Crank-Nicolson Method Project

Jacob Moujalli - 13282064

November 6, 2022

1 Crank-Nicolson Method

The Crank-Nicolson (CN) method is an implicit finite difference scheme that approximates non-linear differential systems [1].

The CN method is used, in the context of this code, to solve the heat equation–

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{1}$$

–as it uses a tridiagonal linear system which can be easily solved with linear algebra packages in python, such as Scipy. It also has second order convergence and unconditional stability [3].

The Crank-Nicolson (CN) method uses a combination of the forward Euler method at n and the backwards Euler method at n+1. This is visualised with the stencil in figure 1.

Crank-Nicolson Stencil

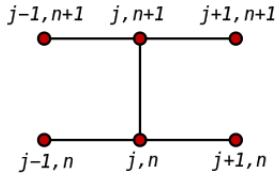

Figure 1: A visualisation of how the Crank-Nicolson method steps through time and space. [3]

The CN method in terms of the heat equation can be written as:

$$-ru_{i+1}^{n+1} + (1+2r)u_i^{n+1} - ru_{i-1}^{n+1} = ru_{i+1}^{n} + (1-2r)uu_i^{n} + ru_{i-1}^{n} \tag{2}$$

Where the left side is for the unknown values of u (n+1), the right side is for the known values of u (n), "i" represents the position in space and $r = \frac{\alpha \Delta t}{2(\Delta x)^2}$.

This system can be broken down into 4 arrays, 2 matrices and 2 vectors, which create the linear system:

$$Au_{n+1} = Bu_n \longrightarrow u_{n+1} = A^{-1}Bu_n \tag{3}$$

2 1D Numerical Implementation

The "heatpde" function in the "heat.py" file returns the numerical solution for the 1D heat equation.

The u array is created such that it has the size nx by nt. Where nx is the number of space steps and nt is the number of time steps. This gives the u_n and the u_{n+1} vectors for any point. The array is filed, using a loop, with some specified function, f(x), after which the boundary conditions are set.

The A and B matrices are sparse tridiagonal matrices and so to speed up calculations the scipy.sparse package is used. The matrices are defined as:

$$A = \begin{bmatrix} -r & 1+2r & -r \\ ... & ... & ... \end{bmatrix} \tag{4}$$

$$B = \begin{bmatrix} r & 1-2r & r \\ ... & ... & ... \end{bmatrix} \tag{5}$$

Where the left column is -1 from the center diagonal, the center column is 0 from the center diagonal, and the right column is +1 from the center diagonal.

The matrices are square of size nx by nx. After being defined using the sparse notation they are put into a usable form with the sparse.csc_ matrix function.

Now with the u, A, and B arrays defined the solution is found by iterating through time, n. At each point the solution to equation 3 is found using the sparse.linalg.spsolve function. The arrays A and f are parsed into the function where: $f = B * u +$ fix, at all points in space for the given time. The added array, fix, is created and added to fix the ends of the B*u array. This is done in order to correct the ends which are transformed at each iteration.

Once u_{n+1} is solved the boundaries are defined once again and the next iteration run.

3 Verification

To verify the accuracy of the numerical solutions some analytical solutions are needed. In the case of the heat equation these can be found using separation of variables and a Fourier series [2]. The functions and solutions used are:

$f(x) = 0.5$

$$u(x,t) = \sum_{n=1}^{\infty} -\frac{(-1)^n - 1}{2\pi n} sin(n\pi x) e^{-0.5(n\pi)^2 t} \tag{6}$$

With $\alpha = 0.5$, Length (L) = 1, and the boundaries equal to zero.

$f(x) = \cos(\pi x)$

$$u(x,t) = \sum_{n=1}^{\infty} \frac{2n((-1)^n + 1)}{\pi(n+1)(n-1)} sin(n\pi x) e^{-0.5(n\pi)^2 t} \tag{7}$$

With $\alpha = 0.5$, Length (L) = 1, and the boundaries equal to zero.

These analytical solutions are implemented in the "analytical.py" file which are then used in the "test2.py" and "test4.py" files for comparison with the numerical solution.

The comparison between the analytical and numerical solutions is done using the mean-squared average which is defined as a function in the test 2 and 4 files. The square root of the mean-squared average is taken to be the average error in the solution.

4 2D Numerical Implementation

There are several methods for solving the 2D heat equation using the Crank-Nicolson method. In the context of this code the 1D implementation is reused.

The u array has 3 indices: x, y, and t. It is created using two loops with a 2D function, f, used to define the temperature at each point. The A and B matrices and the solution are found using the same method as in the 1D implementation.

The implementation solves the heat equation at each vertical position in the 2D array and then again at each horizontal position in the array. The r value is divided by two as there are two calculations for each x,y position.

5 Test 1

Test 1 demonstrates the 2D implementation.

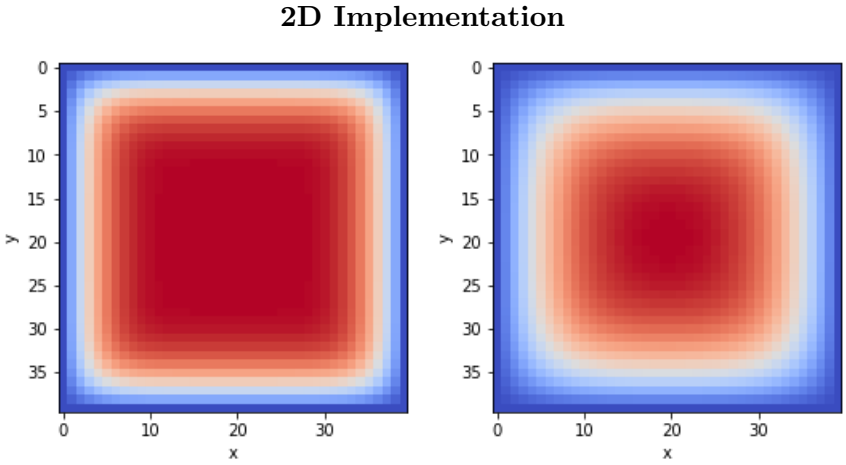

Figure 2: Two plots showing the heat at two different points. Right is t = 0 and left is t = 0.5 s. Red is hot, blue is cold.

6 Test 2

The test 2 functions verify the 1D numerical implementation against analytical solutions.

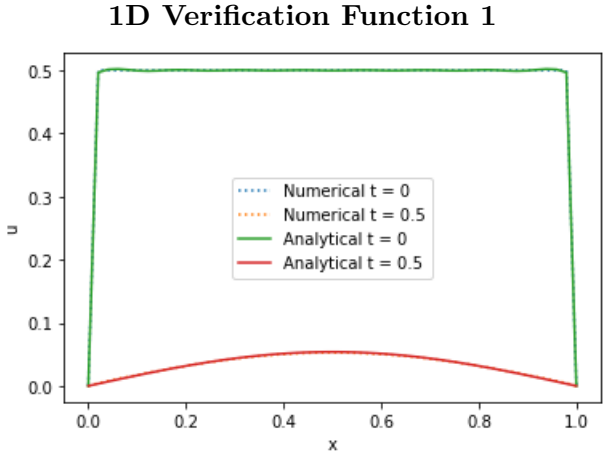

Figure 3: Plot of the numerical and analytical solutions for the function f = 0.5 at two times. Average Error = 0.002402.

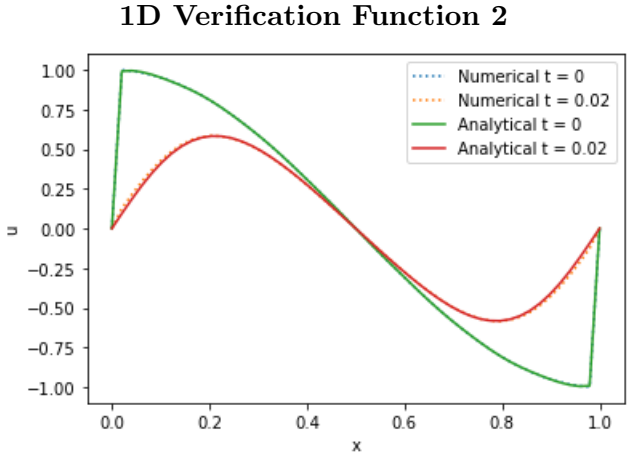

Figure 4: Plot of the numerical and analytical solutions for the function f = cos(πx) at two times. Average Error = 0.002900.

7 Test 3

The test 3 functions demonstrate that the numerical method can solve any function whereas the analytical solution can only be found for continuous functions.

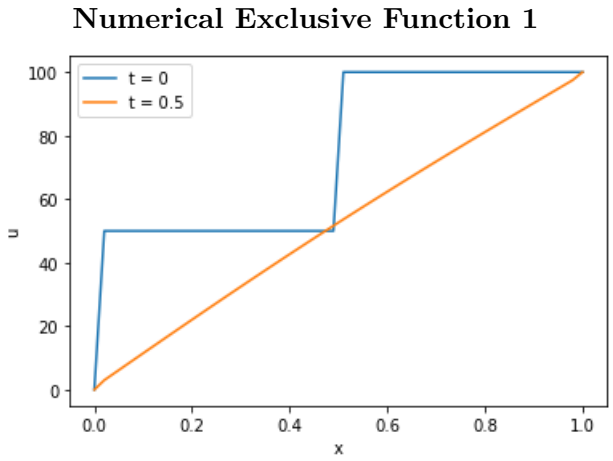

Figure 5: Plot of the step-wise function f = 50, x $<$ 0.5; 100, x $>$ 0.5 for two times.

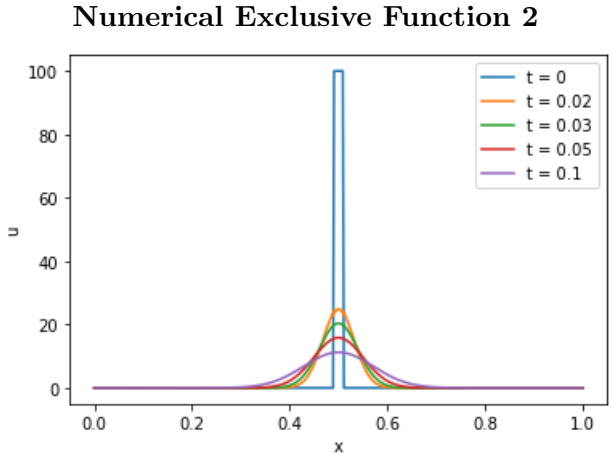

Figure 6: Plot of the impulse function f(0.5) = 100 for five points in time.

8 Test 4

Test 4 calculates the convergence rate for the functions from test 2. The convergence rate for example 1 is 1.517 and 3.044 for example 2 where h is the step size. These agree with the known convergence for CN method of 2.

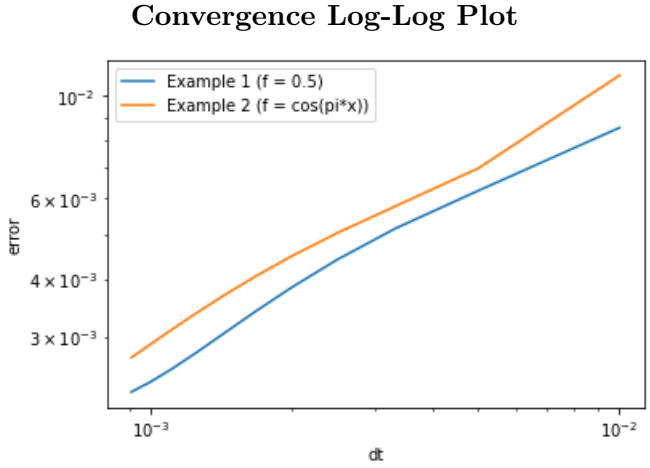

Figure 7: Plot of the convergence for the two functions seen in test 2.

References

[1] Encyclopedia of Mathematics. Crank-nicolson method, 2022. [Online; accessed 3-November-2022].

[2] Paul Dawkins. Paul's online notes — solving the heat equation, 2022. [Online; accessed 3-November-2022].

[3] Wikipedia contributors. Crank–nicolson method — Wikipedia, the free encyclopedia, 2022. [Online; accessed 3-November-2022].