Point Source Heat Diffusion Simulated In Python With Customizable Discretized Bodies

Max Mandel

PHYS 25000: Computational Physics, University of Chicago

Autumn 2024

Background

The Heat Equation is a partial differential equation describing how a quantity like heat diffuses through a given body or region. It was first developed in 1822 by Joseph Fourier.

Statement: the function $u : U \times I \to R$ is a solution of the heat equation if

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x_1^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2}$$

Given a cartesian coordinate system useful for computational and engineering contexts, we re-write as:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

Where α is a positive coefficient, called the thermal diffusivity of the material/medium.

As a strictly deterministic PDE, we expect that given fixed boundary conditions, the diffusion of heat (and other quantities) in a body is highly predictable. This is a process that is useful to simulate for an enormous range of engineering applications, including the design of lightbulbs, ovens, slow cookers, space heating products, hot water systems, and even medical procedures and analysis.

The Finite Difference Method is a technique for solving differential equations by approximating derivatives with finite differences. Using Python and freely available libraries for visualization and computation, it's feasible to use the finite difference method to produce simulations of a range of cases of heat diffusion.

Project Statement

In this project, I present a simple computational approach to simulating heat diffusion in two- and three-dimensional bodies, as well as a technique to visualize snapshots of the diffusion over time.

I am also interested in the impact of meshing technique and degree of refinement on simulation results. I present a method for creating and discretizing to user specifications both two- or three-dimensional models, which can be stored and used for simulation.

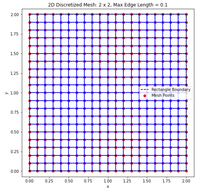
Of course, neither the heat equation, heat transfer simulation, or the finite difference method are new. Heat diffusion simulation is a solved problem, and the impacts of a change in thermal conductivity, density, or dimensions of a medium on simulation results are well-understood. The goal of this project is to demonstrate the feasibility of producing accurate simulation results in a range of non-trivial cases, without the use of third-party software or any Python libraries beyond Matplotlib and Numpy.

I also find that enormous computational power and speed is not necessary for many simulation applications, as this method is still able to generate detailed and accurate results quickly without parallelization or low-level optimization.

Discretization & Meshing

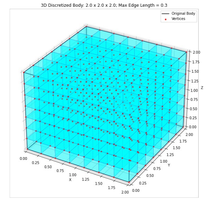
2D

The function to create a discretized mesh/grid takes length, width, and a maximum edge length as inputs, where all are floats. By dividing length and width by maximum edge length and rounding up, we get the number vertices along each side. By dividing length and width by this newly found value, we derive the actual edge lengths in the vertical and horizontal directions – always less than or equal to the maximum edge length provided. Iterating over a double for-loop of vertical and horizontal coordinates, we generate a grid of vertices, and by iterating through the grid (skipping vertices on the rightmost and bottom edges), we generate rectangular cells defined as arrays of four indices into the array of vertices.



3D

The three-dimensional extension generates a field of 3D vertices analogously to the above method. Instead of rectangular cells, we store cuboids as arrays of eight indices into the array of 3D vertices. We store both types of regions as objects {V, C}.



Implementation

For the two-dimensional case, we are interested in the 2D heat equation, which can also be expressed as:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

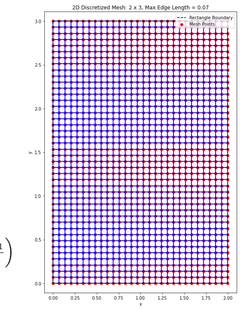
where T(x, y, t) is the temperature at a point (x, y) at time t, and α is the thermal diffusivity of the material.

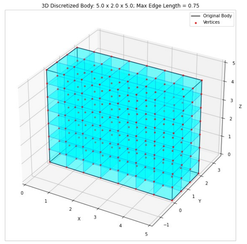
To numerically solve, we employ an explicit time-stepping scheme using the Forward Euler method:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \alpha \Delta t \left(\frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta x^2} + \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta y^2} \right)$$

In 2D, each cell represents a rectangular element with four vertices, and the temperature is defined at each vertex. At each time step (with a step magnitude dt being passed in by the user) we iterate through all cells, through each vertex in each cell, and calculate heat flux between neighbor vertices. We apply a distance weighting, to account for different edge lengths in differently-refined grids.

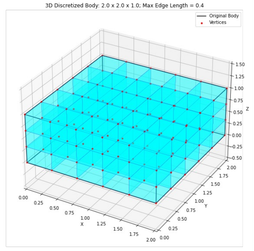
This computation generates an array of temperatures corresponding to vertices for each step. If the step time has been requested by the user, we store the temperature array in an array of snapshots and return. For visualization, cells are colored according to an average of the temperatures at the cell-incident vertices.





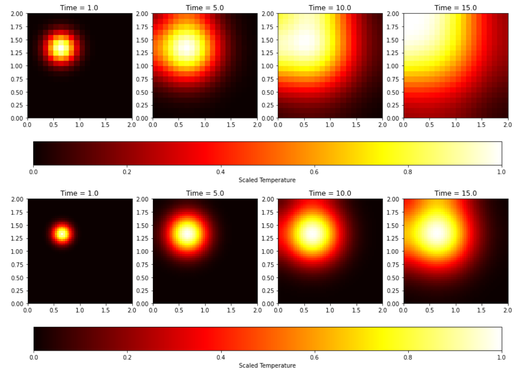
The extension to three-dimensions is once again analogous, as instead of considering four vertices in each cell, we consider 8. As a result, in addition to 3D bodies being generally heavier, containing more vertices (depending on the given dimension), we expect the heat diffusion simulation to take twice as as long, as each cell that is iterated over is considered for 100% more vertices.

For more details about the implementation of this project, see: https://github.com/maxmandel03/MandelPHYS250FinalProject



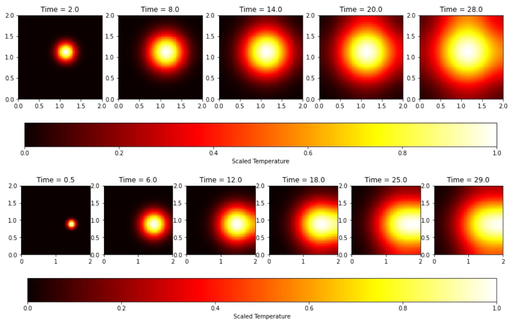
Examples & Results

Impact of improved mesh refinement on simulation results



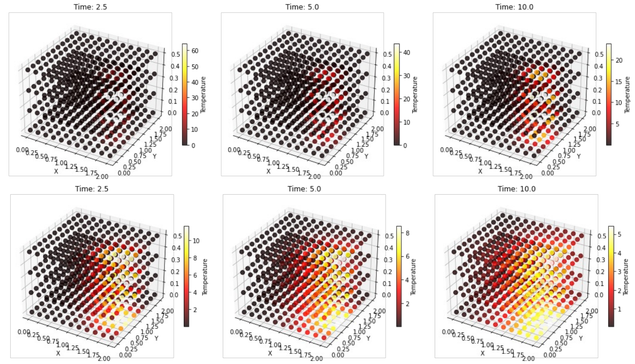
Impact of greater thermal diffusivity constant on simulation results in 3 dimensions (Steel, Aluminum, ~5x larger thermal diffusivity constant)

Demo: longer timescales, additional snapshots, modified heat source, greater thermal diffusivity (copper)

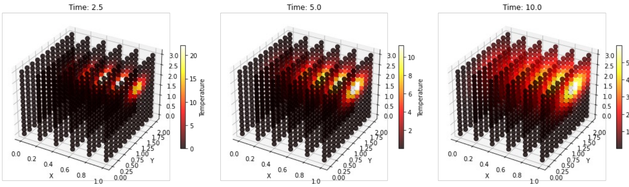


3D Extension

Impact of greater thermal diffusivity constant on simulation results in 3 dimensions (Steel, Copper, ~10x larger thermal diffusivity constant)



Demo: modified thermal diffusivity, body, and discretization



Errors, Limitations, and Further Improvement

While this simple method is surprisingly effective in generating detailed simulations in many cases, there are of course limitations given the narrow scope and simplicity of the approach. The following are improvements I hope to make as I iterate on this project.

1) Support for non-rectangular / rectangular-prism models. Though a CAD modeler would be necessary to support the creation of more complex geometries, straightforward modifications are possible to support spheres, half-moons, triangles, and other simple part shapes. Support for different types of meshing (triangular, quadrilateral) would also be necessary.

2) Simulating multiple heat sources on one medium. This should also be a straightforward extension of the existing functionality and would allow the user to pass in an array of coordinate-temperature pairs as initial conditions. Practically, this expands the range of problems this simulation can be applied to, as the user would be able to make an entire edge (or wall) heated.

3) Experimentation with different computational implementations and simplifications of the heat equation. There exist multiple explicit time-stepping schemes.

Citations and Acknowledgements

Hugo, Ron. "Heat Transfer L10 p1 - Solutions to 2D Heat Equation." YouTube. Uploaded by Ron Hugo, 15 September 2015, https://www.youtube.com/watch?v=lY2QVs7aFMY.

Silvestre, Luis. (2024, May). *Partial Differential Equations* [Lecture Notes]. University of Chicago. https://math.uchicago.edu/~luis/pdenotes.pdf

Kim, Seongjai . (2023, December). *Numerical Methods for Partial Differential Equations* [Lecture Notes]. Mississippi State University. https://skim.math.msstate.edu/LectureNotes/NumerPDEs_Lecture.pdf