Solution to Heat Equation

Ian William Hoppock

May 5, 2017

1 General Remarks and Documentation

The programme is written in C, and it solves the one-dimensional heat equation

$$u_t = \alpha u_{xx}$$

using a Crank-Nicolson method in time and a finite difference for space. The Dirichlet boundary conditions are $u(0,t) = u(1,t) = 0$, and the initial condition is $u(x,0) = \sin 4\pi x$.

In this situation, Crank-Nicolson is

$$\frac{u_j^{n+1} - u_j^n}{dt} = \frac{\alpha}{2dt^2}\left((u_{j+1}^{n+1} - 2_j^{n+1} + u_{j-1}^{n+1}) + (u_{j+1}^n - 2_j^n + u_{j-1}^n)\right).$$

For ease of calculation, allow $r = \alpha dt/(2dx^2)$. Then we have

$$-ru_{j+1}^{n+1} + (1+2r)u_j^{n+1} - ru_{j-1}^{n+1} = ru_{j+1}^n + (1-2r)u_i^n + ru_{j-1}^n,$$

a tridiagonal problem. We simply employ Thomas's algorithm to solve the now computationally easy $Ax = b$ linear problem.

The parameters of the partial differential equation may be modified in the `main` C function (in the `new.c` file); it is straight forward. The `main` function uses the header `linear_algebra.h`, which is the home of many of my personal macros, to call on `heatequation_cn` function (in the `heat_cn.c` file). This, in turn, calls the `norm` function (in the `norm.c` file), for calculating the 2-norm of a one-dimensional array, and the `tri` (in the `tri.c` file) function, for calculating the solution to the linear $Ax = b$ problem such that A is strictly tridiagonal. This function is Thomas's algorithm for tridiagonal matrices, and it was taken and adapted from open source code. The user may play around with the algorithm in the `test_tri.c` file. Beyond the called functions, the `vector.c` file is used heavily.

Due to the number of files and for ease of use, there is an accompanying `Makefile`, and the user should compile this by using some variation of the following steps: (1) `autoreconf -i` (2) `./configure` (3) `make` (4) `./new`. The results will be printed in *rows* in the `results.txt` file and should be graphed appropriately (e.g., MatPlotLib can transpose this and then simply graph each column with a single command). The accompanying file that will be produced is `spacesteps.txt`, which can be plotted as the x-axis, against the `results.txt`, such that one may see the appropriate boundary conditions are maintained (as opposed to the results being graphed against the step count). Moreover, the `norms.txt` and `timesteps.txt` will also be produced. The latter serves as the x-axis to the former.

Feel free to modify any of the variable inputs, provided they make sense. If the user wishes to modify the initial condition, he or she must simply change the initialising function (called `InitialCondition`) for that, which is located immediately above the `main` function, as functions within functions are a certain kind of unspeakable evil in C, and the subsequent functions that are called have the necessary abstraction.

If the user does not appreciate Autotools (of if the user does not have them installed), he or she must use some appropriate variation of the shell command `gcc new.c vector.c tri.c norm.c heat_cn.c`.

2 Exact Solution

We solve the heat equation with boundary conditions $u(0,t) = 0$ and $u(1,t) = 0$ and initial condition $u(x,0) = \sin 4\pi x$. Assume separability such that $u(x,t) = X(x)T(t)$. Then

$$\frac{T'(t)}{\alpha T(t)} = \frac{X''(x)}{X(x)} = -\lambda, \text{ constant}$$

which results in the reduced system of ordinary differential equation

$$\begin{cases} T'(t) + \alpha\lambda T(t) = 0 \\ X''(x) + \lambda X(x) = 0 : \quad X(0) = 0,\, X(1) = 0. \end{cases}$$

The first equation, of course has solution of,

$$T(t) = e^{-\lambda\alpha t}.$$

For the second equation, we now have a complete Sturm-Liouville eigenvalue problem, and we must evaluate three possible cases of λ:

1. $\lambda = 0$:

$$\begin{aligned} X(x) &= C_1 x + C_2 \\ X(0) &= 0 = C_2 \\ X(1) &= 0 = C_1. \end{aligned}$$

   Hence, $\lambda = 0$ is not a non-trivial eigenvalue.

2. $\lambda > 0$:

$$\begin{aligned} X(x) &= C_1 \cos\sqrt{\lambda}x + C_2 \sin\sqrt{\lambda}x \\ X(0) &= 0 = C_1 \\ X(1) &= 0 = C_2 \sin\sqrt{\lambda}. \end{aligned}$$

Here, we cannot have C_2 be zero or we will again be stuck with a trivial set of eigenvalues. Instead, allow $\sin\sqrt{\lambda} = 0$ such that we have a spectral equation and discrete eigenvalues

$$\lambda_n = (n\pi)^2,$$

where $n = 1, 2, \ldots$. Hence our eigenfunctions on this discretised domain are

$$X_n(x) = C_2 \sin n\pi x.$$

3. $\lambda < 0$:

$$\begin{aligned} X(x) &= C_1 \cosh\sqrt{-|\lambda|}x + C_2 \sinh\sqrt{-|\lambda|}x \\ X(0) &= 0 = C_1 \\ X(1) &= 0 = C_2 \sinh\sqrt{-|\lambda|} \Rightarrow C_2 = 0. \end{aligned}$$

Thus, we, again, have no non-trivial eigenvalues.

The results are, of course, immediately obvious given the Dirichlet-Dirichlet boundary conditions. Nonetheless, we gave a thorough treatment. Now, the superposition of solution gives

$$u(x,t) = \sum_{n=1}^{\infty} C_n \sin(n\pi x) e^{-\alpha n^2 \pi^2 t}$$

Solving for the coefficients, we must have

$$u(x,0) = \sin 4\pi x = \sum_{n=1}^{\infty} C_n \sin n\pi x.$$

It is immediately obvious without using orthogonality conditions that $C_4 = 1$ and all others equal zero. Hence, we have an exact solution of

$$u(x,t) = \sin(4\pi x) e^{-16\alpha\pi^2 t}.$$

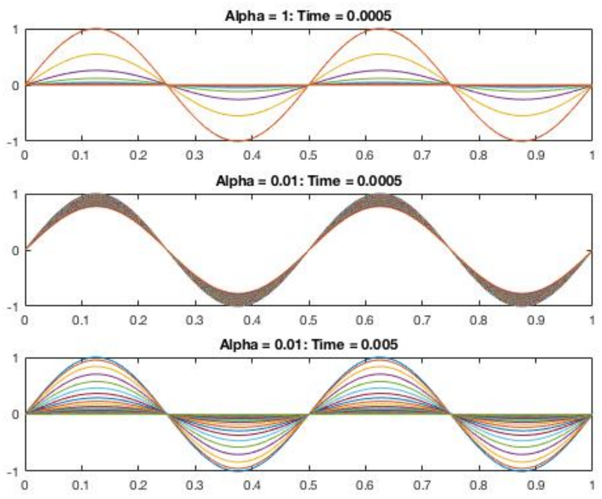
We can see the damping ratio σ is

$$\sigma = 16\alpha\pi^2,$$

which we will compare to the norms of the diffusion time steps shortly. It is clear that, where the damping ratio is smaller, there will be slower diffusion. I.e., it is fully reasonable to expect $\alpha = 1$ to be the fastest equation to diffuse and $\alpha = 0.0001$ to be the slowest. Moreover, since the difference is of four orders of magnitude, we expect this to be substantially slower.

3 Results

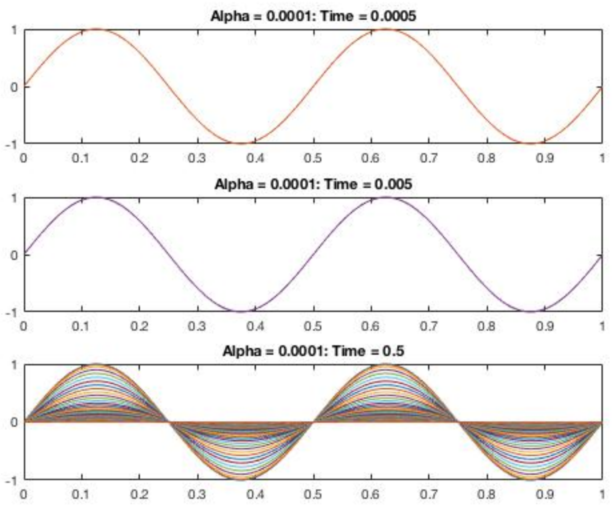
We see we have a working code that solves the one dimensional heat equation using Crank-Nicolson in time and finite difference in space. First consider the following figure in which we consider the first two values of $\alpha = 1$, 0.01:



It is clear from the first panel that we have complete diffusion of the heat equation in this time interval for $\alpha = 1$. This is consistent with our understanding of the damping ratio is the exact solution.

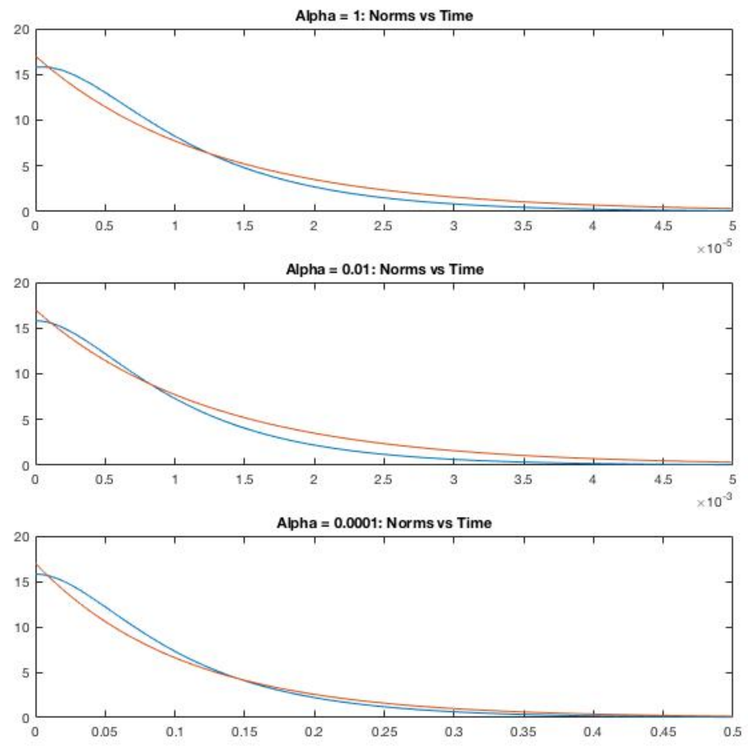
The the second and third panels, we consider diffusion with $\alpha = 0.01$. The former is a snapshot of where the diffusion process is just as $\alpha = 1$ has completed, i.e., they are the same time. The latter shows approximately the time at which $\alpha = 0.01$ has completely diffused. It is substantially slower.

The graphs for $\alpha = 0.0001$ at these time steps are shown in the first and second panels of the following figure:



It is clear that there is very little to no diffusion at the first two end times. It is only when we move two full orders of magnitude through time that we see complete diffusion, as seen in the third panel. This is a snap shot of the time at which the graph of the final time step achieves a reasonably flat line (i.e., full diffusion).

How do we know that we are right? Consider the following figure:



In each panel we see two lines. One is the numerically obtained damping ratio (blue) and the other is the appropriately scale exact value (not blue). To obtain the numerical value, we simply took the 2-norm of the solution for the heat equation at each time step. What is visible makes logical sense, since we are viewing the process of diffusion.

Now, the two curves in each panel match. This strongly suggests that we have a functioning numerical simulation. Beyond this, there are certain features that can be seen in the images that have show we have high numerical stability, e.g., the roots of the equation are consistent through time, the points at the boundary do not show a 'wiggle' or discontinuity, the wave never goes out of the natural condition provided by $f(x) = \sin(4\pi x)$, i.e., $x \in [0, 1]$ and $f(x) \in [-1, 1]$.