*MECE 5397*

*Scientific Computing for Mechanical Engineers*

**Final Project**

**Solving the 2D Diffusion Equation using the Explicit and ADI Method**

Samar Mathur

*University of Houston*

**The Diffusion equation is a widely applicable partial differential equation. It was first used to solve problems related to heat transfer, but it can be used in many other ways. It has applications in other fields such as material science, information sciences, and life sciences. Though there are analytical solutions to the diffusion equation, in most real world applications they cannot be used. Numerical methods are used to solve diffusion equations without an analytical solution. Two methods are going to be examined in this study. The first method is the explicit discretization of the diffusion equation. This discretization can be solved using iterative methods. The second method is the implicit discretization of the diffusion equation. This method requires the use of the alternating direction implicit (ADI) scheme to solve.**

**Problem Statement**

The equation presented that is being solved is

This is the 2D diffusion equation. The domain being examined in this paper is

And the boundary conditions given are

**Discretization**

The given diffusion equation consists of one first-order term and two second-order terms. The second order terms can be discretized using the standard second-order approximations

(x-direction approximation)

(y-direction approximation)

The first-order term can be discretized using the forward formula as follows

The discretized diffusion equation can now be written as

**Explicit Method**

Assuming that the equation can be rearranged to become

Where . This can be easily solved using an explicit scheme.

The explicit method is implemented as follows:

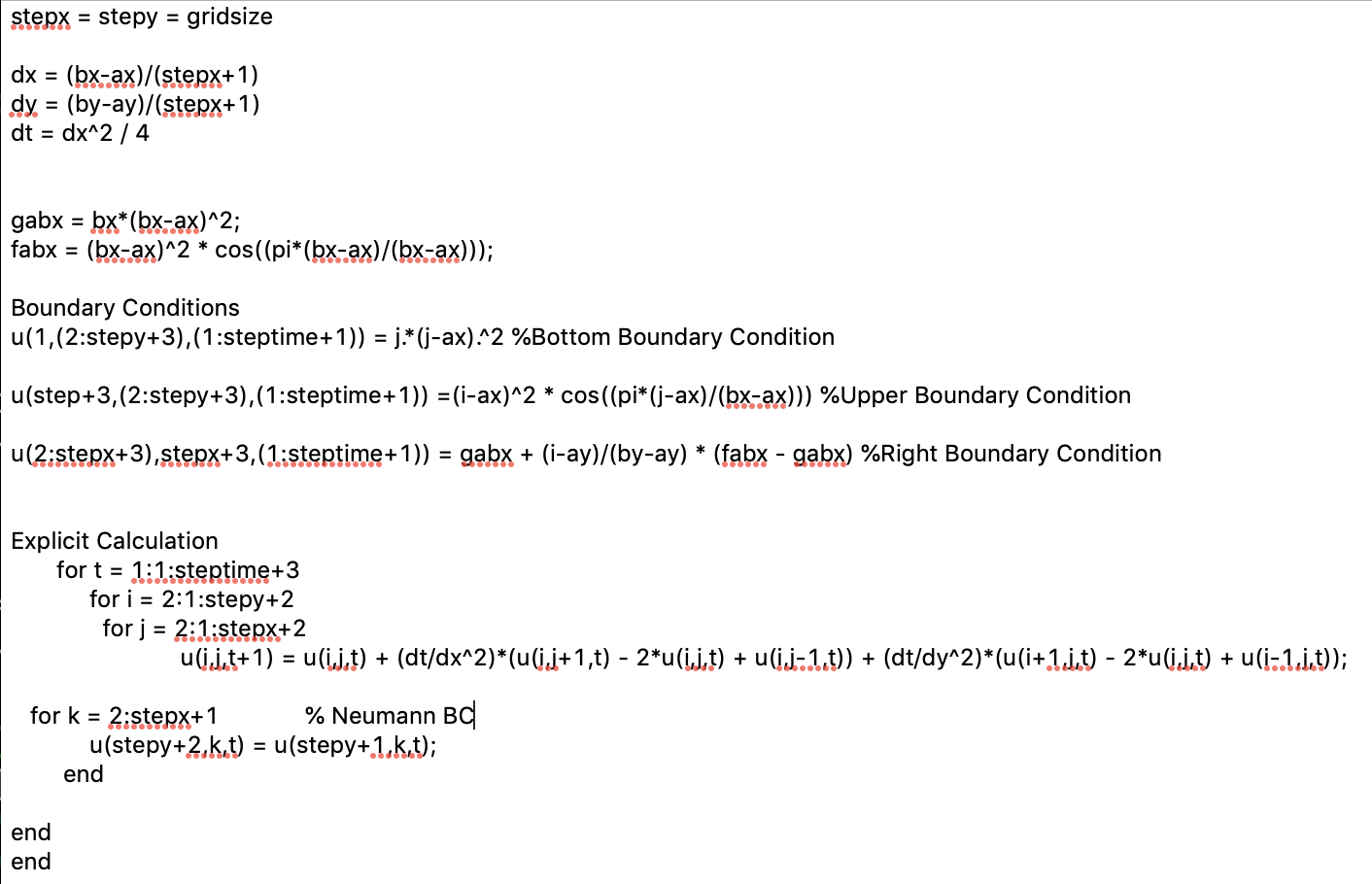


Figure : Pseudocode of explicit method

**Alternating Direction Implicit (ADI) Method**

An alternative discretization that can be used is the ADI scheme. Using the ADI scheme the discretization becomes

This is a two step implicit method. In the first step the x-discretization is set to be implicit and the y-discretization is set to be explicit. The order is reversed during the second step. This allows for the use of two tri-diagonal matrices instead of one pentadiagonal matrix. The equations above simplify to

The left hand side of the equation is solved implicitly while the right hand side is solve explicitly. A skeleton code of the ADI method is shown below:

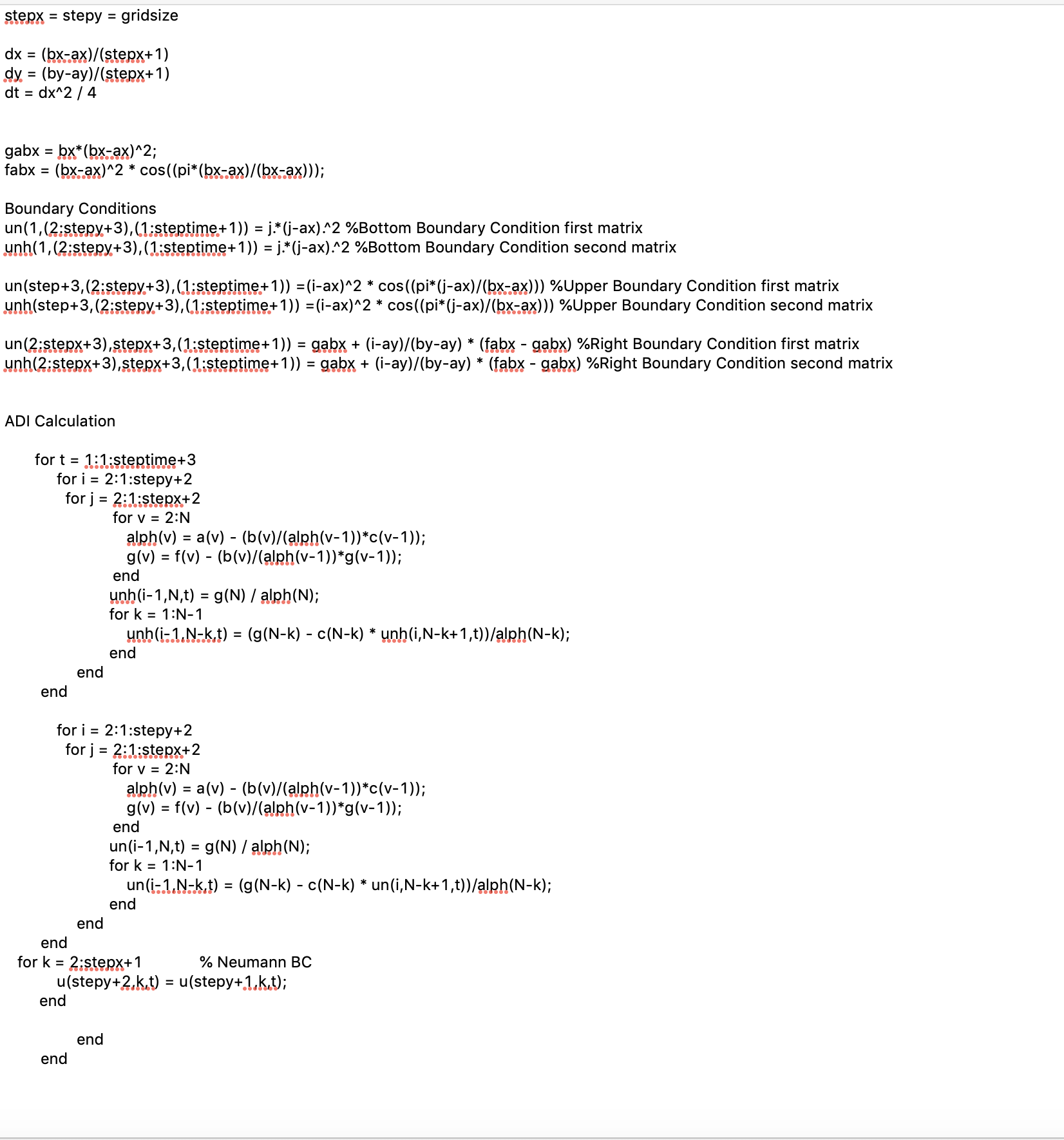


Figure : Pseudocode of ADI Method

**Computer Specifications:**

**Model:** Macbook Pro (15-inch, 2018)  
**CPU:** Core i7-8850H - Intel

**Number of Cores:** 6

**Number of Threads:** 12s

**Max CPU Clock Speed:** 2.6 GHz

**L1 Cache:** 32 KB

**L2 Cache:** 256 KB (Per core)

**L3 Cache:** 9 MB

**DRAM Size:** 16 GB 2400 MHz DDR4

**Results**

Once the code was written, several tests were done to test the accuracy of the numerical methods. The first test was verifying that the numerical solutions were lining up with manufactured solutions. The final test was a speed test between the explicit and ADI method.

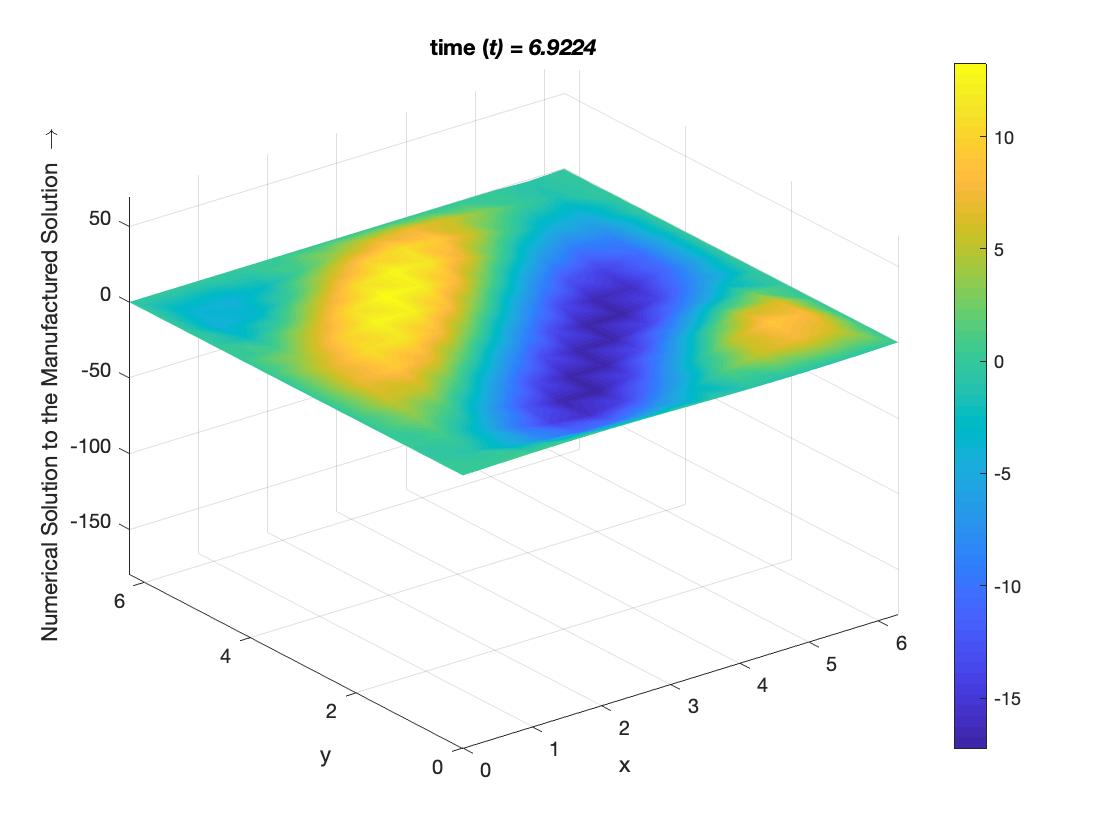
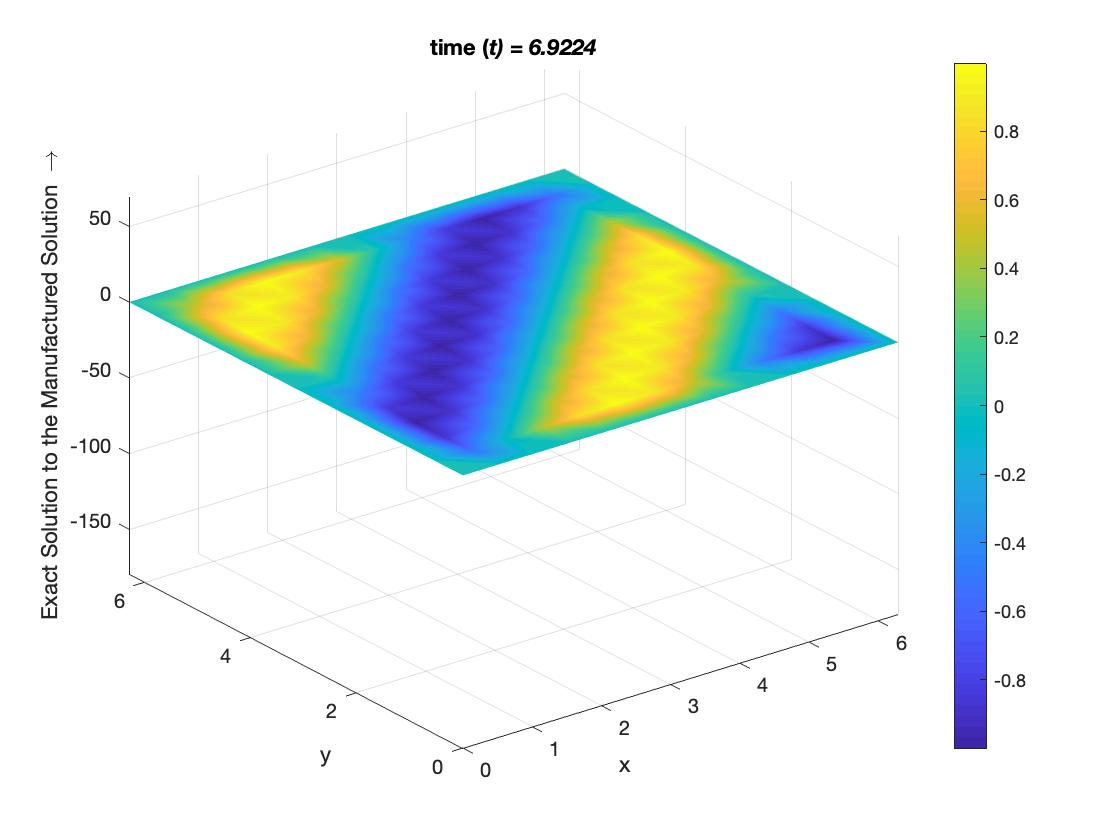
**Verification** 

Figure : Exact Solution

Manufactured solutions were used to ensure that the numerical solution derived was able to solve equations as intended. The initial manufactured solution used was

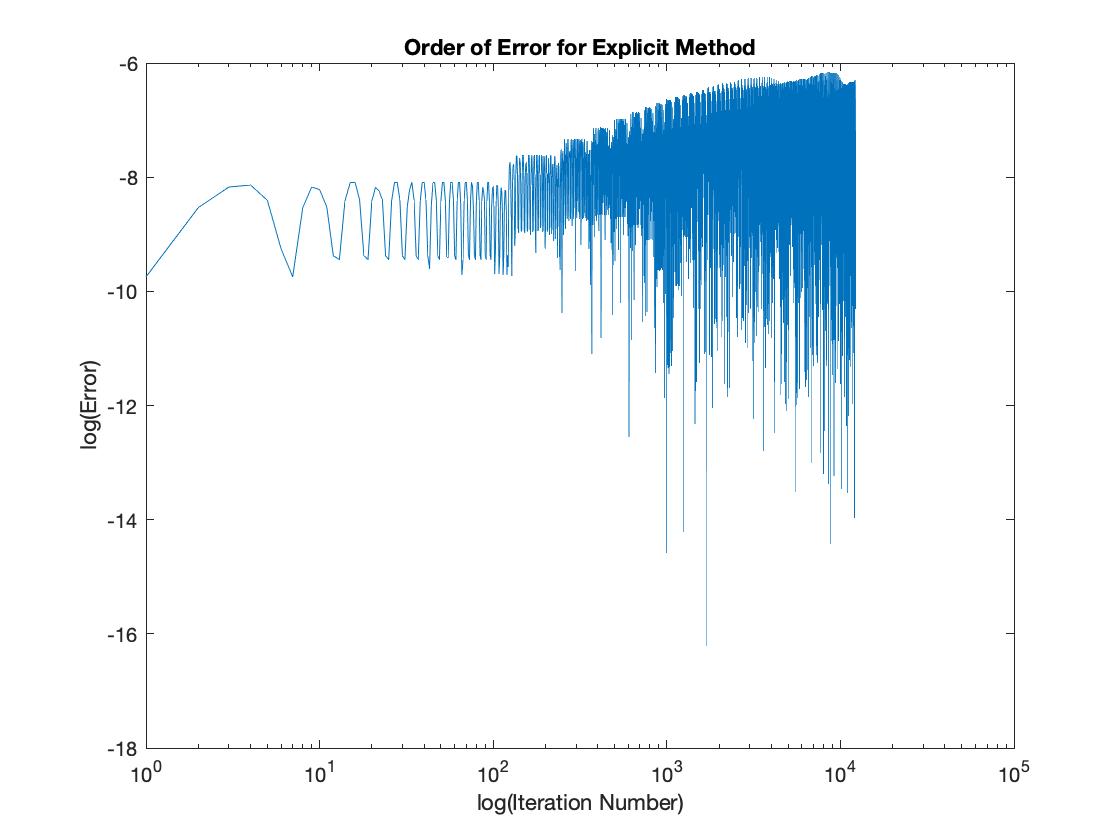
This equation was used because it was easy to implement and because it oscillates instead of continuously growing. A visualization of the original solution can be seen in the figure 3. This solution was plugged into the diffusion equation as seen below

The residual on the right side of the equation is added to numerical solver. When using the explicit method this changes the equation to:



Dirichlet boundary conditions are generated for the manufactured solution. They generated boundary conditions are

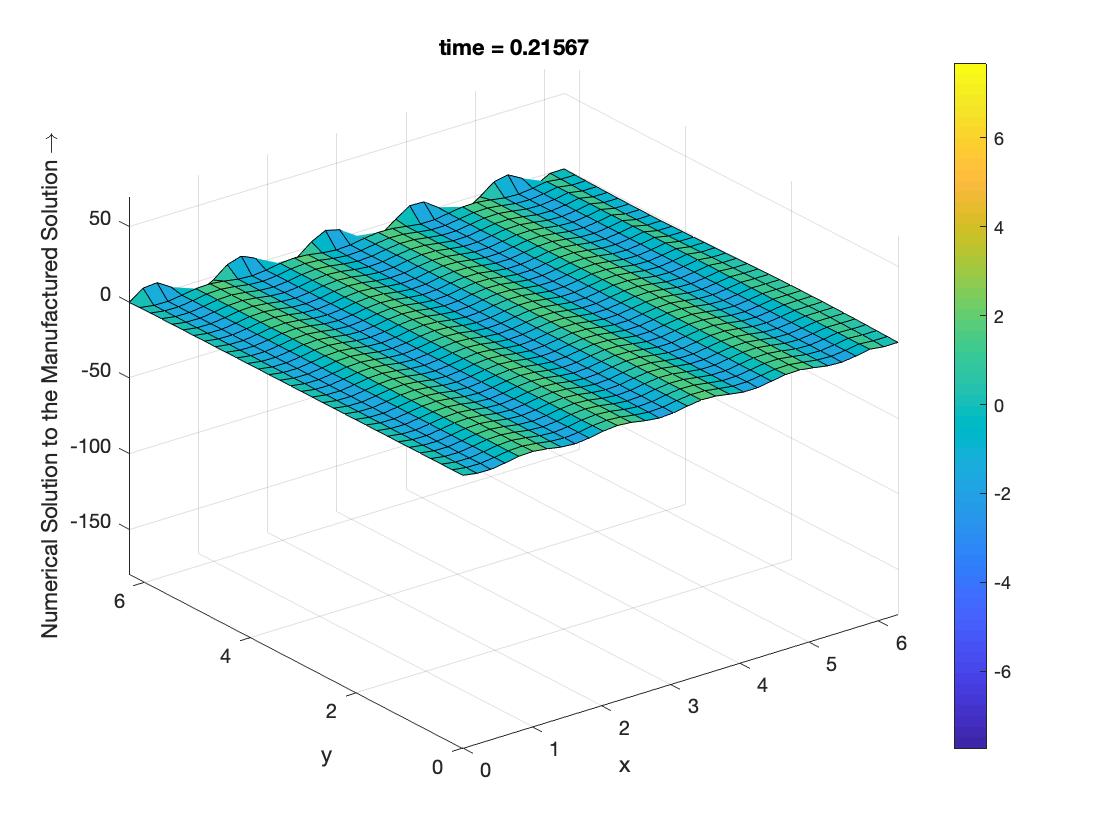
Figure : Explicit Numerical Solution



When the boundary conditions and the new explicit method are solved, the produce a remarkably similar result to the exact manufactured solution. A plot of the explicit method is shown in figure 4. It has almost the same structure and motion as the exact solution. A log-log plot of the error of the exact solution the explicit numerical solution is shown below. The error, overall is extremely low.

Figure : Error plot of numerical solution

In contrast to the explicit solution, the ADI solution does not line up with the exact solution at all. This shows that, unfortunately, the ADI method was not properly implemented.



**Timing**

The main solving loops for the explicit and ADI methods were timed using matlab’s tic toc functionality. The results are shown in the table below.

Figure : ADI Numerical Solution

|  |  |  |
| --- | --- | --- |
| **Spatial Steps** | **Explicit** | **ADI** |
| **40** | **0.155 s** | **5.5906 s** |
| **80** | **0.38 s** | **31.4429 s** |
| **100** | **1.2593 s** | **54.7537 s** |

Table 1: Timing of explicit and ADI methods with different number of spatial steps

It can be immediately seen that the ADI method takes significantly longer than the explicit method. For the case of 80 spatial steps, the explicit method finishes in 1.2% of the time it takes for the ADI method to finish. This is most likely due to the relative complexity of solving two thomas algorithms.