MECE 5397: Scientific Computing for Mechanical Engineers

Solution of the discretized diffusion equation in two spatial dimensions with several time integration schemes

David Newman

Project # B01-5

Contents

Abstract

Two different time integration schemes are explored in the pursuit of solving the diffusion equation in two spatial dimensions. Grid independence as well as error analyses are carried out for several relevant time steps in the domain of interest. The two schemes utilized are the Alternating Direction Implicit (ADI) and the explicit scheme. Good agreement has been found between both schemes provided the stability criteria for the explicit method is satisfied. A solution to the equation is manufactured and the ADI scheme is compared to an analytical solution from which the order of accuracy of the scheme is verified.

Problem Statement

Write a computer code to solve the two-dimensional diffusion equation

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

on the domain

$$0 < x < 2\pi$$
$$0 < y < 2\pi$$
$$t \geq 0$$

Subject to

$$\frac{\partial u}{\partial y}(x, y = 0) = 0 \tag{2}$$

$$\frac{\partial u}{\partial y}(x, y = 2\pi) = 0 \tag{3}$$

$$u(x = 0, y) = y^2 \sin\left(\frac{y}{4}\right) \tag{4}$$

$$u(x = 2\pi, y) = \cos(\pi y)\cosh(2\pi - y) \tag{5}$$

$$u(x, y, t = 0) = 0 \tag{6}$$

Discretization

For the purposes of this assignment, the solution to the differential equation will be denoted by u and the exact solution will be noted by u_{exact}. For compactness, the solution at time coordinate t^n and spatial coordinates x_j and y_k will be given by

$$u\left(t^n, x_j, y_k\right) = u_{j,k}^n$$

The second order centered discretization of the second derivatives in space will be denoted,

$$\Delta_x^2 u_{j,k} = \frac{u_{j-1,k} - 2u_{j,k} + u_{j+1,k}}{\Delta x^2}$$

$$\Delta_y^2 u_{j,k} = \frac{u_{j,k-1} - 2u_{j,k} + u_{j,k+1}}{\Delta y^2}$$

With this notation in place, the Crank-Nicolson discretization of the diffusion equation is a two-level time scheme given by

$$\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} = \frac{1}{2} D\left(\Delta_x^2 u_{j,k}^{n} + \Delta_y^2 u_{j,k}^{n} + \Delta_x^2 u_{j,k}^{n+1} + \Delta_y^2 u_{j,k}^{n+1}\right) \tag{7}$$

Which we know is unconditionally stable for Dirichlet boundary conditions and second order accurate in time and space. Note that the diffusion coefficient, *D*, is shown in equation (7) and hereafter to allow for a general discussion of discretizations of the diffusion equation.

While the Crank-Nicolson method will certainly solve the differential equation given in (1), its implementation in two spatial dimensions will require the solution of a matrix with five diagonals. There is no simple algorithm for solving such a system, so we will instead use the Alternating Direction Implicit (ADI) scheme which is similar in form. The ADI scheme is characterized by taking two half steps in time, one implicit in x and one implicit in y. Refer to equation (8) and equation (9).

$$\frac{u_{j,k}^{n+\frac{1}{2}} - u_{j,k}^{n}}{\frac{\Delta t}{2}} = D\left(\Delta_x^2 u_{j,k}^{n+\frac{1}{2}} + +\Delta_y^2 u_{j,k}^{n}\right) \tag{8}$$

$$\frac{u_{j,k}^{n+1} - u_{j,k}^{n+\frac{1}{2}}}{\frac{\Delta t}{2}} = D\left(\Delta_x^2 u_{j,k}^{n+\frac{1}{2}} + \Delta_y^2 u_{j,k}^{n+1}\right) \tag{9}$$

The ADI scheme results in a tri-diagonal matrix structure for each half step for which there are efficient algorithms to solve. In addition, the ADI scheme is second order accurate in the temporal and spatial domains.

Another method used to solve the equation in (1) is the explicit method, so named for the capability of isolating a single unknown at each time step. The discretization is shown in equation (10).

$$\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} = D\left(\Delta_x^2 u_{j,k}^{n} + \Delta_y^2 u_{j,k}^{n}\right) \tag{10}$$

This method is efficient to solve, however stability analysis reveals the stability criteria given in equation (11) which is extremely restrictive for two dimensions of space.

$$\frac{D\Delta t}{\Delta x^2} + \frac{D\Delta t}{\Delta y^2} \leq \frac{1}{2} \tag{11}$$

For a fine spatial grid spacing, the time step will have to be reduced to extremely small levels to meet the stability criteria.

Description of the Numerical Method

A Note on Implementing Neumann Boundary Conditions

Neumann boundary conditions are implemented for both schemes using the second order centered difference formula given by equation (12) which leads to the creation of so called "ghost nodes" which lie outside the domain of interest.

$$\frac{u_{j,k+1} - u_{j,k-1}}{2\Delta y} = \frac{\partial u_{exact}}{\partial y} + O(\Delta y^2) \tag{12}$$

For this discussion, we effectively replace the right-hand sides of equations of equations (2) and (3) with *v* and *w* for generality. At the first row in the domain we have *k = 0*,

$$\frac{u_{j,1} - u_{j,-1}}{2\Delta x} = v$$

So that

$$u_{j,-1} = u_{j,1} - 2v\Delta x$$

Similarly, for the last row in the domain we have *k = N* and following a similar procedure we obtain

$$u_{N+1} = u_{j,N-1} + 2w\Delta x$$

These expressions are then inserted into the discretization of the scheme providing for a second order accurate representation of the first derivative specified in equations (2) and (3).

ADI Scheme

For the ADI scheme, the following pseudocode describes the algorithm used to solve the problem:

1. Obtain all relevant parameters that specify the problem such as boundary conditions in space, an initial condition in time, and constant coefficients.
2. Compute derived parameters from step (1) that will be used frequently in the scheme.
3. Build the diagonals of each half step array to be solved. These arrays stay the same for each time step.
4. Main loop:
    a. Create the right-hand side for each tri-diagonal system in the first half step (each row of points in the spatial domain).
    b. Solve the first half step.
    c. Create the right-hand side for each tri-diagonal system in the second half step (each column of points in the spatial domain).
    d. Solve the second half step.
    e. Repeat starting at step *a)* until a specified number of iterations has been reached.

The result of this computation is a three-dimensional array with each page representing the results from a single time step, or equivalently, a single iteration of the main loop.

Some rearrangement of equations (8) and (9) reveals the following equation for the first and second half steps, respectively.

$$-\lambda_x u_{j-1,k}^{n+\frac{1}{2}} + (1+2\lambda_x) u_{j,k}^{n+\frac{1}{2}} - \lambda_x u_{j+1,k}^{n+\frac{1}{2}} = \lambda_y u_{j,k-1}^{n} + \left(1-2\lambda_y\right) u_{j,k}^{n} + \lambda_y u_{j,k+1}^{n} \qquad (12)$$

$$-\lambda_y u_{j,k-1}^{n+1} + \left(1+2\lambda_y\right) u_{j,k}^{n+1} - \lambda_y u_{j,k+1}^{n+1} = \lambda_x u_{j-1,k}^{n+\frac{1}{2}} + (1-2\lambda_x) u_{j,k}^{n+\frac{1}{2}} + \lambda_x u_{j+1,k}^{n+\frac{1}{2}} \qquad (13)$$

Where $\lambda_x = \frac{D\Delta t}{2\Delta x^2}$ and $\lambda_y = \frac{D\Delta t}{2\Delta y^2}$

Equation (12) is iterated for each row of points in the domain and results in a tri-diagonal system. Equation (13) is iterated for each column of points in the domain and results in a similar structure. It should be noted that the lower and upper boundaries are included with the interior points as unknowns because of the Neumann conditions given by equations (2) and (3). The resulting system for each half step is shown below:

$$\begin{bmatrix} 1+2\lambda_x & -\lambda_x & 0 & & & \\ -\lambda_x & 1+2\lambda_x & -\lambda_x & & 0 & \\ 0 & -\lambda_x & 1+2\lambda_x & & & \\ & & & 1+2\lambda_x & & \\ & 0 & & & \ddots & \\ & & & & -\lambda_x & 1+2\lambda_x \end{bmatrix} \begin{bmatrix} u_{1,0} \\ u_{2,0} \\ u_{3,0} \\ \vdots \\ u_{n-1,n} \\ u_{n,n} \end{bmatrix} = B_1$$

$$\begin{bmatrix} 1+2\lambda_y & -2\lambda_y & 0 & & & \\ -\lambda_y & 1+2\lambda_y & -\lambda_y & & 0 & \\ 0 & -\lambda_y & 1+2\lambda_y & -\lambda_y & & \\ & & & \ddots & \ddots & \\ & 0 & & & \ddots & \ddots \\ & & & & -2\lambda_y & 1+2\lambda_y \end{bmatrix} \begin{bmatrix} u_{1,0} \\ u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{n,n-1} \\ u_{n,n} \end{bmatrix} = B_2$$

While this represents the large-scale structure of the problem, in practice each row or column may be solved separately as its own tri-diagonal system. Only the diagonal elements of each array are stored in computer memory since both systems are sparse. Each system is solved using the Thomas algorithm.

Explicit Scheme

The following pseudocode is used to solve equation (1) via the explicit scheme:

1. Obtain all relevant parameters that specify the problem such as boundary conditions in space, an initial condition in time, and constant coefficients.
2. Compute derived parameters from step (1) that will be used frequently in the scheme.
3. Main Loop:
    a. Loop through first and last row simultaneously and determine the value of the solution at these points. Special treatment is required here due to the upper and lower boundary conditions.
    b. Loop through interior rows of the domain and determine the value of the solution at these points.

The explicit scheme given by (10), after solving for the single unknown is given by equation (14).

$$u_{j,k}^{n+1} = \left(1 - 2\frac{D\Delta t}{\Delta x^2} - 2\frac{D\Delta t}{\Delta y^2}\right)u_{j,k}^{n} + \frac{D\Delta t}{\Delta x^2}\left(u_{j-1,k}^{n} + u_{j+1,k}^{n}\right) + \frac{D\Delta t}{\Delta y^2}\left(u_{j,k-1}^{n} + u_{j,k+1}^{n}\right) \qquad (14)$$

There is no matrix system to solve for this scheme, making it very simple and quick to construct and execute.

Technical Specifications of Machine Used for Simulations

Table 1. Technical specifications of computer used for simulations

CPU Model	Intel Core i7-7700K
Number of Cores	4
Number of Threads	8
Base Frequency	4.20 GHz
Cache	8 MB SmartCache
TDP	91 W
Main Memory Quantity	16 GB (2 x 8 GB DDR3)
Storage (Boot)	250 GB (M.2 NVMe SSD)
Storage (HDD)	1 TB
GPU Model	NVIDIA GeForce GTX 1080

Results

The results for running the code for both methods solving equations (1) – (6) at a relatively fine discretization can be found in Figure 1 and Figure 2. A qualitative comparison of the two plots shows that they appear to be approximately the same.

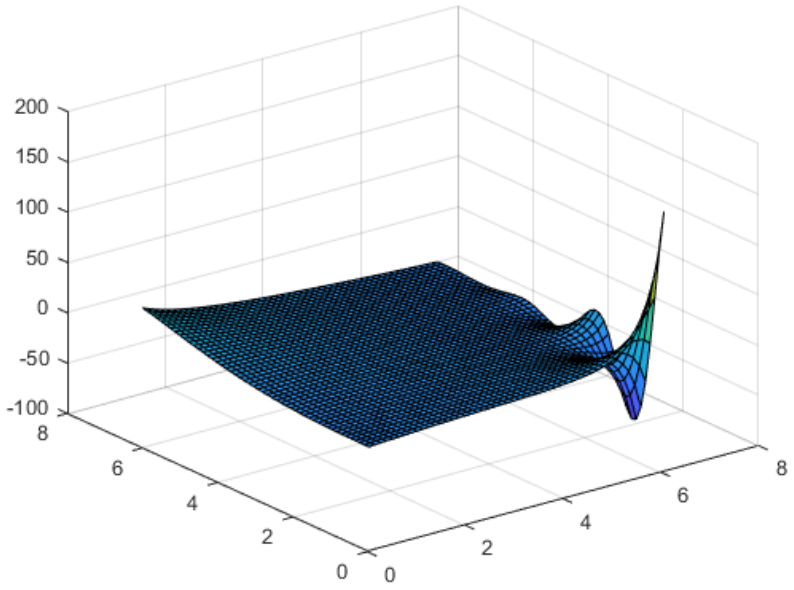

Figure 1. Explicit method for t = 30 and n = 50

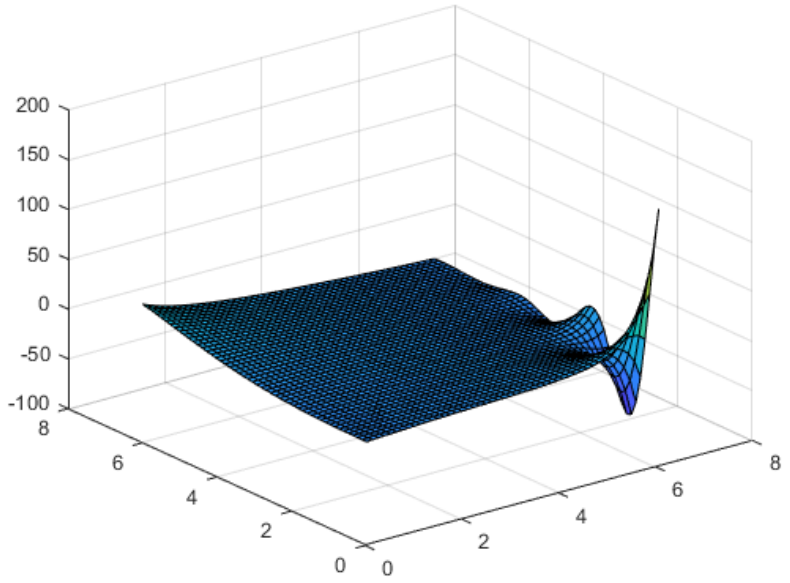

Figure 2. ADI method for t = 30 and n = 50

The simulation was performed for both methods on a square grid spacing by discretizing both spatial axes with 50 internal grid points. The simulation was halted at t = 30. The L^1, L^2, and L^∞ error between each solution at the final time step and at the half way point are shown in table 2.

Table 2. Error (differences) between ADI scheme and explicit scheme at maximum time step and half way point

Error type	**Error x 10^5 at T_{max}**	**Error x 10^5 at $T_{1/2}$**
L^1	1.0059	21.382
L^2	1.1067	23.526
$L\infty$	1.5497	33.673

Note that the vector norm of each solution is taken at each of the time steps mentioned in table 2.

To determine the necessary time to run the simulation until steady state was reached, a measure of the maximum change in the solution between time steps was analyzed. Figure 3 shows the maximum change in the solution between time steps plotted against time.

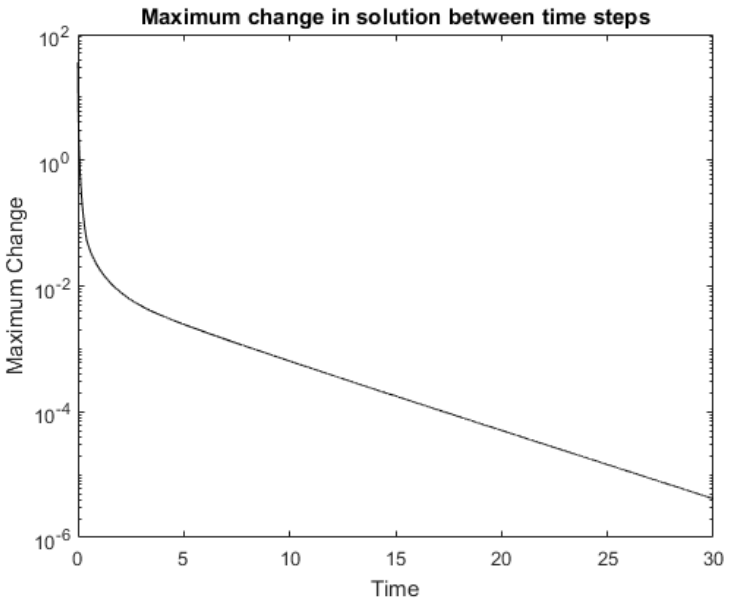

Figure 3. Change in time

From Figure 3, the maximum change in the solution is asymptotically approaching zero. We will consider a maximum change in the solution of less than 10^{-5} as steady in time.

To determine if the solution is grid independent, it would be preferable if the solution was changing in time in order to avoid the effects of the steady state region in the temporal domain. For this reason, the initial condition is changed from identically zero to identically 100 for the following qualitative comparison and the maximum time is changed from 30 to 2. For this discussion, the computation was carried out on a square domain such that each spatial axis contained the same number of nodes. Note that only the interior nodes are shown in these comparisons. Figures 4 – 6 show the explicit scheme and Figures 7 – 9 show the ADI scheme.

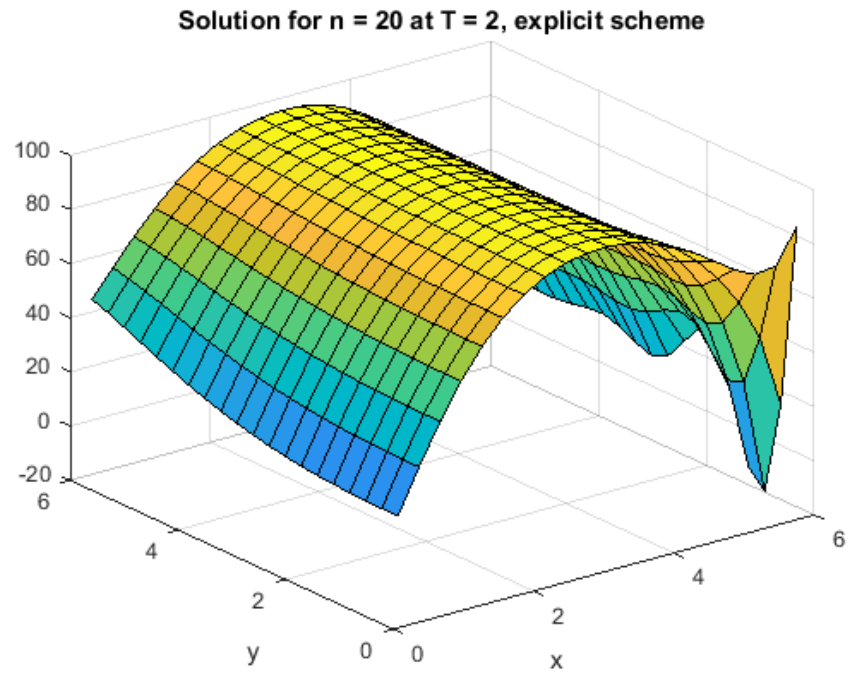

Figure 4. Explicit scheme for n = 20, T = 2

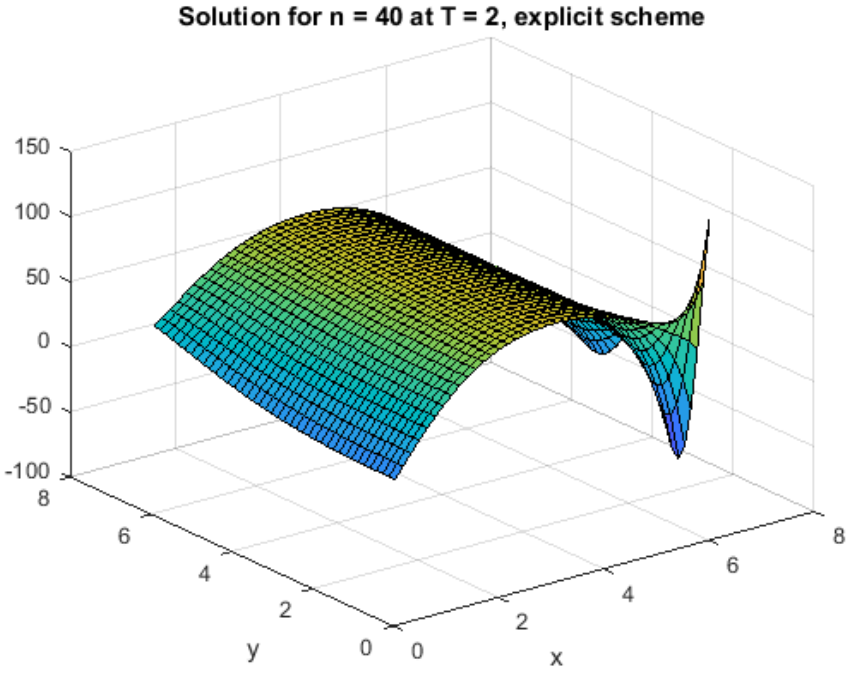

Figure 5. Explicit scheme for n = 40, T = 2

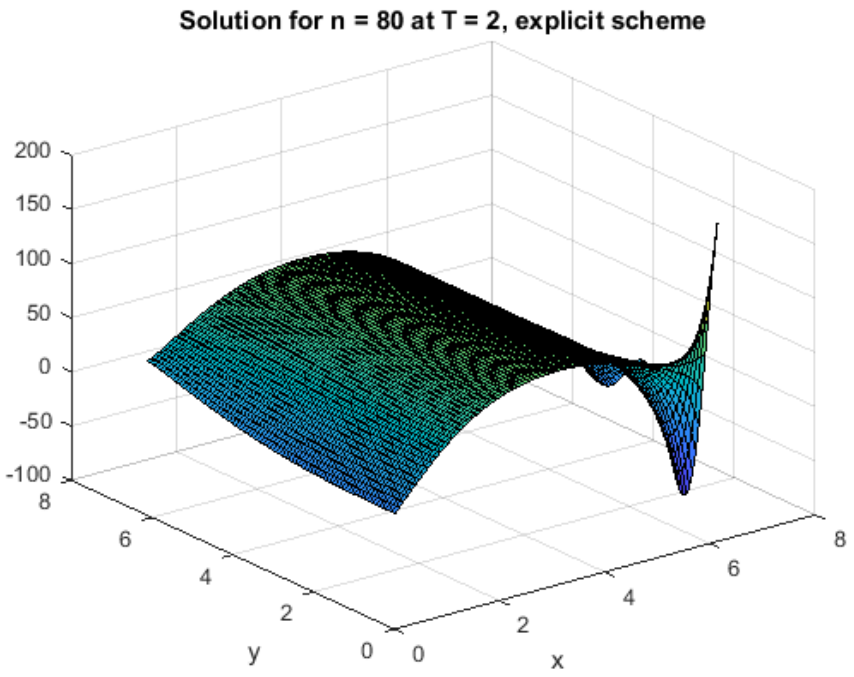

Figure 6. Explicit scheme for n = 80, T = 2

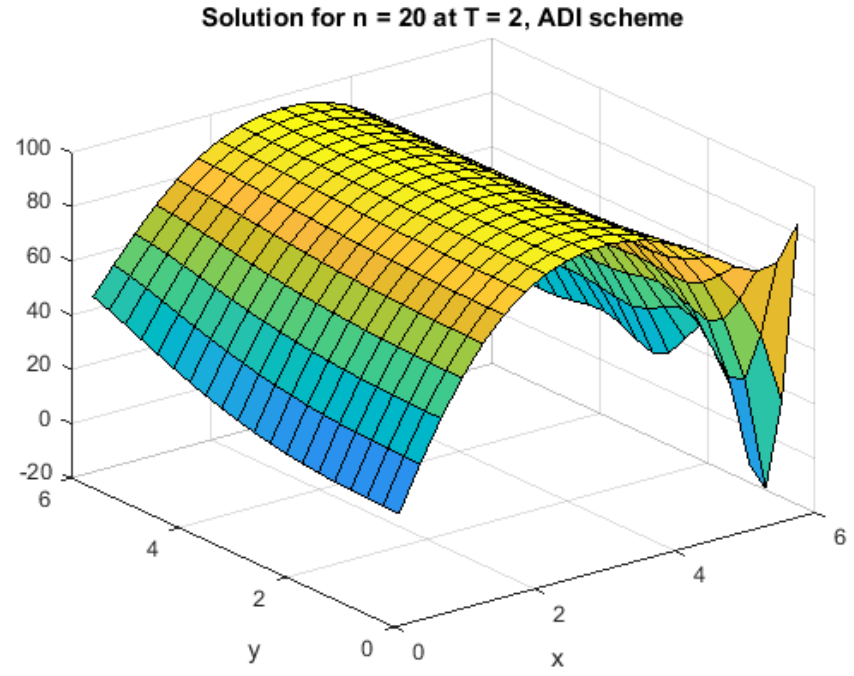

Figure 7. ADI scheme for n = 20, T = 2

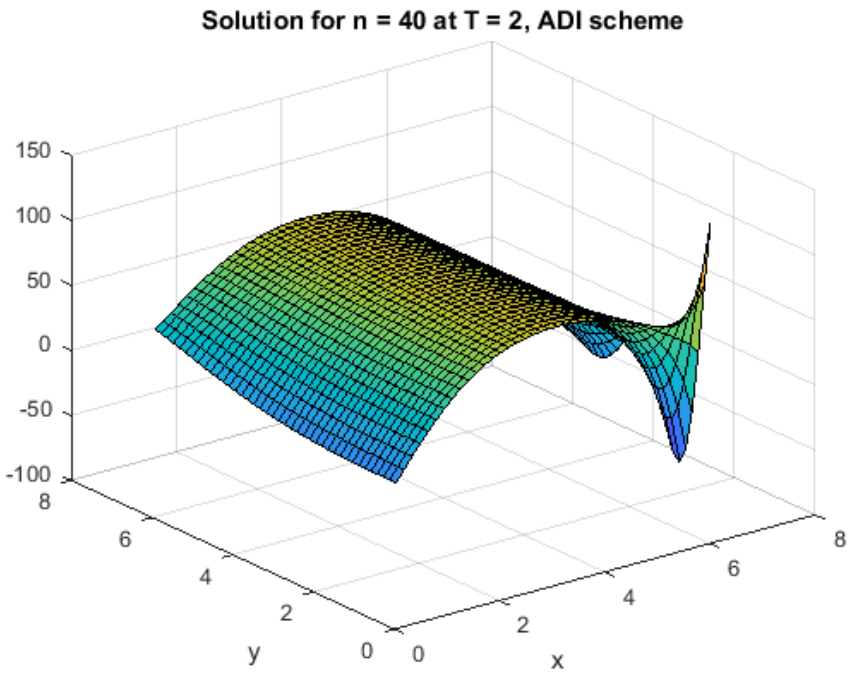

Figure 8. ADI scheme for n = 40, T = 2

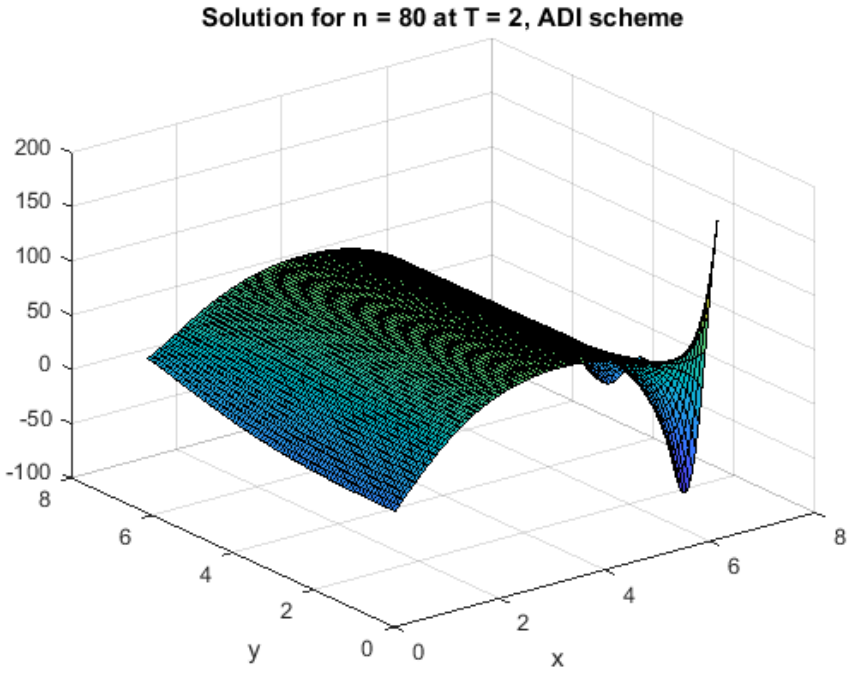

Figure 9. ADI scheme for n = 80, T = 2

Due to the stability criteria in equation (11), the time and spatial domains are coupled for the explicit scheme so strictly speaking the spatial domain was not the only thing tested in figures 4 – 6 however the time step was changed only as much as was required to satisfy the stability criteria.

One way to determine the order of accuracy of the scheme and its implementation in code is to "manufacture" a solution to the differential equation. One such solution that satisfies (1) along with its boundary conditions is shown below.

$$u_m(x, y, t) = e^{\frac{t}{\tau}}(\sin(x)\cos(y))$$

$$\frac{\partial u}{\partial y}(x, y = 0) = 0$$

$$\frac{\partial u}{\partial y}(x, y = 2\pi) = 0$$

$$u(x = 0, y) = 0$$

$$u(x = 2\pi, y) = \cos(\pi y)\cosh(2\pi - y)$$

$$u(x, y, t = 0) = \sin(x)\cos(y)$$

With this solution and boundary conditions, the only other parameter that needs to be satisfied in the software is the diffusion coefficient, *D*, which in this case must be

$$D = -\frac{1}{2\tau}$$

The initial condition of the manufactured solution is shown in Figure 11.

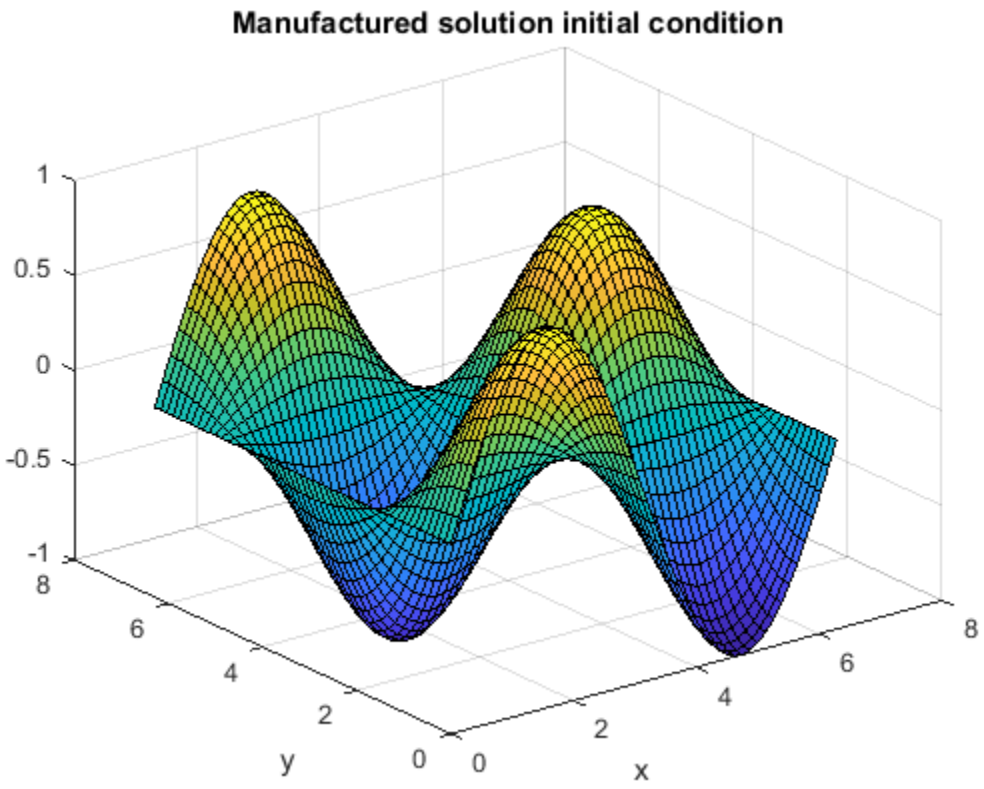

Figure 10. Initial condition for manufactured solution

The order of accuracy of the scheme in the spatial domain is summarized in Figures 11 – 13 as well as table 3. Note that this has only been performed for the ADI scheme as its spatial discretization is not coupled to the temporal discretization as is the case for the explicit scheme.

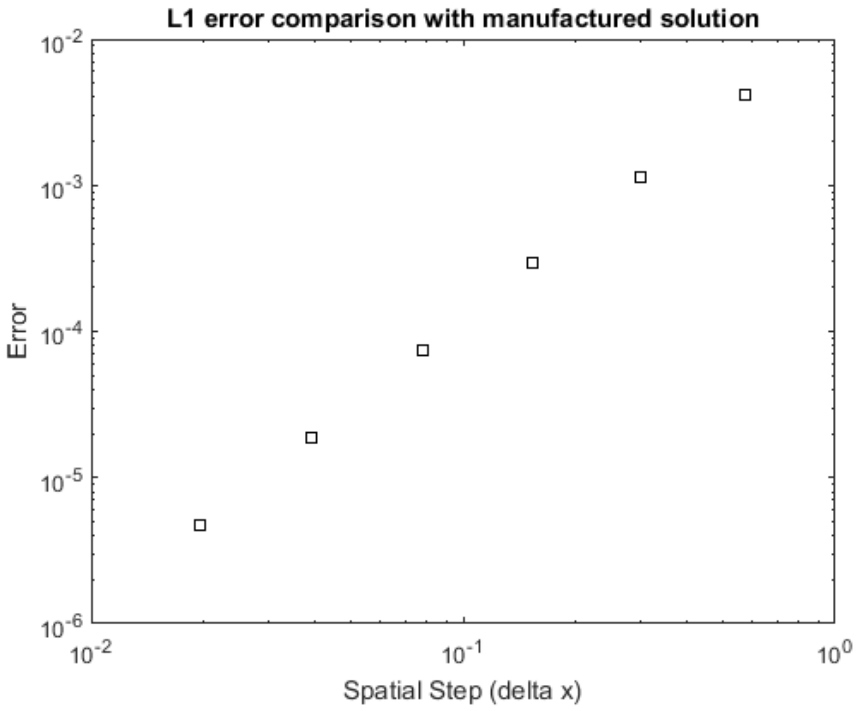

Figure 11. L1 error with manufactured solution

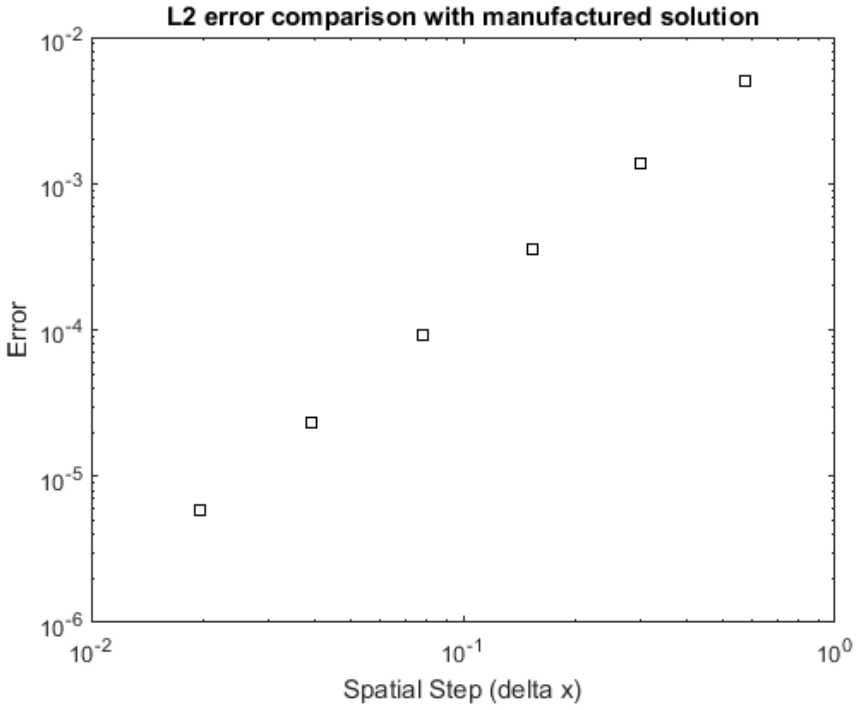

Figure 12. L2 error with manufactured solution

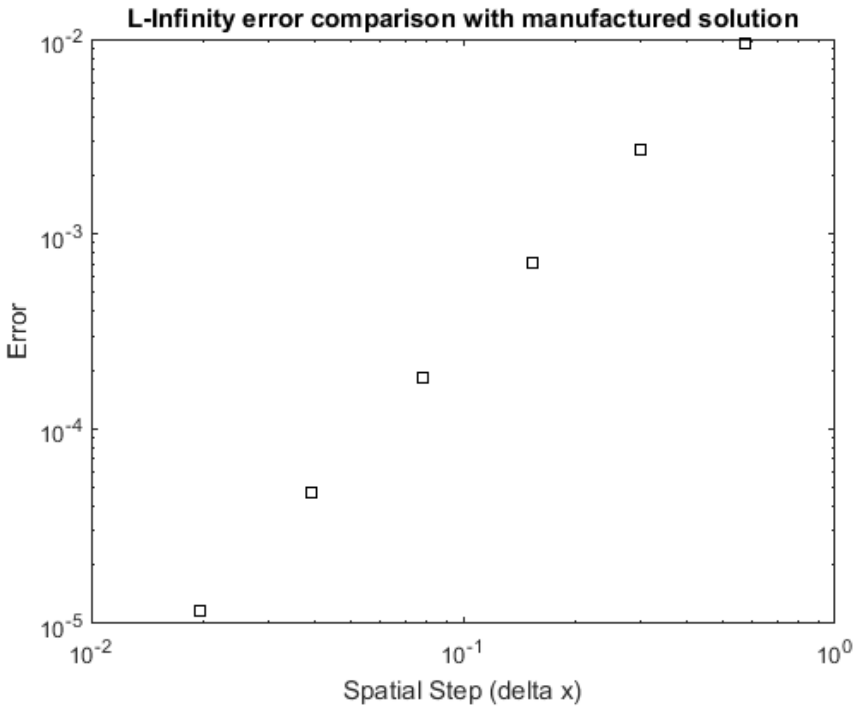

Figure 13. L-Infinity error with manufactured solution

Table 3. Calculated slope of error lines

	L1	L2	L-Infinity
Calculated slope	2.0099	1.9996	1.9882

Conclusion

Two numerical methods were used to solve the diffusion equation in two spatial dimensions. The ADI (Alternating Direction Implicit) scheme and explicit scheme have been implemented and tested in MATLAB. Good agreement has been found between both solutions. The order of accuracy of the ADI scheme has been confirmed with comparison to a manufactured solution. There is much left on the table in the MATLAB implementation of both schemes. Further optimization would include stride one access to the innermost loops of each script. In the current implementation, spatial locality is not maintained. Sacrifices to readability could also be made in favor of optimization. Moving the functions inside the main loop of each scheme would accomplish this by reducing overhead.