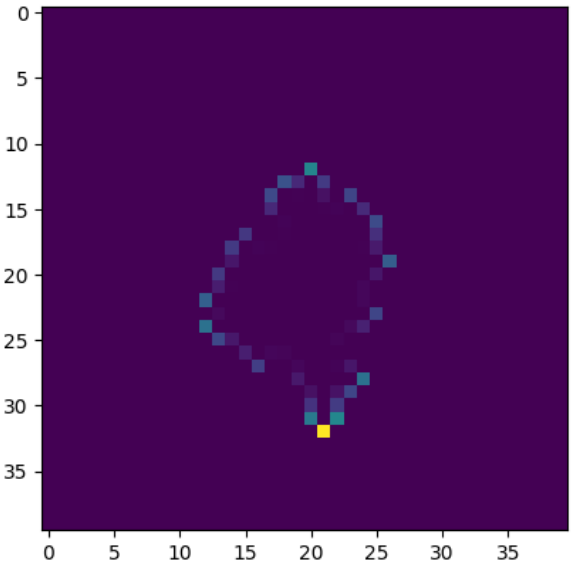
Contour plot:



Potential field:



Probability distribution:



7. Any known shortcomings / bugs:

As of writing this report I do not know of any bugs in the code. However, one shortcoming is that the numerical solution to the electrostatic Laplace equation is inefficient. There are two routes to improving this inefficiency:

a. create a look-up table to store the harmonic functions which solve Laplace's equation for various grid structures and boundary conditions;
b. use optimized methods from the Scipy library to improve the numerical algorithms in this project.

8. Results and conclusions:

The code written in this project correctly generates electrical trees in accordance with the dielectric breakdown model (DBM). In particular, one can animate the construction of an electrical tree structure, the probability distribution governing its evolution, the electrostatic scalar field in the grid, and the contour plot of this electrostatic scalar field (to analyze the corresponding electric field).

By visualizing the probability fields, one can easily see the existence of the well-known Faraday effect and tip effect from electrostatic physics. Importantly, as predicted by the original scientists who developed this computational model, the value of eta in the computation affects the influence of the Faraday effect. Interesting examples of eta are:

a. 0.0, which removes the influence of the potential field's values in the computation of the probability distribution, thus removing the Faraday and tip effects in addition to causing the simulation to replicate the Eden model of cancerous tumor growth;
b. 0.5, which mitigates the effects of potential field values, causing the the Faraday effect and tip effect to have less influence;
c. 1.0, which neither mitigates nor amplifies the influence of potential field values in the computation of the probability distribution, causing the model to replicate the results of the diffusion limited aggregation model;
d. and 2.0, which amplifies the influence of the potential field upon the development of the probability distribution, causing the Faraday and tip effects to have a large influence upon the development of the electrical tree structure.

Finally, quantities obtained from the gyration tensor showed that the acylindricity and radius of gyration of an electrical tree can be quantified to analyze the distribution of its cells relative to the centroid of the structure.

9. References:

I.M Irurzun, P Bergero, V Mola, M.C Cordero, J.L Vicente, & E.E Mola (2002). Dielectric breakdown in solids modeled by DBM and DLA*. Chaos, Solitons & Fractals, 13(6), 1333-1343.*

Niemeyer, L., Pietronero, L., & Wiesmann, H. (1984). Fractal Dimension of Dielectric Breakdown*. Phys. Rev. Lett., 52, 1033–1036.*

Pietronero, L., Wiesmann, H.J. Stochastic model for dielectric breakdown. *J Stat Phys* **36**, 909–916 (1984). https://doi.org/10.1007/BF01012949