

COMPUTATIONAL PHYSICS & ASTRONOMY (188B)
Project II: Solving Field Equations

Erik Lamb

February 3, 2016

1 Electrostatic Potential in a P-N Junction

1.1 Introduction

P-N junctions are used ubiquitously throughout modern electronics, particularly in diodes and transistors. A P-N junction is where a p-type and n-type semiconductor meet. P-type and n-type semiconductors are simply silicon crystals that have been “doped” with (usually) boron and phosphorus atoms, respectively [3]. This doping creates an excesses of electrons in the n-type semiconductor and a deficiency or electrons in the p-type semiconductor. The absence of electrons in the p-type semiconductor is called a “hole,” and is treated as its own particle. Therefore, the movement of particles across the P-N junction can be treated through thermodynamics, and when holes and electrons are in thermal equilibrium, the electrostatic potential, $U(x)$, across the junction can be found using Poisson’s Equation.

$$\frac{d^2U}{dx^2} = -\frac{\rho(x)}{\epsilon} \quad (1)$$

Poisson’s equation is a partial differential equation (PDE) and is used to described the potential energy field created by a source of charge, precisely what we are interested in with regards to the P-N junction. Recall that ϵ is the permittivity and $\rho(x)$ is the charge density which depends on how the P-N junction was made. For linearly graded junctions, the charge density is given by

$$\rho(x) = \begin{cases} \epsilon ux/a^3, & -a < x < a \\ 0, & |x| \geq a \end{cases} \quad (2)$$

where u is some value characteristic to the potential and a is the distance from the P-N junction to the edge of the non-neutral region that surrounds the junction. Notice that between $-a$ and a , $\rho(x)$ is an odd function, thus the Poisson equation (1) may only be solved on the interval $0 \leq x \leq a$ otherwise the charge density will vanish when integrated, as expected for a P-N junction. Immediately we find that $U(0) = 0$ since x cannot be less than 0 and because at $x = 0$ the net potential between the two junctions is zero. As previously mentioned, equation (1) applies for a P-N junction in thermal equilibrium, thus $dU(a)/dx = 0$ since the charge of the semiconductor is neutral beyond the non-neutral region $x \leq a$. Our boundary conditions for Poisson’s equation are therefore $U(0) = 0$ and $dU(a)/dx$. After normalizing x with a and U with u , we have following simplified form of Poisson’s equation the boundary conditions necessary to solve,

$$\begin{aligned} \frac{d^2U}{dx^2} &= -x, \\ U(0) &= 0, \\ \frac{dU(1)}{dx} &= 0 \end{aligned} \quad (3)$$

where $0 \leq x \leq 1$. Fortunately, the above equation is simple enough to solve for $U(x)$ and has the following simple analytical solution,

$$U(x) = \frac{x(1 - x^2/3)}{2} \quad (4)$$

Equation (4) provides an easy way to check our solutions after we integrate equation (3) with numerical methods described in the next section.

1.2 Numerical Methods

For smooth functions on a finite interval $[a, b]$, a forward finite difference $\Delta f_i = f_{i+1} - f_i$ can be used to approximate the first derivative. Known as the forward difference derivative [4], the discrete approximation for the first derivative is given by

$$D_+ f_i = \frac{\Delta f_i}{h} = \frac{f_{i+1} - f_i}{h} \quad (5)$$

for a discrete interval h . The above result is useful for solving the Poisson equation for the electrostatic potential, as well as the magnetic diffusion equation in the next section, since both PDEs must be discretized spatially as well as temporally in order to solve numerically. We can discretize space just as we did for time by letting $x_i = i\Delta x$ for a small spatial step Δx and $i = 0, 1, 2, \dots$. Furthermore, the Poisson PDE in equation (3) can be discretized, giving us,

$$\frac{d^2 U(x_i)}{dx^2} = -x_i \quad (6)$$

As shown in chapter 2 of [4], successive use of the forward, backward, and central difference derivative operators, combined with a lot of messy work, yields a result that allows us to approximate derivatives up to an arbitrary order using finite differences. In particular, we can approximate the second derivative with

$$f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \mathcal{O}(h^2) \quad (7)$$

Therefore the second derivative of the discretized electrostatic potential in (6) can be approximated as

$$\frac{d^2 U(x_i)}{dx^2} = \frac{U(x_{i+1}) - 2U(x_i) + U(x_{i-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) \quad (8)$$

Notice that the error of the second order finite difference approximation goes like $(O)(\Delta x)$, which we will explore later. Plugging (8) back into (6) and finding an expression of the $U(x_i)$ terms in terms of x_i yields the following set of equations

$$\begin{aligned} -2U_i + U_{i+1} &= -\Delta x^2 x_i, & i = 1, \\ U_{i-1} - 2U_i + U_{i+1} &= -\Delta x^2 x_i, & 1 < i < M, \\ 2U_{i-1} - 2U_i &= -\Delta x^2 x_i, & i = M \end{aligned} \quad (9)$$

Note that the factor of 2 on the U_{i-1} term in the $i = M$ equation in (9) is the result of the boundary condition $\frac{dU(1)}{dx} = 0$. The boundary condition $U(0) = 0$ is not considered in (9) and must be added later. Of greater interest, however, is the fact that equation (9) represents a system of linear equations of the form $A\mathbf{x} = \mathbf{b}$, where A is an $M \times M$, tridiagonal matrix. The vector \mathbf{x} is of length M and represents the solutions to the Poisson PDE; $\mathbf{x} = \mathbf{U} = (U_1, U_2, \dots, U_M)$. Finally, vector \mathbf{b} is also of length M and it typically contains initial or known values, that allow us to solve for \mathbf{x} , in this case $\mathbf{b} = (-\Delta x^2 x_i, -\Delta x^2 x_i, \dots, \Delta x^2 x_i)$. Thus we can rewrite equation (9) in the following matrix form,

$$\begin{pmatrix} -2 & 1 & \cdots & 0 \\ 1 & -2 & 1 & \vdots \\ & 1 & \ddots & \ddots \\ \vdots & \ddots & & 1 \\ 0 & \cdots & 2 & -2 \end{pmatrix} \begin{pmatrix} U_0 \\ U_1 \\ \vdots \\ U_{M-1} \\ U_M \end{pmatrix} = \begin{pmatrix} -\Delta x^2 x_1 \\ -\Delta x^2 x_2 \\ \vdots \\ -\Delta x^2 x_{M-1} \\ -\Delta x^2 x_M \end{pmatrix} \quad (10)$$

Notice the matrix A is tridiagonal, meaning it has one diagonal and a super- and a subdiagonal. Equation (10) can now be solved for the vector \mathbf{U} via Gaussian Elimination. Using Gaussian elimination repeatedly to solve for \mathbf{U} generates an algorithm known as the *Tridiagonal Matrix Algorithm*, for obvious reasons. This method of numerical integration goes like $\mathcal{O}(M)$ for M number of equations. This is more computationally efficient than other methods of solving linear systems of equations, such as finding the inverse matrix A which go like $\mathcal{O}(M^3)$. The computational speed of the tridiagonal matrix algorithm will be discussed more in depth in the next section.

Before moving on to the solutions of Poisson's equation for an electrostatic potential in a P-N junction, the value M requires more explanation. M is the total number of equations in the linear system of equations in (9), and is therefore the dimension of the vectors and matrix in (10). M is much like the maximum number of steps N for the simple integrators used in the harmonic oscillator problem. However, there is one key difference between M and a maximum number of steps N ; M is not independent of the spatial step, or the time step as we will see in the next section, whereas N is independent. For problems like equation (3), the PDEs only apply for a certain range of x values, in this case $0 \leq x \leq 1$, so M and Δx must be chosen such that $x_M = M\Delta x = 1$. Thus $\Delta x = 1/M$. In the next section we will find another slightly modified value for M .

1.3 Results

1.3.1 Solution Plot and Discussion

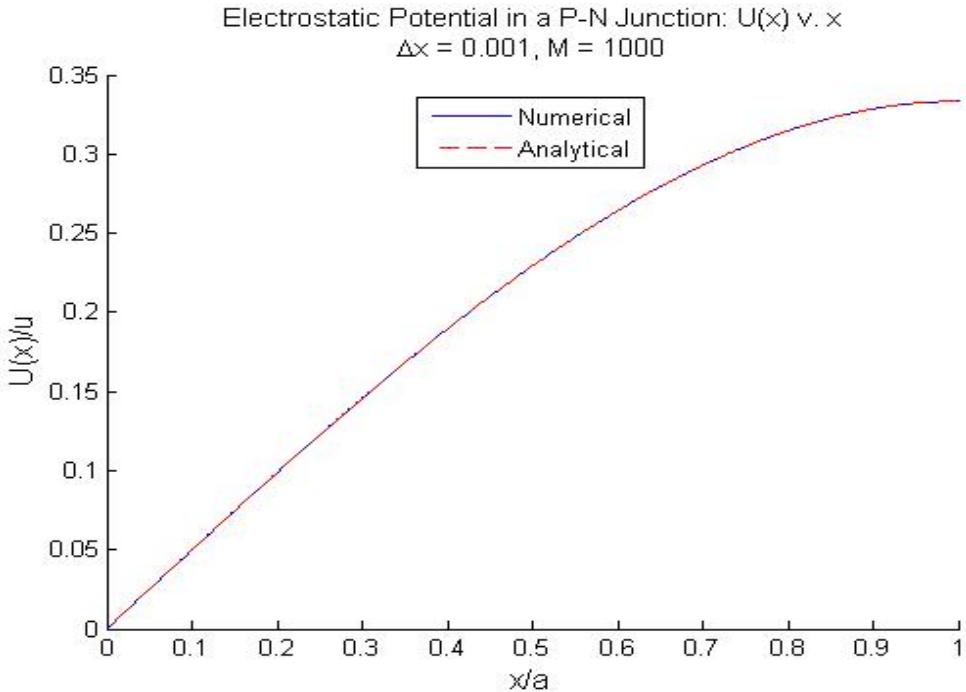


Figure 1: The numerical solution was computed for large number of equations $M = 1000$ and a correspondingly small spatial step $\Delta x = 0.001$.

Tridiagonal Matrix Algorithms, written in a variety of programming languages, are readily available online. A provided algorithm written in Python was used to solve for \mathbf{U}_i in (10), thereby solving the one-dimensional Poisson PDE for an electrostatic potential in a P-N junction. A solution generated by applying the numerical methods discussed previously and by using the tridiagonal matrix algorithm is shown above in Fig. 1.

From the plot we see that the numerical solution is in excellent agreement with the analytical solution, equation (4), by virtue of the fact that they are indistinguishable. However, we must also compare our numerical solution with the Poisson equation and the boundary conditions used. Recall from equation (3) that the second derivative of the electrostatic potential is negative, $d^2U/dx^2 = -x$. We can easily check that our numerical solution has a negative second derivative since it is concave down. Also recall the Poisson equation was said to be asymmetric about $x = 0$, and although we cannot say with absolute certainty given the x interval, the numerical solution may very well be asymmetric about $x = 0$ since $U(x \rightarrow 0) \approx x$ and approaches zero as x approaches zero. Since the function $U(x \rightarrow 0) \approx x$ is asymmetric, $U(x)$ as a whole may be asymmetric. This brings us to the boundary conditions. As we just mentioned, $U(0) = 0$ thereby confirming the first boundary condition. At $x = 1$, it certainly appears that derivative of the numerical solution is zero, thereby confirming the second boundary condition $dU(1)/dx = 0$. The purpose of this discussion was to explore other means of checking our numerical solutions without relying on an explicit analytical solution. Using the initial conditions and common sense, we can confirm our numerical solution is correct and have no idea what $U(x)$ actually looks like.

Knowing that our solution is correct, we can now discuss what Fig. 1, as well as $U(x)$ beyond the scope of Fig. 1, might mean physically. Notice the solution $U(x)$ shown in Fig. 1 is clearly positive on the interval $0 < x < 1$. The electrostatic potential is related to an electric field by $\mathbf{E} = -\nabla U(x)$, thus when electrostatic potential is positive, the source of the potential must also be positive. This would indicate that our solution to the Poisson equation for the electrostatic potential in a P-N junction only represents the potential of a p-type semiconductor. This is because the absence of electrons in the p-type semiconductor, called “holes,” are considered positive charges thereby generating a positive potential. The n-type semiconductor must then occupy the space for $-1 < x < 0$.

Because the n-type semiconductor has an excess of electrons, the electrostatic potential generated by it must be negative and equal in magnitude to the potential generated by the p-type semiconductor, since both semiconductors are in equilibrium across the P-N junction. This confirms our suspicions that $U(x)$ is asymmetric as well as indicates the existence of an electrostatic potential or voltage difference across the P-N junction. The maximum potential value in the p-type semiconductor is around $U_{max} = 0.3$, according to Fig. 1, thus the minimum potential value must be located in the n-type semiconductor and equal to $U_{min} = -0.3$. We find the difference in the electrostatic potential across the P-N junction as being $\Delta U = U_{max} - U_{min} = 0.6$, in normalized units of V/u . The electrostatic potential difference across the P-N junction is the same thing as the voltage difference across the junction. Thus, the voltage difference across the P-N junction is $\Delta V = 0.6$ in V/u , which incidentally is the same voltage difference across a common P-N junction diode or across the base-emitter junction of a NPN transistor [1], further confirming the correctness of our solution.

1.3.2 Effects of Varying M

This result in Fig. 1 is for a large M value of 1000. Therefore, 1000 discrete, linear equations with a time step of $\Delta x = 1/M = 0.001$ were solved for the vector \mathbf{U} using the tridiagonal matrix algorithm in order to generate the solution above. Naturally, the large number of equations used and the correspondingly small spatial step generated a solution in excellent agreement with the analytical solution, as seen in Fig. 1. Using more equations and a therefore a smaller spatial step, however, costs computing time. But since the tridiagonal matrix algorithm is linear in M , as mentioned earlier, the computational time for such a large M is not dramatically longer than with

smaller M values. This topic will be discussed further at the end of section 2.

For now we turn our attention to very small values of M and their effects on our solution $U(x)$, having already examined very large values of M in Fig. 1. Below are three different plots with $M = 2$, $M = 5$, and $M = 10$.

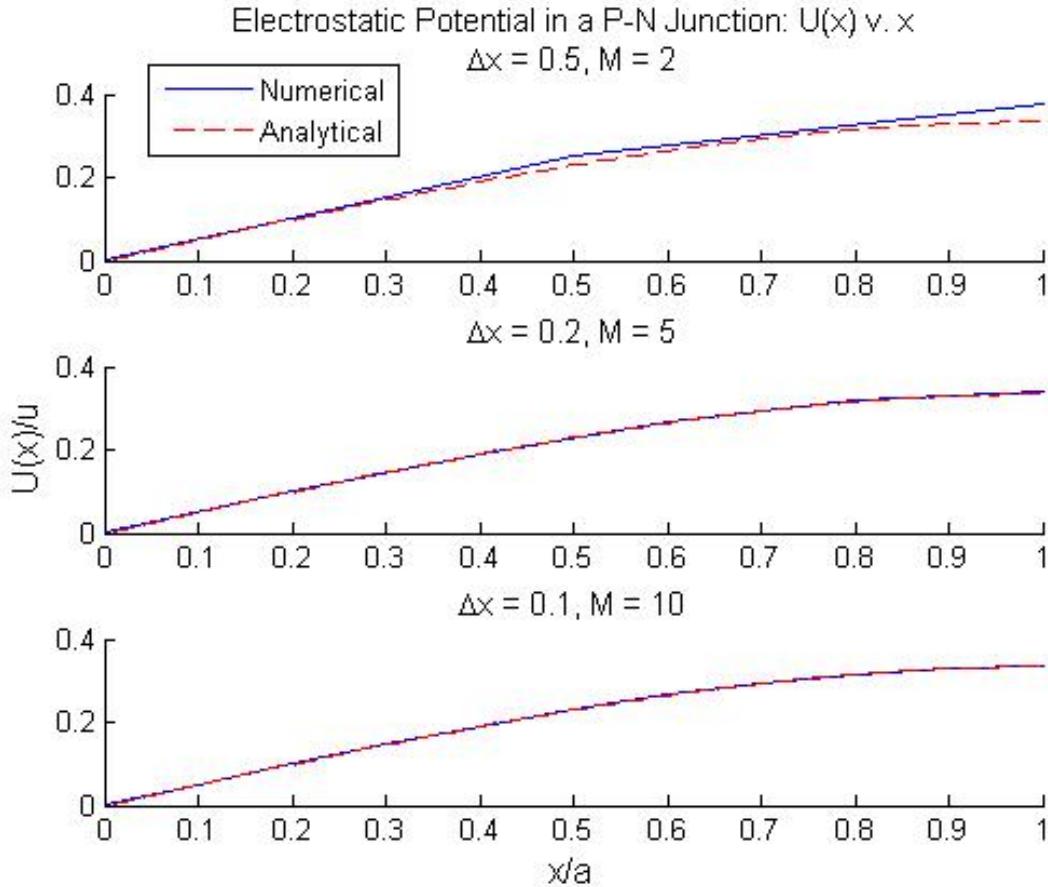


Figure 2: Three plots of the numerical solution to the Poisson equation using three small M values. Notice the relation Δx and M , $\Delta x = 1/M$. Also notice how using more equations M results in better approximations.

From the plots above we can clearly see using larger values of M , and therefore more equations in our tridiagonal matrix algorithm, results in better and numerical solutions. Even a value for M as low as $M = 10$ results in a pretty decent approximation. For $M = 5$, the numerical solution becomes noticeably choppier, especially as x approaches one. $M = 2$ is the smallest possible number of equations used due to the indexing of vectors in the Python code as well as the structure of the algorithm itself. This is a good thing, since the result for $M = 2$ clearly diverges from the analytical solution around $x = 0.5$ and as x approaches one. While the solutions for $M = 10$ and $M = 5$ appear to obey the boundary conditions in (3), $U(0) = 0$ and $dU(1)/dx = 0$, the solution for $M = 2$ does not since $dU(1)/dx$ is clearly non-zero and in fact diverges from the analytical solution.

There are a few reasons using less equations to numerically solve the Poisson equation (3) results in poor approximations. The first, explained in Project 1, is due to round-off error. As has been stated multiple times, M and Δx are not independent, in fact $\Delta x = 1/M$. This means that less equations we use, the larger our spatial step Δx is. Recall that multiplication by a relatively large number results in round-off error and $\Delta x = 0.5$ may just be large enough as computations are made using Δx in the tridiagonal algorithm.

The choppiness mentioned earlier is due to the second order finite difference approximation

made of the Poisson equation. Recall that finite difference methods approximate a function or its derivative by using such approximations as the forward difference derivative, equation (5), which essentially find successive tangent lines to considered function. In order for the forward difference derivative $D + f_i$ to be considered a good approximation $f_{i+1} - f_i$ must be considered small, meaning the spatial step Δx must be small. If Δx is too large then $f_{i+1} - f_i$ will be large and the tangent lines found in approximating f will become noticeable. This is most noticeable in the plot for $M = 2$, which appears to be comprised of two straight lines, as well as in the choppiness of the plot for $M = 5$. The failure of the solution for $M = 2$ to obey the boundary condition $dU(1)/dx = 0$ can also be attributed to using the finite difference approximations. Since $M = 2$ and $\Delta x = 0.5$, the fine tangent lines needed to approximate $U(x)$ cannot be made, and thus the gradual decline of the slope dU/dx cannot be shown.

Finally, the finite approximation method used to solve the Poisson PDE, which was equation (8), is second order. This means its truncation error goes like $\mathcal{O}(\Delta x^2)$, so the larger Δx is due to using small number of equations M , the larger the truncation error and the worse the result. A confirmation that the finite difference approximation used is second order accurate is given in the next section.

1.3.3 Optional: Finite-Difference Approximation Accuracy

Recall the second-order finite difference approximation used to numerically solve Poisson's PDE as being

$$\frac{d^2U(x_i)}{dx^2} = \frac{U(x_{i+1}) - 2U(x_i) + U(x_{i-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) \quad (11)$$

The second-order specification refers to the truncation error $\mathcal{O}(\Delta x^2)$, and we say that this finite difference approximation is “second order accurate.” The truncation error can be found by taking the absolute value of the difference numerical solution, $U(x)^*$, and the analytical solution $U(x)$. Explicitly, the error, also known as the absolute error, is $|U(x) - U(x)^*|$. Below is plot of the absolute error as a function of x for three different values of Δx .

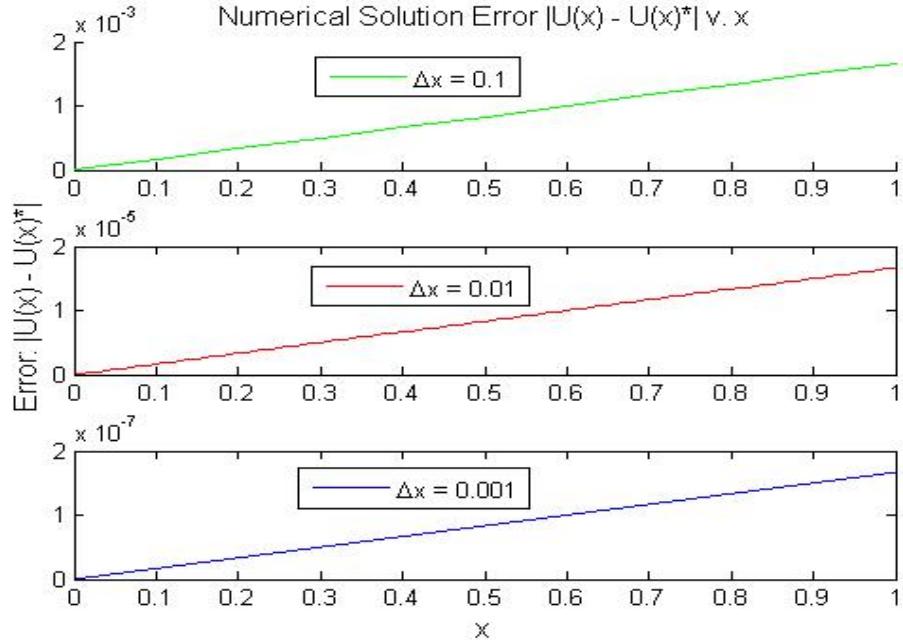


Figure 3: Three plots of the absolute error $|U(x) - U(x)^*|$ for different spatial steps Δx , each of which are off by a factor of 10. Notice that y-axis scales differ by a factor of 100.

Notice that the plots above are ordered largest Δx to smallest Δx from top to bottom and that each successive Δx differs by a factor of 0.1 from the last. At first glance the three plots appear identical. However, a close look at the error axis scales reveals that the scale of each successive plot differs by a factor of 0.01 from the last, from top to bottom. Because each successive Δx differs by a factor of 0.1, the fact that $(0.1)^2 = 0.01$, which is the difference in scales, seems to confirm that the second order finite difference method is in fact second order accurate.

We now will directly show that the second order finite difference method is in fact “second order accurate,” that its truncation error goes like $\mathcal{O}(\Delta x^2)$. First, we chose two spatial steps Δx_1 and Δx_2 such that $\Delta x_2 = \Delta x_1/2$. We then solved Poisson’s equation with the usual second order finite difference method for the two different spatial steps Δx_1 and Δx_2 . Then, we found the absolute error $|U(x) - U(x)^*|$ as a function of x for both spatial steps and plotted both errors curves on the same plot. Finally, we found the slopes of each line and compared them, expecting the slope of the line for Δx_2 to be smaller than Δx_1 by a factor of 4. We expect this because for second order accuracy, $\Delta x_2^2 = (\Delta x_1/2)^2 = \Delta x_1^2/4$. The results of this procedure are shown below, From the plot

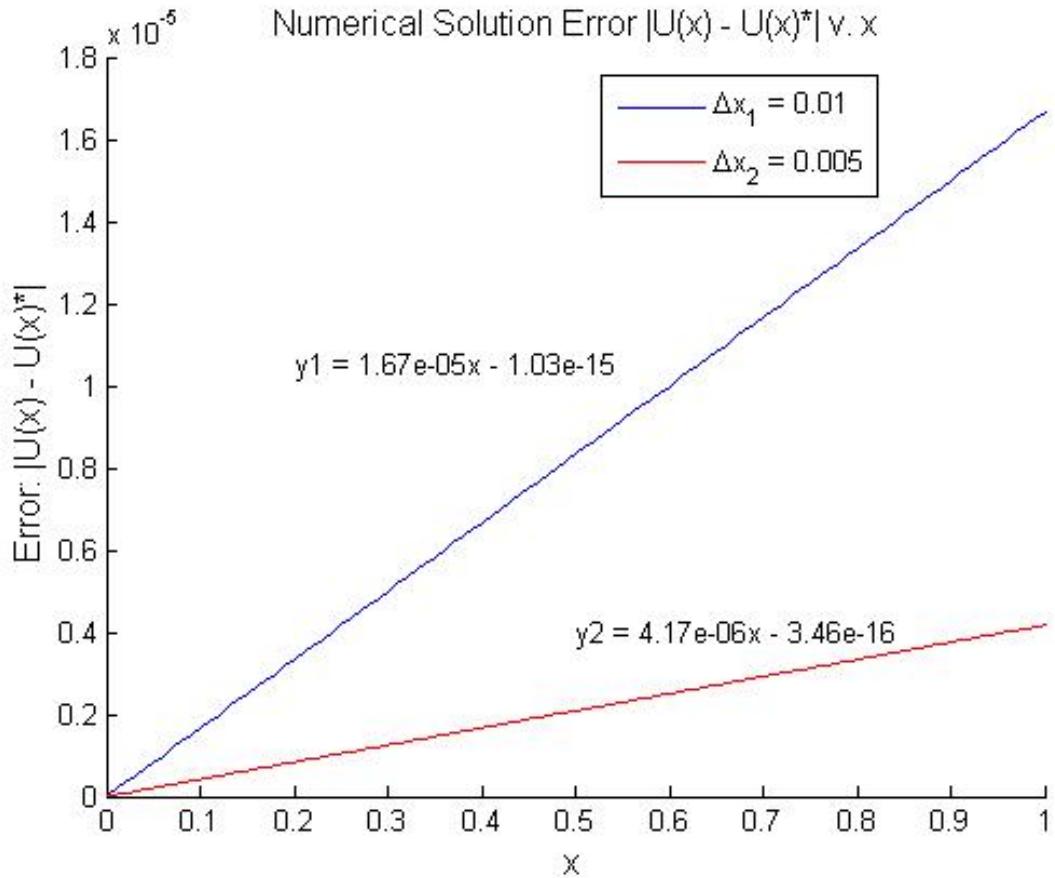


Figure 4: The absolute error $|U(x) - U(x)^*|$ for two spatial steps Δx_1 and Δx_2 related by $\Delta x_2 = \Delta x_1/2$.

above, we find that the slope of the error line for Δx_1 is equal to $m_1 = 1.67 \times 10^{-5}$, while slope of the error line for Δx_2 is equal to $m_2 = 4.17 \times 10^{-6}$. Comparing these we found that $\frac{m_1}{m_2} = 4.00$. Thus, solving for m_2 gives us

$$m_2 = \frac{m_1}{4} \quad (12)$$

Thus confirming the second-order finite difference approximation is “second order accurate.”

2 Magnetic Diffusion

2.1 Introduction

The next physical problem we wish to solve is more complicated and requires the following visual aid,

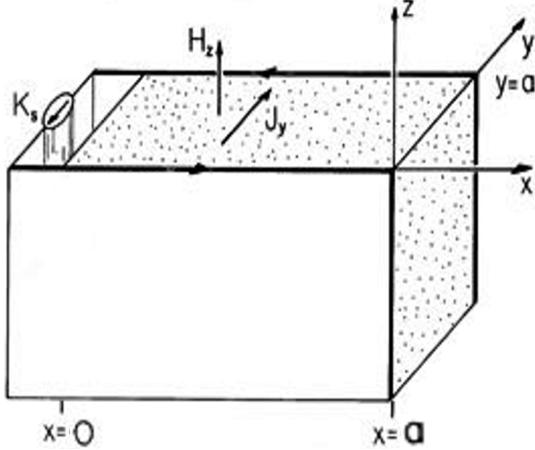


Figure 5: This image was taken from [2]. It illustrates a block of conducting material between two electrode plates with a current source indicated by K in the diagram. The system along the z -axis is assumed to be infinitely long, like a solenoid. This physical system is described by the diffusion PDE.

In the image above, a nonideal conducting material with conductivity σ and permeability μ is placed between two electrode plates. A current source, indicated in the diagram by K , generates a current that goes through the electrode plate closest to us, goes through the conducting material, and then goes through the electrode plate on the opposite side and back to the source at K . For a current flowing in a counter-clockwise direction around the system, a $\mathbf{H} = H\hat{z}$ field will be generated in the z -direction and the current density $\mathbf{J} = J\hat{y}$ will move in the y -direction through the conducting material. Note that for simplicity, the z -component of the entire system, material and plates included, is assumed to be infinitely long so the system can be handled much like a solenoid.

The problem posed to us, therefore, is “when the current source is rapidly turned on at $t = 0$, how will the H field and current density J respond?” with the opinion that both H and J will not be uniform until after some transient time because of the nonideal properties of the conductive material placed between the two electrodes. This physical problem can be described by the diffusion equation, a PDE, which is of the form,

$$\kappa \frac{\partial^2 H}{\partial x^2} = \frac{\partial H}{\partial t} \quad (13)$$

for a magnetic diffusivity of $\kappa = 1/\sigma\mu$. As with Poisson’s equations, we need appropriate boundary conditions $x = 0$ and $x = a$ and an initial condition for $t = 0$ in order to determine H . For the initial condition, no current is turned on before $t = 0$, therefore no magnetic field could be present at $H(x, t = 0) = 0$. Because of the infinitely long z -axis for the infinitely long solenoid approximation, we can assume that the H -field does not extend past the conducting block at $x = a$ and thus $H(x = a, t) = 0$ gives us one boundary condition. Since the closed circuit extends beyond the $x = 0$ edge of the conducting block, back to the current source, Ampere’s Law is required to find the other boundary condition at $x = 0$. The result is $H(x = 0, t) = K$, for an electric current surface density K , which flows in the $y - z$ plane.

Having found our boundary and initial conditions, we simplify our problem by normalizing, as usual. Distance is normalized by a , time is normalized by a^2/κ , and H is normalized by K . Our set

of equations to be solved are therefore,

$$\begin{aligned} \frac{\partial^2 H}{\partial x^2} &= \frac{\partial H}{\partial t}, \\ H(x = 0, t) &= 1, & \text{for } 0 < x < 1 \\ H(x = 1, t) &= 0, \\ H(x, t = 0) &= 0 \end{aligned} \tag{14}$$

This is the form of the diffusion equation we will solve numerically. This problem is different from the past in that we do not have an explicit analytical solution against which to check our numerical solution. Instead, we have an asymptotic solution for $H(x, t)$ for times $t \rightarrow \infty$. Designated as $H_\infty(x)$, the asymptotic solution is,

$$H_\infty(x) = 1 - x \tag{15}$$

Therefore, for long times the magnetic field $H(x)$ should uniformly decrease as we move away from the current source at $x = 0$.

2.2 Numerical Methods

In the previous section, the PDE to solved was only dependent on one variable, x . We now consider a PDE dependent on the two variables x and t , both of which must discretized using the usual $x_i = i\Delta x$ and $t_n = n\Delta t$. The discretized diffusion equation is thus,

$$\begin{aligned} \frac{\partial^2 H(x_i, t_n)}{\partial x^2} &= \frac{\partial H(x_i, t_n)}{\partial t}, \\ H(x_0, t) &= 1, & \text{for } 0 < x < 1 \\ H(x_{M+1}, t) &= 0, \\ H(x_i, t_0) &= 0 \end{aligned} \tag{16}$$

for M equations. We now wish to numerically integrate equation (16) by finding terms of $H(x_i, t_{n+1})$ in terms of $H(x_i, t_n)$. Once again, we utilize finite-difference approximations for both sides of the diffusion equation. An approximation for $\partial H(x_i, t_{n+1/2})/\partial t$, where $t_{n+1/2} = t_n + \frac{\Delta t}{2}$, is given by

$$\frac{\partial H(x_i, t_{n+1/2})}{\partial t} \approx \frac{H(x_i, t_{n+1}) - H(x_i, t_n)}{\Delta t} \tag{17}$$

which is implicit, since it requires knowledge of the t_{n+1} term. $\partial^2 H(x_i, t_{n+1/2})/\partial x^2$ can be approximated using a time averaged version of the second order finite difference approximation, given by

$$\begin{aligned} \frac{\partial^2 H(x_i, t_{n+1/2})}{\partial x^2} &= \frac{1}{2\Delta x} (H(x_{i-1}, t_n) - 2H(x_i, t_n) + H(x_{i+1}, t_n) + \\ &\quad H(x_{i-1}, t_{n+1}) - 2H(x_i, t_{n+1}) + H(x_{i+1}, t_{n+1})) \end{aligned} \tag{18}$$

If we plug equations (17) and (18) into the diffusion equation and group the $n+1$ terms on one side and the known n terms on the other, we get the following system of linear equations,

$$\begin{aligned} 2(1+r)H_i^{n+1} - rH_{i+1}^{n+1} &= 2(1-r)H_i^n + rH_{i+1}^n + 2r, & i = 1 \\ -rH_{i-1}^{n+1} + 2(1+r)H_i^{n+1} - rH_{i+1}^{n+1} &= rH_{i-1}^n + 2(1-r)H_i^n + rH_{i+1}^n, & 1 < i < M \\ -rH_{i-1}^{n+1} + 2(1+r)H_i^{n+1} &= rH_{i-1}^n + 2(1-r)H_i^n, & i = M \end{aligned} \tag{19}$$

where r is given by,

$$r = \frac{\Delta t}{\Delta x^2} \tag{20}$$

and $H_i^{n+1} = H(x_i, t_{n+1})$. Note that the boundary conditions $H(x_0, t_i) = 1$ and $H(x_{M+1}, t_i) = 0$ are not included in equation (19) and must be added later. Because we must add two more equations later, to keep our solution on the interval $0 < x < 1$ our spatial step must be $\Delta x = 1/(M + 2)$.

Notice that at all terms on the right hand side of equation (19) are known at $t = 0$, or t_0 , by the initial condition $H(x_i, t_0) = 0$. Therefore, at $n = 0$ we have the set of equations,

$$\begin{aligned} 2(1+r)H_i^1 - rH_{i+1}^1 &= 2r, & i &= 1 \\ -rH_{i-1}^1 + 2(1+r)H_i^1 - rH_{i+1}^1 &= 0 & 1 < i < M \\ -rH_{i-1}^1 + 2(1+r)H_i^1 &= 0 & i &= M \end{aligned} \quad (21)$$

which looks exactly like the form of equation (9), a linear system of equations that we solved using the tridiagonal matrix algorithm. Indeed equation (21) is of the form $A\mathbf{x} = \mathbf{b}$ and we can use the tridiagonal matrix algorithm to solve (21). After finding all H_i^1 terms, we can find the $n = 2$ terms, then use that to find all $n = 3$ terms and so on until we have found all $n + 1$ terms in equation (19) for a given $n = 0$ initial condition. In this way we generate a matrix of data for H_i^n . After including the boundary conditions, H_i^n appears as such,

$$H_i^n = \begin{pmatrix} H_0^0 & H_0^1 & H_0^2 & \cdots & H_0^N \\ H_1^0 & H_1^1 & H_1^2 & \cdots & \vdots \\ H_2^0 & H_2^1 & H_2^2 & \ddots & \\ \vdots & \vdots & \ddots & \ddots & H_M^N \\ H_{M+1}^0 & \cdots & & H_{M+1}^{N-1} & H_{M+1}^N \end{pmatrix} \quad (22)$$

Which is an $(M + 2) \times (N + 1)$ matrix where $M + 2$ is the total number of spatial steps taken plus the two boundary conditions and $N + 1$ is the total number of time steps taken, including the $i = 0$ and $n = 0$ values. Each n th column of the H_i^n matrix in (22) represents the values of $H(x_i, t_n)$ at a particular time t_n . A n th column is then used to calculate the right hand side of equation (19) for time t_n , which finds the values for $(n + 1)$ th column of equation (22), which are the values $H(x_i, t_{n+1})$, and so on until maximum steps are reached. In this ways we solve the diffusion equation (16).

Before we continue to the results, the constant r requires some explanation. Known as the CFL condition, r influences the accuracy, and for explicit methods the stability, of our numerical methods. Typically there is some maximum value C_{max} , dependent on the numerical method used, which for values $r \leq C_{max}$ the method is stable and accurate and above which the method becomes unstable and inaccurate. Because the method employed in solving the diffusion equation, the Crank-Nicolson method, is implicit we do not need to worry about stability, bur r value of $r \leq 1$ is required for accuracy. Because our numerical methods will use $r = 1$ and our spatial step is $\Delta x = 1/(M + 2)$, then using the relation r in equation (2) our time step must be $\Delta t = 1/(M + 2)^2$. We now move on to the results of our numerical methods.

2.3 Results

2.3.1 $H(x,t)$ Solution Plot

The solution plot for the diffusion equation, and boundary and initial conditions, given in equation (14) for $r \approx 1$ and $M = 100$ is shown below in Fig. 6. From the solution plot, we see that the values for $H(x, t)$ do approach the asymptotic solution $H_\infty(x) = 1 - x$ as solutions for longer times are given. Also notable is the fact that both boundary conditions and the initial conditions are behaved. We see that for all solutions at all times, $H(x = 0, t) = 1$ and $H(x = 1, t) = 0$, the very boundary conditions given in equation (14). Also, for shorter times we see that the initial condition

$H(x, t = 0) = 0$ is behaved, since the magnetic field appears to be approach zero for all x as $t \rightarrow 0$, that is except for $x = 0$ where the boundary condition is applied.

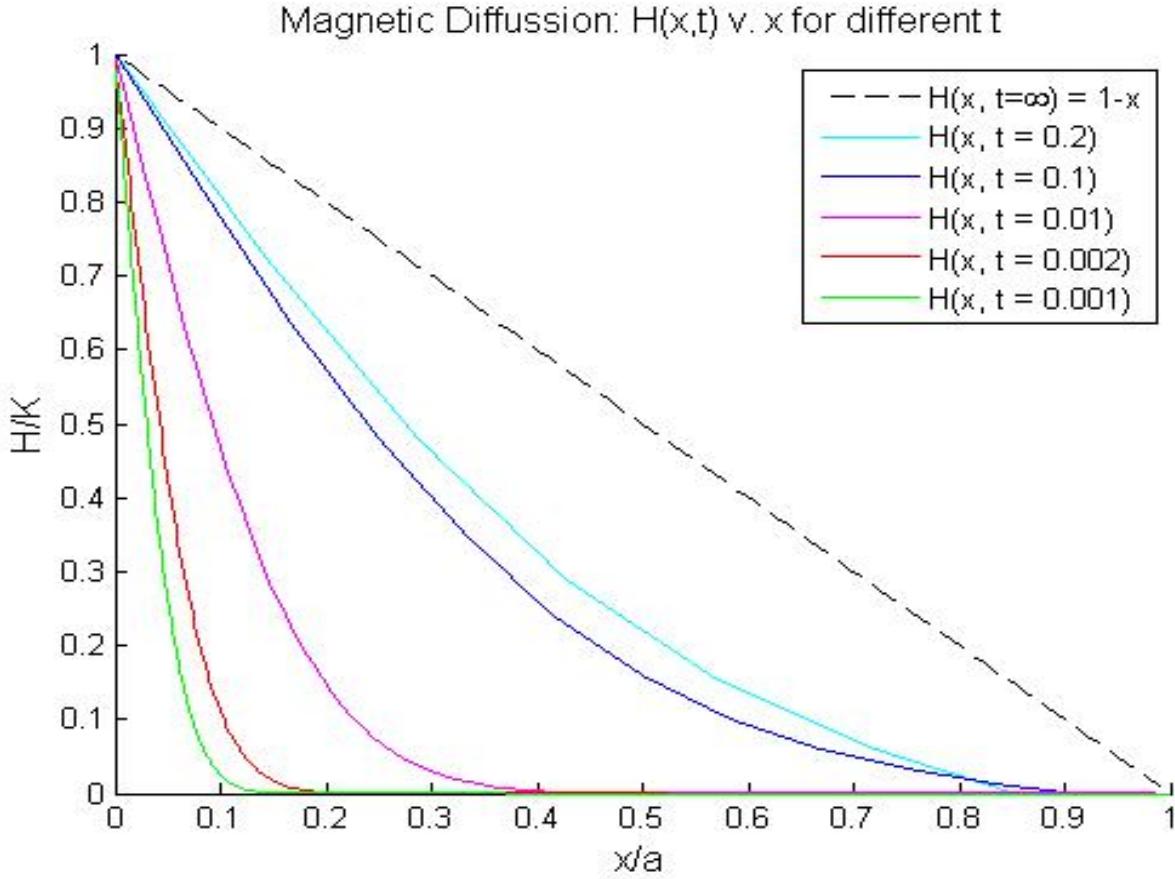


Figure 6: Solutions to the diffusion equation at various times for $r \approx 1$ and $M = 100$. Notice how as times progress the curves approach the asymptotic solution $H_\infty(x) = 1 - x$, as desired.

Everything seems to suggest that our solution to the diffusion equation (14) is correct. Therefore, we can now discuss what is occurring physically. Recall that the conducting material stuck between the two electrodes is nonideal. It was predicted that, because of the nonideal conductivity of the material, the H -field would not be uniform for long times. Our result confirms our prediction since, as mentioned earlier, for short time the H -field is zero everywhere except for at $x = 0$. However as time progresses, it appears as though the magnetic field is “charging up” for all x .

This physical effect can be explained by the fact that our system of two, separated electrode plates is practically a capacitor while our conducting block with nonideal conductivity, meaning that block has resistivity, acts as a resistor. The circuit shown in Fig. 5, then, is basically a resistor-capacitor (RC) circuit. RC circuits have an time constant τ , measured in seconds, of the resistance R times the capacitance C , or $\tau = RC$ [1]. The time constant is the time required for the capacitor to charge through the resistor and once the capacitor is fully charged the current moving through the circuit becomes uniform. Because the magnetic field is generated by the current moving through the system, this explains why it takes time for the magnetic field H to grow and become uniform everywhere; it depends on the RC-time of the circuit.

2.3.2 Optional: Electric Current Density

The electric current density J is related to the magnetic field H by

$$J = -\frac{\partial H}{\partial x} \quad (23)$$

We can calculate J with our discrete data found for H by using the finite difference approximation to estimate the first spatial derivative of H . That approximation is given by

$$f'_{i+1/2} = \frac{f_{i+1} - f_i}{h} \quad (24)$$

for a spatial step h . Since $h = \Delta x$, the expression for the current density is given by

$$J_n = \frac{H(x_i, t_n) - H(x_{i+1}, t_n)}{\Delta x} \quad (25)$$

at a particular time t_n . We can then loop through various columns of the matrix in (22) and, using equation (25), calculate the current density at different times. Our results for $r = 1$ and $M = 100$ are shown in the figure below,

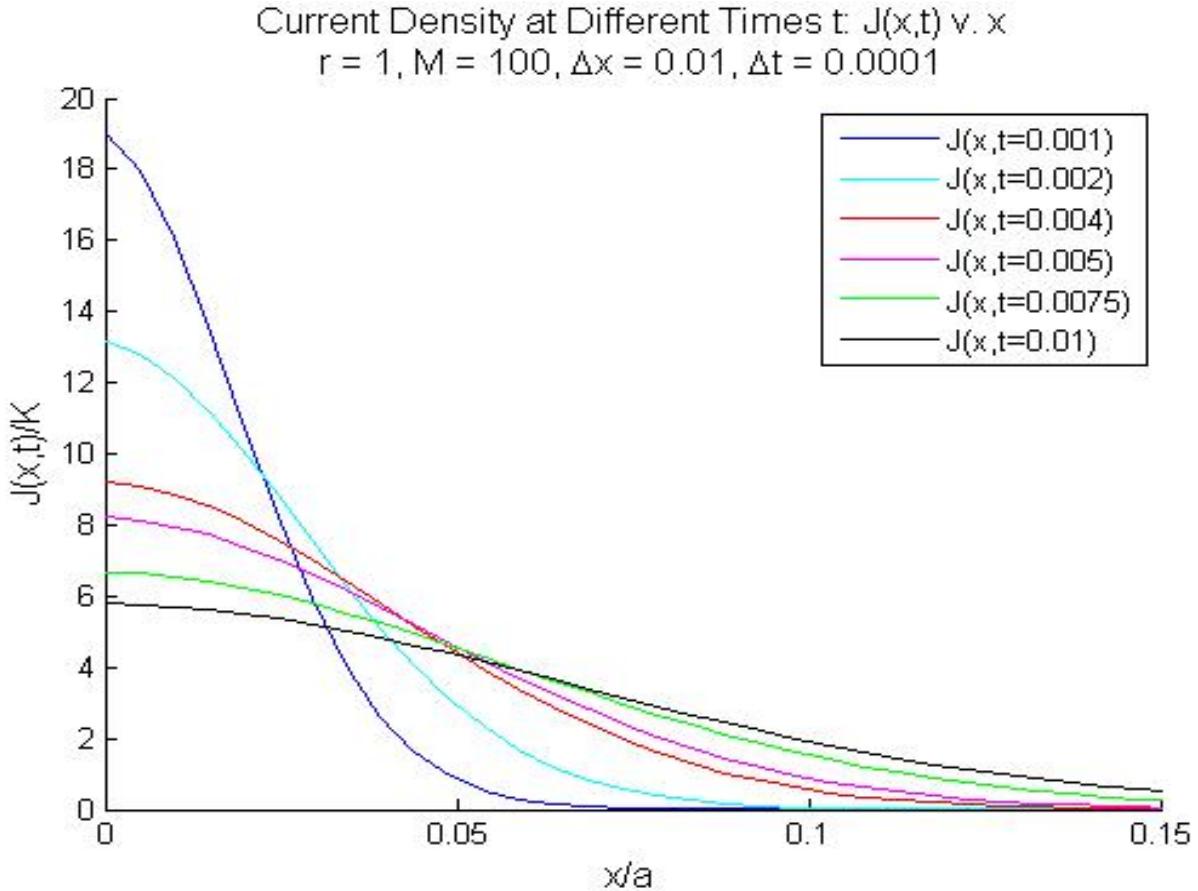


Figure 7: The current density at various times for $r = 1$ and $M = 10$. Notice how at short times the current density becomes very large. This is because the derivative of H approaches ∞ as $t \rightarrow 0$, as seen in Fig. 6.

Much like the magnetic field H , the current density J needs long times in order to become uniform through the system due to the RC time constant. See, for example, how the current density

is very large and becomes larger at $x = 0$ for ever shorter times. This is because the derivative of H approaches ∞ at $x = 0$ as $t \rightarrow 0$ as a consequence of the boundary condition for $x = 0$. This makes sense physically because at $x = 0$ for $t = 0$ the current I is confined to an infinitely small volume V . Since the current density is given by,

$$J = I/V \quad (26)$$

J clearly blows up for infinitely small V .

From the asymptotic expression for the H -field given by equation (15), we know that the current density asymptotically approaches one, or $J_\infty(x) = 1$, after finding J using equation (23). It certainly appears from Fig. 7 that for long times the current density decreases towards one for small x and increases toward one for larger x . Again, this makes sense physically because as time becomes larger and surpasses the RC time constant, the current I should be the same value everywhere in x . Since the volume V of the whole system is constant, the current density given by equation (26) will be constant at long times when I is constant in x . Because we are using normalized unit, our value for the current density at long times should and will be $J_\infty(x) = 1$.

2.3.3 Optional: Computation Time versus M

It was mentioned in section one that the triadiagonal matrix algorithm goes like $\mathcal{O}(M)$. This means that the number of steps required, and therefore the time required, for the tridiagonal algorithm used to implement the Crank-Nicolson method should grow linearly with M , the number of equations to compute. This can be checked by placing a timer at the beginning of the algorithm in our code and then stopping the timer after the algorithm has completed. By finding the time T required for the algorithm to finish for different values of M and plotting this data in a T versus M plot, we can confirm that tridiagonal matrix algorithm goes like $\mathcal{O}(M)$ if the resulting data forms a straight line.

After implementing the procedure described above, we generated the flowing plot,

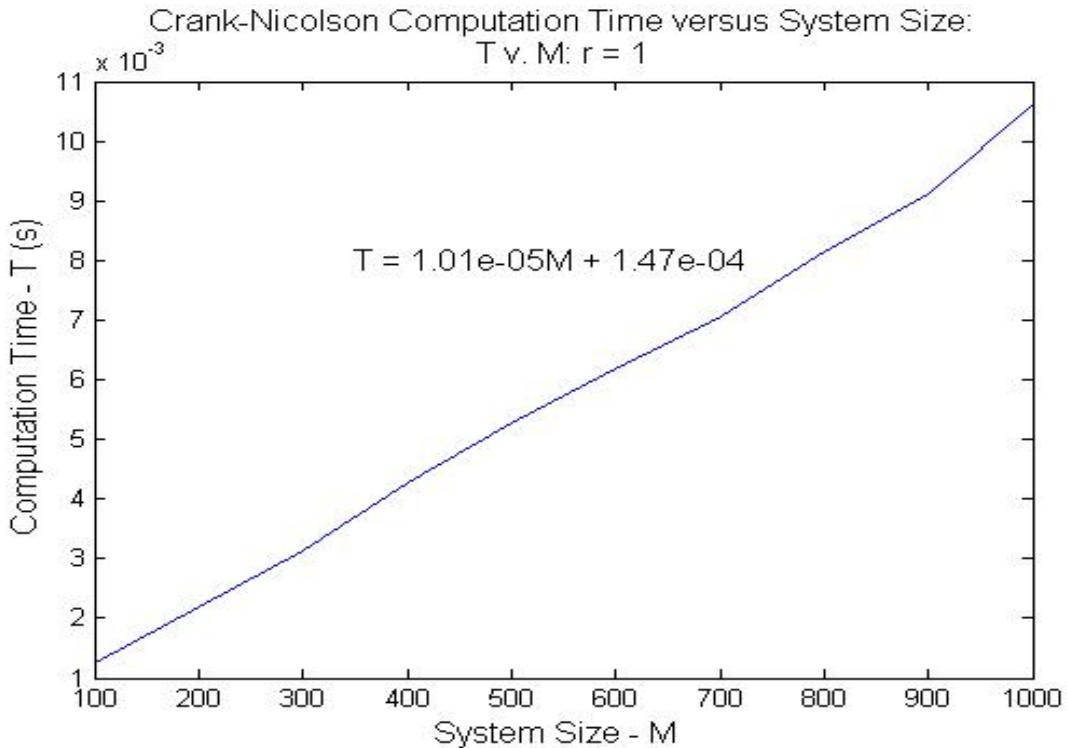


Figure 8: T v. M plot for the triadigonal matrix algorithm implementing the Crank-Nicolson method. Notice the plot is a straight line, confirming that the tridiagonal algorithm goes like $\mathcal{O}(M)$.

Because the T versus M plot above is in fact a straight line, it confirms that the tridiagonal matrix algorithm goes like $\mathcal{O}(M)$, or that the order of operations required to implement the Crank-Nicolson method is only M .

The tridiagonal matrix algorithm implementing the Crank-Nicolson method is not the entire algorithm used to compute H at all time steps, however. The triadiagonal algorithm by itself only implements the Crank-Nicolson method at one time step. Therefore, a for-loop was used to find all values of H at every time step. Placing the timer outside of this for-loop resulted in the following plot,

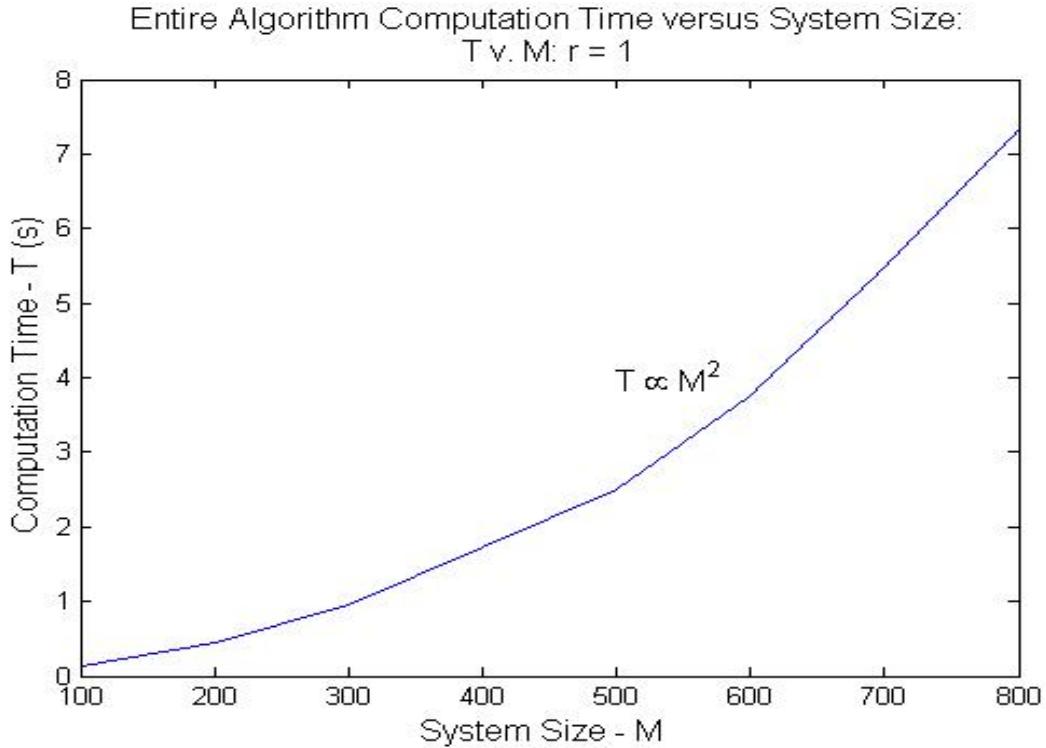


Figure 9: T v. M plot for the entire algorithm, including the for-loop which calculates H at successive times. The tridiagonal algorithm and for-loops go like $\mathcal{O}(M)$, so the tridiagonal algorithm nested within a for-loop goes like $\mathcal{O}(M^2)$

The plot of T versus M above if for the entire algorithm used to compute the data points of H for time steps specified. The plot also happens to look a lot like a parabola, indicating the out entire algorithm, for-loop included, goes like $\mathcal{O}(M^2)$. This makes sense because for-loops are linear in M , going like $\mathcal{O}(M)$. Because the triadiagonal algorithm also goes like $\mathcal{O}(M)$, as we showed above, then the tridiagonal algorithm nested inside a for-loop should go like $\mathcal{O}(M^2)$. Therefore, the number of steps required to evaluate our entire algorithm, for-loop included, is of order M^2 .

References

- [1] Horowitz, Paul, and Winfield Hill. *The Art of Electronics*. 3rd ed. Cambridge: Cambridge UP, 1989. Print.
- [2] "Magnetic Diffusion Transient Response." *MIT*. MIT, n.d. Web. 03 Feb. 2016. web.mit.edu/6.013_book/www/chapter10/10.6.html
- [3] "p-n junction." Wikipedia. Wikimedia Foundation, n.d. Web. 30 June 2016. https://en.wikipedia.org/wiki/P-n_junction
- [4] Stickler, Benjamin A., and Ewald Schachinger. *Basic Concepts in Computational Physics*. Cham, Switzerland: Springer, 2014. Print.