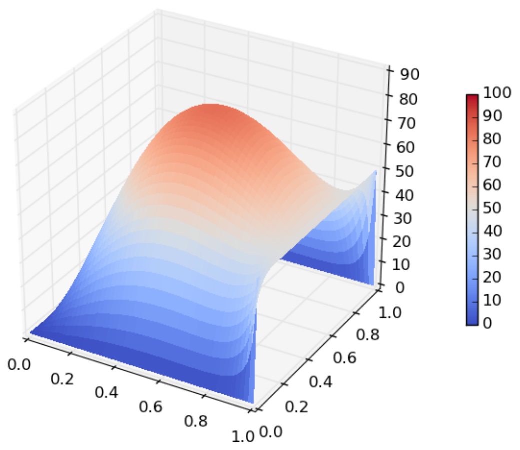
Finite Difference Methods Applied to the Heat Equation in 1D and 2D

Vincent Su

December 14, 2013



Abstract

We turn our attention to the heat equation $\frac{\partial u}{\partial t} = \alpha^2 \nabla^2 u$. This famous equation, also known as a special case of the diffusion equation, has been studied extensively by mathematicians for centuries. There exist a number of approaches for generating analytical solutions, but sometimes these solutions can be expensive to maintain and compute. Thus, for the purposes of modeling physical phenomena, we look for more practical methods of approximating our solution. We begin with applying the Forward Euler, Backward Euler, and Crank-Nicolson schemes to a finite rod in one dimension and then move on to a finite two dimensional plate. Subsequently, we compare the error of the aforementioned methods as a function of the time-step size.

1 Introduction

The heat equation describes the time evolution of a function u (typically denoting temperature) with respect to its spatial concavity. Assuming the system under study has steady boundary conditions, the temperature will converge to a certain profile which is actually a harmonic function (an observation that follows directly from $\frac{\partial u}{\partial t} = 0$). The study of heat flow was the original inspiration for Joseph Fourier's analysis of what is now known as eigenfunction expansion. Despite its roots in the specifics of physics, the heat equation has a wide range of applications in other fields including financial mathematics and probability theory.

Derivation of the heat equation

Let D be a region in $\mathbb{R}^n$. Let x be an element of $\mathbb{R}^n$ and $u(x,t)$ be the temperature at point x, at time t. Let c be the specific heat and ρ be its density. We can calculate the total amount of heat at a time t, $H(t)$, by summing over the region.

$$H(t) = \int_D cpu(x,t)dx$$

We can get the change in heat by applying $\frac{\partial}{\partial t}$ to both sides,

$$\frac{dH}{dt} = \int_D cpu_t(x,t)dx$$

Fourier's Law says that heat flows from hot to cold regions at a rate proportional to the temperature gradient. Additionally, heat can only leave the region D through the boundary. Thus,

$$\frac{dH}{dt} = \int_{\partial D} k\nabla u \cdot n dS$$

By the divergence theorem, we can replace the surface integral with a volume one,

$$\frac{dH}{dt} = \int_D cpu_t(x,t)dx = \int_{\partial D} k\nabla u \cdot n dS = \int_D \nabla \cdot (k\nabla u)dx$$

Equating the two integrands yields

$$c\rho u_t = k\nabla^2 u$$

If c, ρ, k are all constants throughout the region of interest, we arrive at the heat equation in it's familiar form,

$$\frac{\partial u}{\partial t} = \alpha^2 \nabla^2 u$$

Where $\alpha^2 = \frac{k}{c\rho} > 0$ represents the thermal diffusivity of the material.

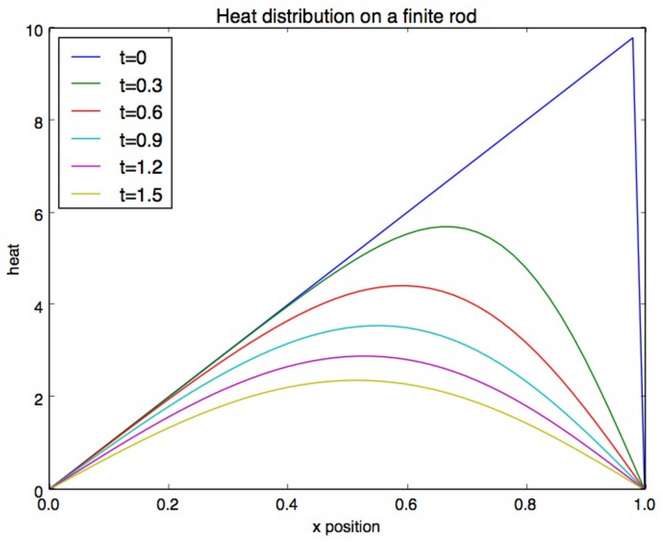


Figure 1: Evolution of $\phi(x) = x$

The Necessity for Computational Methods

For well posed problem, those with homogeneous boundary conditions (Dirichlet, Neumann, or mixed), one can apply the method of eigenfunction expansions. Assuming $u(x,t)$ can be written as the product of a function of t and x separately, the solution to the full system can be expressed as the sum of eigenfunctions, typically the orthogonal $sin(\frac{n\pi x}{L})$ and $cos(\frac{n\pi x}{L})$ multiplied by their appropriate damping factor.

The corresponding amplitudes of each of the individual waves is found by utilizing Fourier's trick, namely, utilizing the orthogonality of the sinusoids to project the initial temperature profile $u(x,0) = \phi(x)$ onto the desired basis vector. For initial conditions which can be expressed as a finite series of sines and cosines, it is extremely easy to calculate the exact value of u for any given x and t. However, for other simple functions like $\phi(x) = 1$, the Fourier series is somewhat troublesome to compute and calculating the analytical solution would require storing and evaluating several terms to maintain accuracy.

For the purposes of modeling the effects of different boundary conditions and initial conditions, we turn to numerical methods to get a sense of how the system evolves with time. Since we can advance the distribution in time as a function of its current state, we eliminate the need to calculate numerous inner products along with their damping factors. Additionally, in many cases, the closed form solution on its own does not provide much intuition for what the heat distribution actually looks like.

Creating a Model

We begin our investigation with a laterally insulated rod of unitary length, diffusivity constant (α^2) of 1, and homogeneous Dirichlet boundary conditions. We discretize our rod into a number of points depending on two factors: how much spatial resolution we want in our solution and computational cost. Though it would be nice to have infinitely fine precision, we cannot afford to wait for days for the simplest of scenarios. Likewise, time is broken up into several smaller intervals. In a similar fashion to the spatial case, the granularity of time steps is affected by computational cost. Ideally, we could pick relatively large time steps to save time in running our simulation. However, excessively large time steps fail to capture accurately the continual changes to the temperature profile. They may also cause instability in the solution for situations with time independent boundary conditions, which for an parabolic PDE should eventually converge to a harmonic function.

Although this pedigogical example may seem overly simple, many problems can be transformed into the form described above. With a change of coordinates, a rod of length L can be transformed to a rod of length 1 by introducing a new spatial variable $\xi = \frac{x}{L}$. If one is interested in modeling the diffusion from an impulse on an infinite or semi infinite rod, then the diffusivity constant can be decreased accordingly so the boundaries of the model do not interfere for a long time. Any time independent Dirichlet boundary conditions can be transformed by subtracting off the line which intersects the two endpoints. Even certain problems with convection or lateral loss of heat can be expressed in terms of solutions to this simplified model.

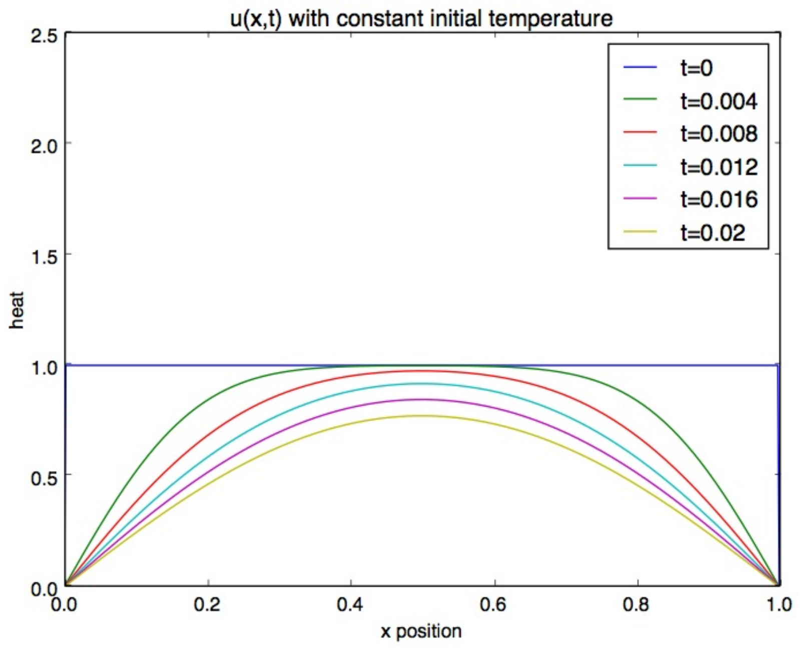
An additional benefit of studying this model is that there exists a number of problems which have easy analytical solutions to compare to. The simplest case being the IC of $\phi(x) = sin(\pi x)$. Using Fourier decomposition and separation of variables, we arrive at a solution of

$$u(x,t) = e^{-\pi^2 t} sin(\pi x).$$

Thus, we calibrated our model by comparing the errors to this base case and afterwards felt confident applying it to a handful of other problems. The one dimensional case may not seem like it gives a particularly large advantage in the space of problems that we're dealing with since computing the Fourier series of a given function is relatively straightforward, but when advancing to two dimensions, the calculations become extremely much more involved. For comparison purposes, the 2D model on the unit square is tested against homogeneous Dirichlet boundary conditions with IC $\phi(x,y) = sin(\pi x)sin(\pi y)$ which has an analytical solution

$$u(x,y,t) = e^{-2\pi^2 t} sin(\pi x)sin(\pi y).$$

It's worth noting that because of the discrete nature of our model, the corners of the sheet may have conflicts in what it's temperature should be. For sake of simplicity we simply choose to have it conform to the temperature specified on the left and right sides ($x = 0, 1$) instead of the top and bottom ($y = 0, 1$). This assumption actually does not affect the results of the simulation because the corner points do not enter in the calculation of the laplacian for interior points.



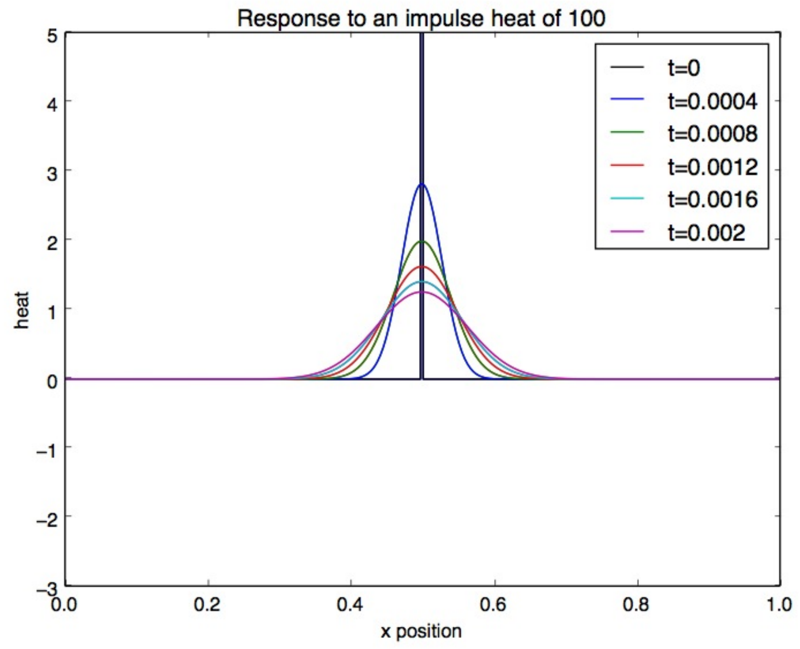


Figure 2: Examples of odd profiles

2 Time-Stepping Schemes in 1D

Forward Euler

Forward Euler is the most intuitive way to evolve a system with time. It's analogous to single variable calculus technique of using linear approximations. Examining the heat equation in 1D once more

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}$$

We can replace the time derivative with a discrete version

$$\frac{\partial u}{\partial t} = \frac{u(x,t_{j+1}) - u(x,t_j)}{\Delta t}$$

where t_j denotes the j-th time step and Δt denotes the length of the time step. And similarly for the second spatial derivative,

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_{i+1},t_j) + u(x_{i-1},t_j) - 2u(x_i,t_j)}{(\Delta x)^2}$$

This scheme is known as fully explicit since each of the $u(x_i, t_j)$ used in the calculation of the future $u(x, t_{j+1})$ are known. This property makes Forward Euler particularly easy to implement. We can further simplify calculations by framing this update scheme as a matrix multiplication problem. For a vectorized $\overrightarrow{u}$ over the spatial domain at a time t_j, we can advance with $\overrightarrow{u_{j+1}} = A\overrightarrow{u_j}$ where A has the following form.

$$A = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ \mu & 1-2\mu & \mu & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \mu & 1-2\mu & \mu \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Courant-Friedrichs-Lewy Condition

The ease with which Forward Euler is calculated comes at a cost. Namely, the main drawback is the limitation of our time step due to instability. If we combine the two derivatives, we arrive at

$$\frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{\Delta t} = \alpha^2 \frac{u(x_{i+1},t_j) + u(x_{i-1},t_j) - 2u(x_i,t_j)}{(\Delta x)^2}$$

Rearranging,

$$u(x_i,t_{j+1}) = \frac{\alpha^2 \Delta t}{(\Delta x)^2}[u(x_{i+1},t_j) + u(x_{i-1},t_j) - 2u(x_i,t_j)] + u(x_i,t_j)$$

Looking at the terms involving $u(x_i, t_j)$, we notice that $\frac{2\alpha^2 \Delta t}{(\Delta x)^2}$ shouldn't exceed 1 or else the solution could explode exponentially. The Courant number (μ) is the ratio of our respective resolutions $\frac{\Delta t}{(\Delta x)^2}$, and the CFL condition states that $\mu \leq \frac{1}{2\alpha^2}$ is needed to ensure stability. Thus, if we double the resolution of our grid, then we must advance with a timestep that is four times smaller.

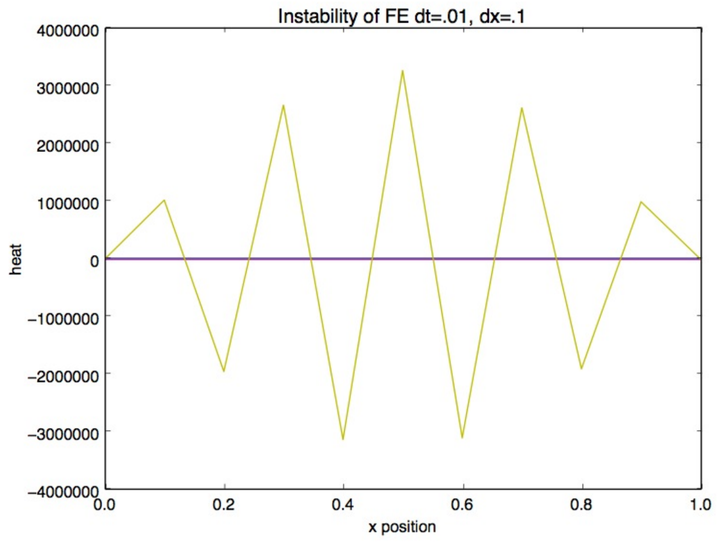


Figure 3: Instability of Forward Euler with μ=1 and $\phi(x) = sin(\pi x)$ up to $t = .1$. Notice the orders of magnitude.

Backward Euler

The Backward Euler approach is very similar to the Forward Euler technique except instead of calculating the laplacian of the current profile, we approximate the time derivative with the laplacian of the future profile. Clearly, we encounter the difficulty of calculating the desired value under the assumption that we can already have it. These class of techniques are called fully implicit, and require more involved calculations, but provide the additional benefit of stability for all time steps. Algebraically, the difference is in our calculation of the second spatial derivative. We use the new definition,

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_{i+1},t_{j+1})+u(x_{i-1},t_{j+1})-2u(x_i,t_{j+1})}{(\Delta x)^2}$$

Taking the vectorized approach, we now have a new system of equations which can be again posed as a matrix algebra problem, this time as $B\overrightarrow{u_{j+1}} = \overrightarrow{u_j}$ where B has the following form.

$$B = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ -\mu & 1+2\mu & -\mu & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -\mu & 1+2\mu & -\mu \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

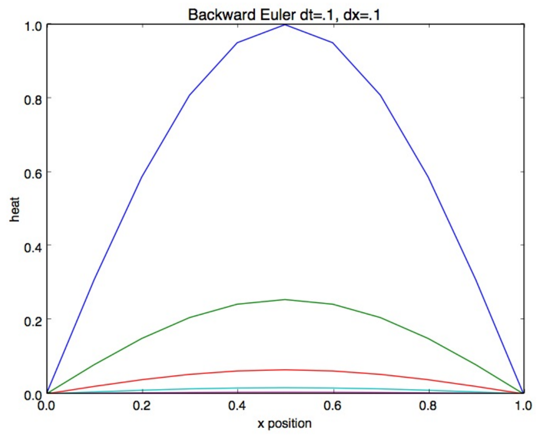


Figure 4: Stability of Backward Euler with $\mu = 10$, $\phi(x) = sin(\pi x)$ up to $t = 1$

Thus, we can obtain the updated values of $\overrightarrow{u}$ by calculating $B^{-1}\overrightarrow{u_j}$. B is also a sparse matrix, so we take advantage of its structure when solving for $\overrightarrow{u_{j+1}}$.

Tridiagonal Matrices

General matrix inversion by a scheme like Gaussian elimination is $O(N^3)$, which is rather impractical for all but a handful of cases. Luckily, there exists an algorithm that operates on matrices with tridiagonal matrices in $O(N)$. Essentially, we manipulate the matrix such that the bandwidth is reduced from 3 to 2, leaving us with an upper triangluar matrix which can easily be solved with back-substitution.

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & c_{n-1} \\ 0 & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_n \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & c_1^* & 0 & 0 & 0 \\ 0 & 1 & c_2^* & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & c_{n-1}^* \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ \vdots \\ \vdots \\ d_n^* \end{bmatrix}$$

Figure 5: General form of Thomas' Algorithm, ending with simple back-substitution

Crank-Nicolson

Crank-Nicolson combines the Forward and Backward Euler schemes by evaluating the laplacian of u at the current and forward time step to result in a semi-implicit scheme. The main advantage is that it combines the stability of the Backward Euler scheme with an improved accuracy with respect to the granularity of the time steps. Algebraically, Crank-Nicolson takes the form,

$$\frac{u(x_i,t_{j+1})-u(x_i,t_j)}{\Delta t} = \frac{\alpha^2}{2}\left[\frac{\partial^2 u(x_i,t_j)}{\partial x^2} + \frac{\partial^2 u(x_i,t_{j+1})}{\partial x^2}\right]$$

Computationally, this equates to solving $\overrightarrow{u_{j+1}} = B^{-1}A\overrightarrow{u_j}$, where A and B, are the same as above except with replacing μ with $\frac{\mu}{2}$.

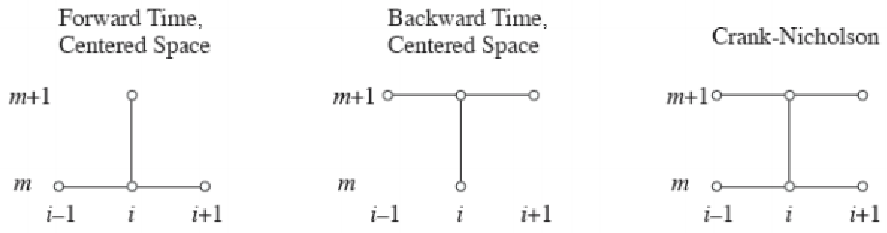


Figure 6: Schematic Diagram of Forward/Backward Euler and Crank-Nicolson

3 Generalizing Schemes to 2D

Interpreting the heat equation in two dimensions merely involves changing the laplacian operator from $\frac{\partial^2}{\partial x^2}$ to $(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$. In addition, our previous model of the rod involving vectorizing u with respect to x now changes to treating u as a two dimensional matrix with x and y.

Forward Euler

For Forward Euler, it is possible to explicitly calculate the laplacian. Assuming our spatial discretizations are equal in both dimensions ($\Delta x = \Delta y$), then we could use a five point stencil,

$$\frac{\partial u}{\partial t} = \frac{u(x_{i-1}, y_j, t_k) + u(x_{i+1}, y_j, t_k) + u(x_i, y_{j-1}, t_k) + u(x_i, y_{j+1}, t_k) - 4u(x_i, y_j, t_k)}{(\Delta s)^2}$$

This approach could also be generalized to a higher dimensional matrix multiplication problem. However, assuming our time steps are small enough, we can alter our laplacian operator to leverage our existing machinery for the one dimensional case without significant error. We break down the net effect of the laplacian into two steps by applying A (as described in the 1D case) first row wise and then column wise (or vice versa).

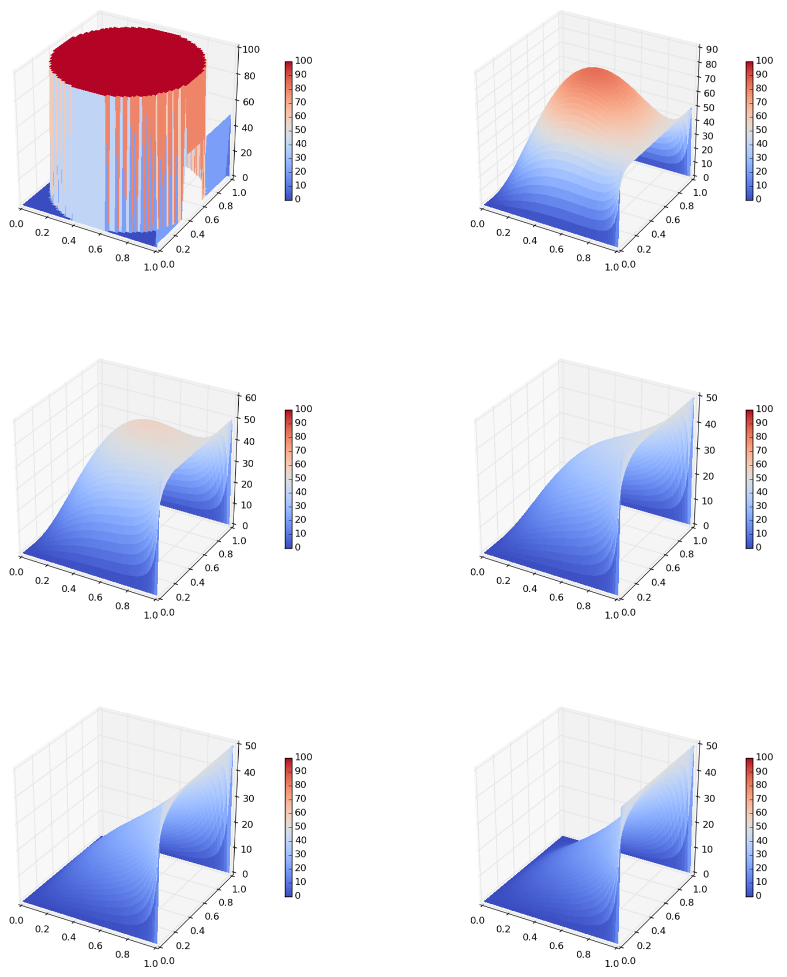


Figure 7: Time evolution of a hot circle on a plate with various BCs

Backward Euler and Crank-Nicolson

In the one dimensional case, we relied on the structure of the matrix representing the laplacian to quickly solve the system of equations. For the implicit schemes in two dimensions, however, we would require a higher dimensional, still sparse, matrix for which inversion becomes extremely more tricky. Thus, we take a similar approach in breaking down our matrix representation of u into slices of rows and then columns and applying the same techniques as the one dimensional case. Although the errors are already typically small enough for the order in which operations are done (row vs. column), the Alternating Direction Implicit (ADI) approach for 2D Crank-Nicolson breaks each timestep into halves where the first update happens with an explicit calculation of the central difference operator in the x direction and an implicit one in the y direction and the second update with x and y switched.

4 Error Analysis

Once the methods have been debugged and verified to have the correct behavior for heat diffusion, we then move on to compare the errors for different methods. First we must address the question of how to measure it. With Dirichlet boundary conditions, heat flows through the perimeter, and gradually all stable solutions converge closer and closer to the true steady state. Thus, it wouldn't be very telling to measure the how the error changes as time advances or to measure the error at times that are too large. For comparison purposes, we model the finite rod with initial profile $\phi(x) = sin(\pi x)$, and calculate the error in the following way: evolve the system for a total time t,at every time interval of $\frac{t}{10}$, add the L2 norm of the rod's modeled temperature versus its analytical solution $u(x,t) = e^{-\pi^2 t} sin(\pi x)$. This method allows us to compare the temporal accuracy of each of the schemes, or more simply put, how our error is affected by our step size.

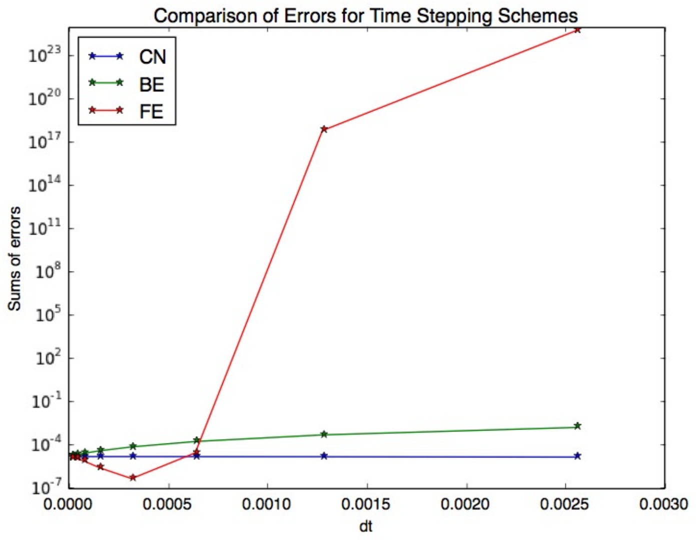


Figure 8: Error plots for the various schemes on the finite rod. $dx = .04$, $n = 10$, $t = .1$

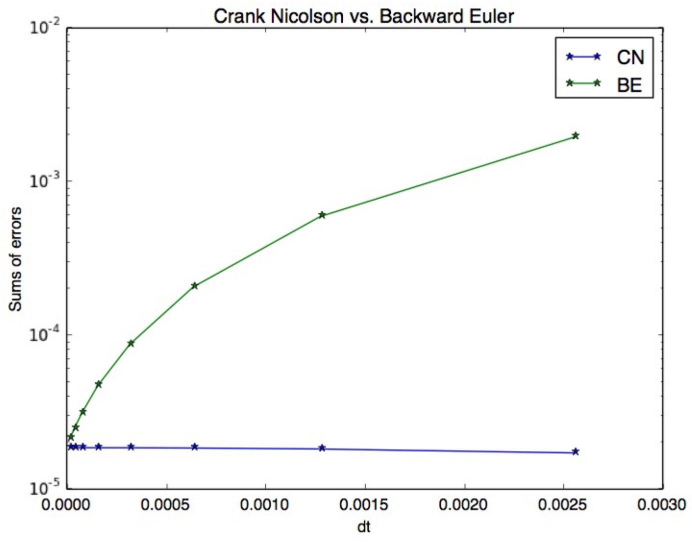


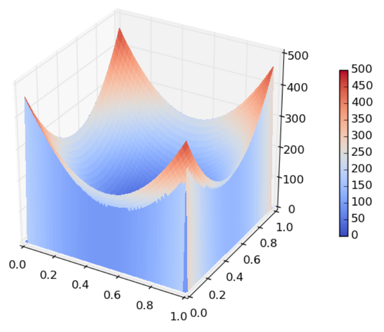
Figure 9: Error comparison for the implicit and semi-implicit methods

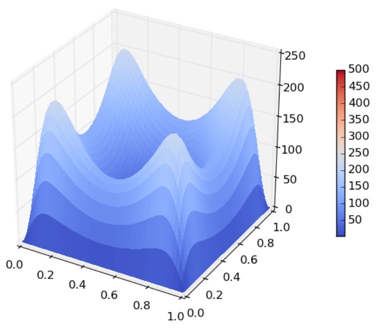
From the data it is quite apparent that the CFL condition comes into play, causing the error for the Forward Euler to explode exponentially. There also appears to be an inexplicable anomaly for very small time steps where the error actually goes down when dt increases. Zooming in on the comparison between Backward Euler and Crank-Nicolson, the two schemes with guaranteed stability, we see that Crank-Nicolson does indeed maintain much better accuracy as dt increases. However, it is very strange that the error with Crank-Nicolson doesn't even seem to increase at all.

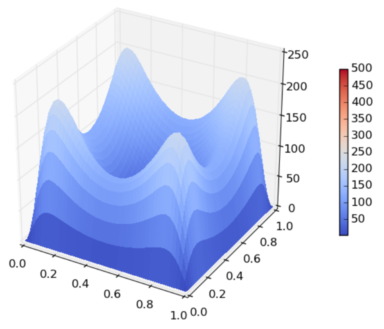
5 Conclusion

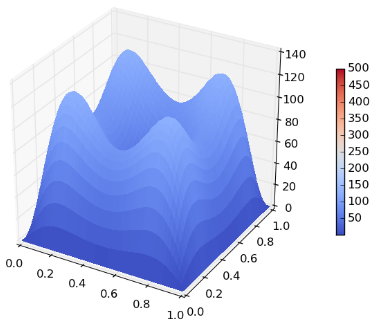
In implementing finite difference methods for the heat equation, we've verified that explicit methods, while much easier to code, have significant shortcomings when it comes to how fast they can run. In one dimension it can be easy to increase the number of mesh points from 100 to 1000, but for a two-dimensional grid, these types of increases are hardly viable. We then turned to the implicit and semi-implicit methods of Backward Euler and Crank-Nicolson. Though they are both harder to implement than Forward Euler, Crank-Nicolson has shown it has a huge advantage over Backward Euler in terms of its accuracy with marginally more effort required to execute.

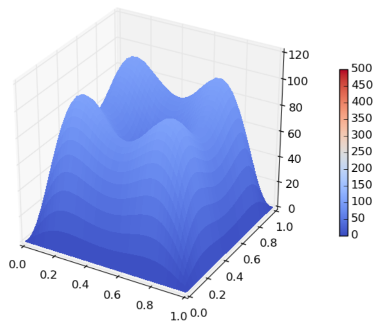
Appendix A. Hot Paraboloid

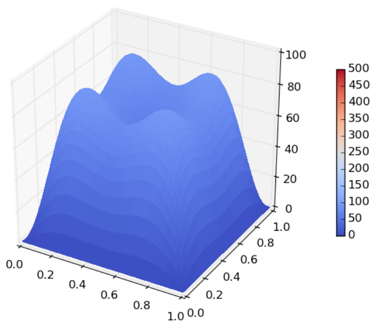


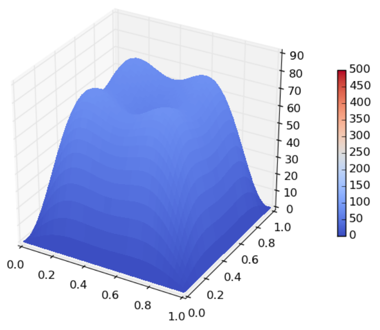


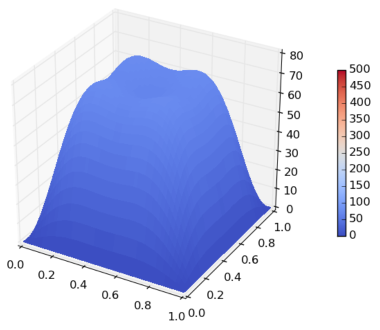


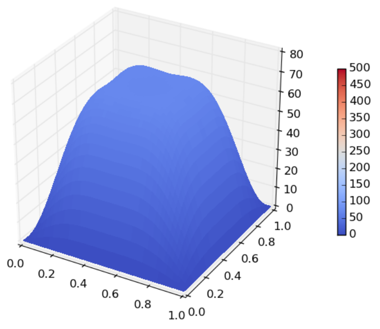


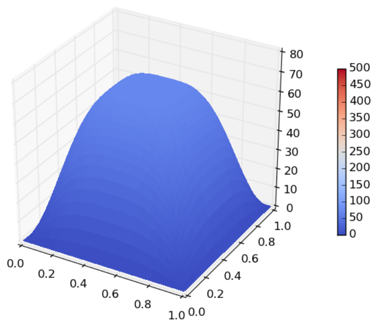





0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
400
450
500


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
400
450
500


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
400
450
500


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
400
450
500

Bibliography

1. Farlow, S., *PDEs for Scientists and Engineers*, Dover Publications, 1982

2. Ouyed, R., *Lectures in Computational Physics*, University of Calgary

3. Recktenwald, G., *Finite − Difference Approximations to the Heat Equation*, March 2011

4. Strikwerda, J. C., *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks, 2004. Ch. 7 p. 172-175

5. Vasy, Andras, *Lectures On PDEs Of Applied Mathematics*, Stanford University, 2012