**Solving the 2D diffusion Equation Using Explicit and ADI methods**

**Oscar Rodriguez**

**1071756**

**MECE 5397**

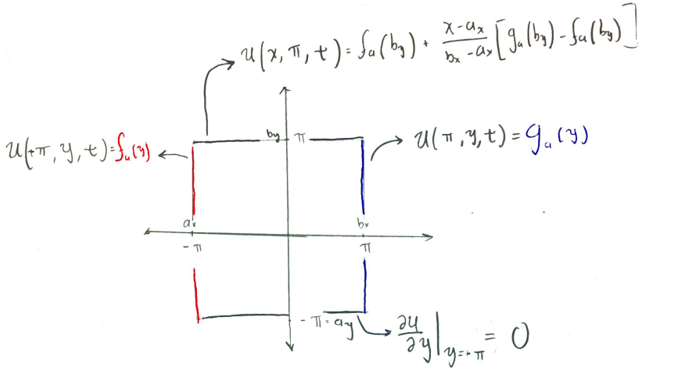
**Abstract**

The following report analyses the utilization of scientific code to solve the two-dimensional diffusion equation. The analysis consists in the solution of the differential equation using the explicit and ADI method. The report indicates all the specifications of the parameters used in the analysis, including the boundary conditions, code language used and specific hardware used to run simulations. The results will evaluate the effects of the number of iterations applied, a study of the grid convergence and the effects of diffusive CFL. The report ultimately concludes with a comparison of the expected results using the theoretical behavior. It was found that with \_\_\_\_,\_\_\_\_\_,\_\_\_\_\_ the solution of the two-dimensional diffusion equation is \_\_\_\_\_\_,\_\_\_,\_\_\_\_\_ respectively. When compared to the expected theoretical behavior it was found that the applied numerical methods had an error of \_\_\_\_,\_\_\_\_,\_\_\_ when using \_\_\_\_\_,\_\_\_\_\_,\_\_\_\_\_. It was also found that the explicit method and the ADI method each had an error of \_\_\_\_ and \_\_\_ respectively when compared to the analytical solution. It can therefore be concluded that the \_\_\_\_\_\_method is more effective than the \_\_\_\_method especially when using\_\_\_\_\_.

**Mathematical Statement of Project**

Using computer code and numerical methods, more specifically the explicit and ADI discretization methods, the following two-dimensional diffusion equation is to be solved:

|  |  |
| --- | --- |
|  |  |
| Boundary Conditions: | =0 |
|  |  |

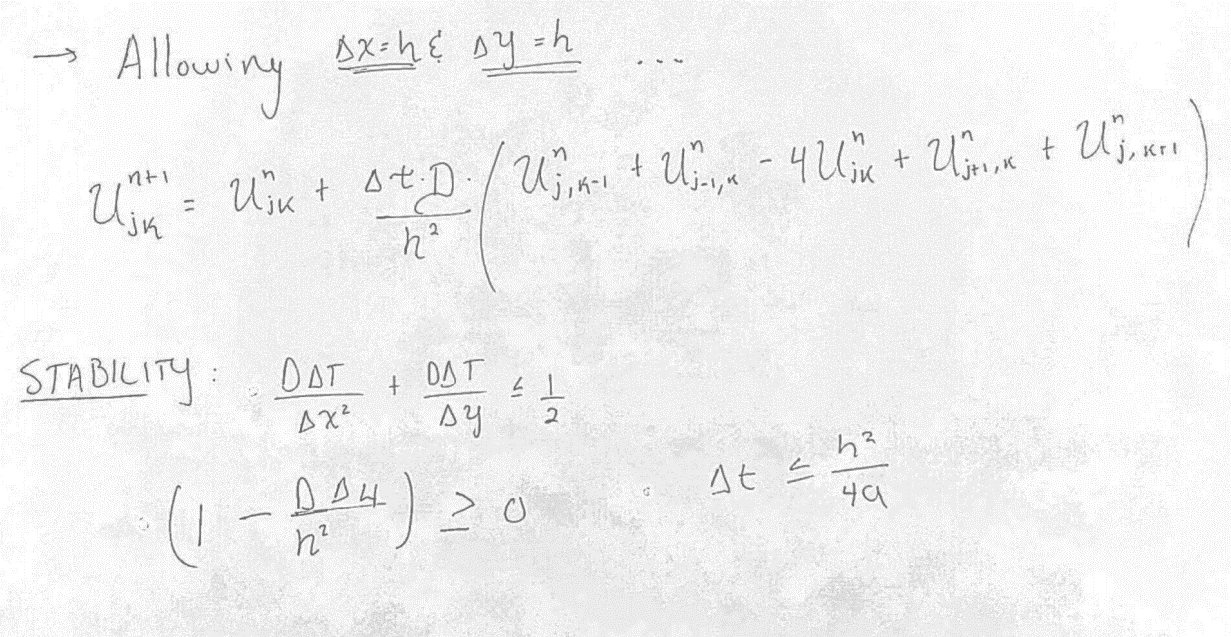


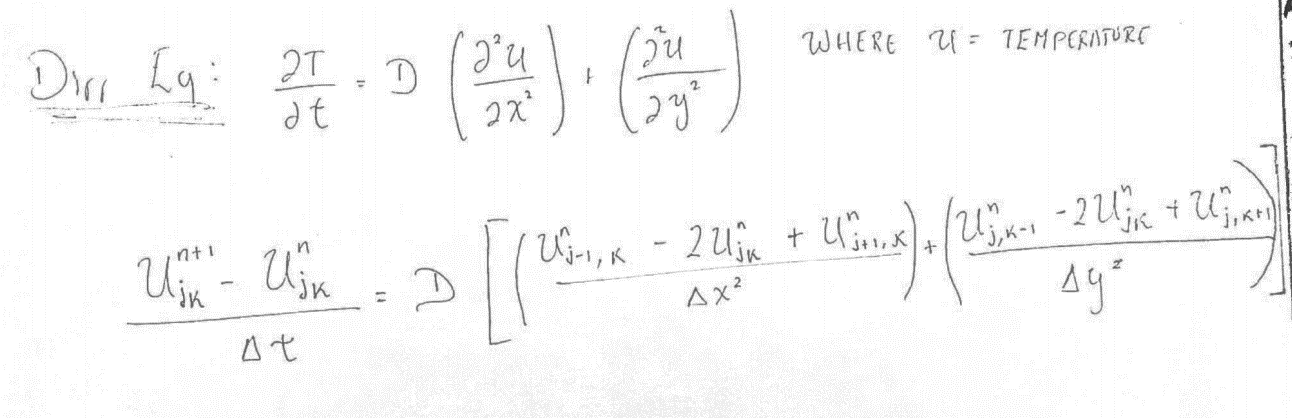
**Discretized version of Equations**

To solve the diffusion equation using computational methods it is necessary to discretize the equations so that they can be translated into a useful code capable of running in MATLAB

**Explicit Method Discretization**

The explicit method implemented in this solution method utilizes a center difference to solve both, the partial derivatibe with respect to x and the partial derivative with respect to y.

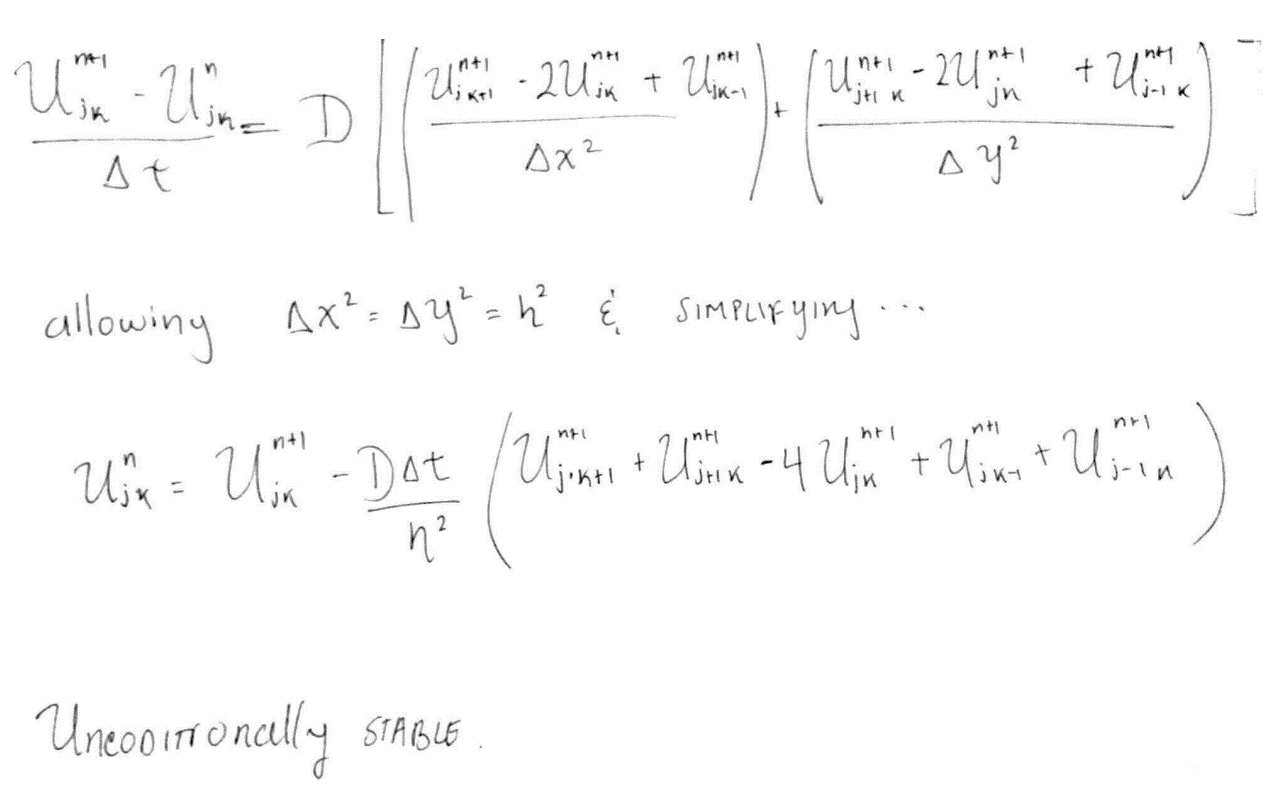
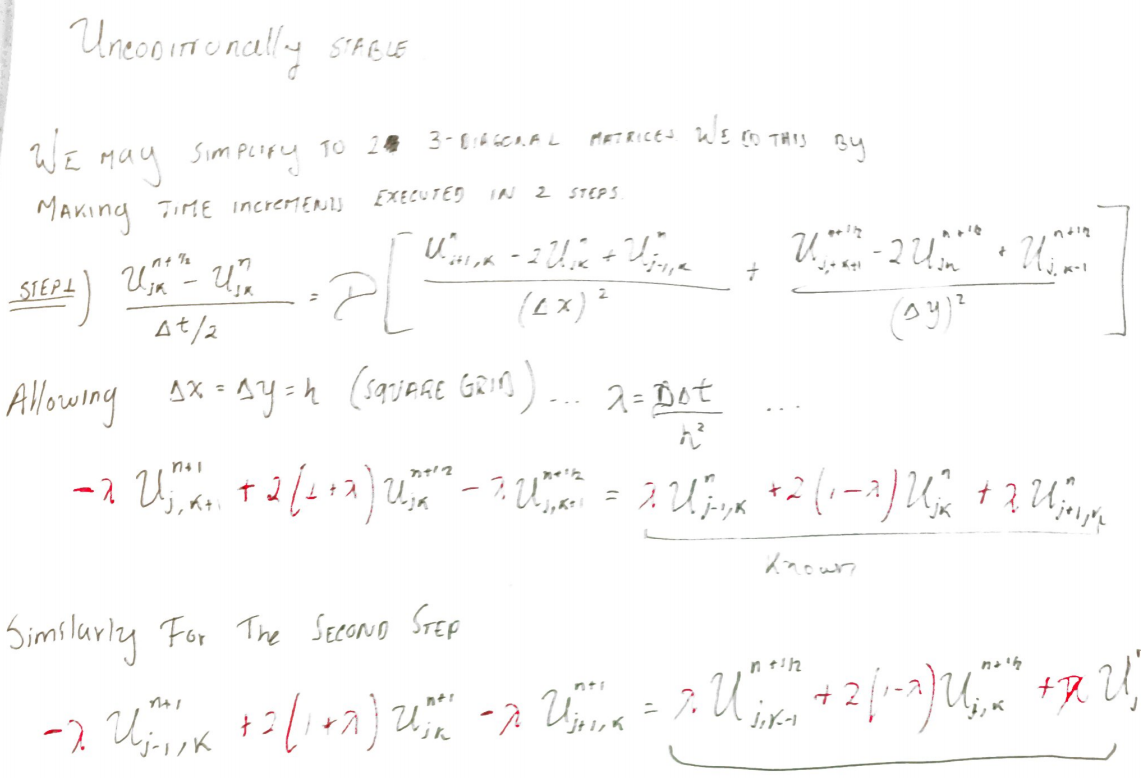




After discretizing the diffusion equation, a scheme was set up that was capable of running in a typical code language such as MATLAB.

**ADI Method Discretization**

The alternating direction implicit scheme is described below. The scheme is an approach so solving partial differential equations in a simpler way as opposed to forming a Pentadiagonal matrix.



**Description of Numerical Methods and Pseudo Code**

**Explicit Method**

The explicit method solves the differential equation by going point by point in the grid and solving for temperature (u) using a center difference approximation.

Pseudocode

% Describe all problem parameters

specify boundaries

specify time limit

specify initial time conditions

specify boundary conditions

specify diffusivity

calculate time step size

calculate space step size

calculate lambda

if % testing if the lambda is appropriate to avoid the calculation to blow up

for start time loop with limits

initiate graphing based on time step

for loop going through x

for look going through y

specify boundary conditions

end loop for x

end loop for y

end time loop

**ADI Method**

The ADI method solves the differential equation by creating 2 tridiagonal matrices that can be solved using the tridiagonal algorithm in a separate function file

*Pseudocode*

% Describe all problem parameters

*specify boundaries*

*specify time limit*

*specify initial time conditions*

*specify boundary conditions*

*specify diffusivity*

calculate time step size

calculate space step size

Define lambda to be used in the tridiagonal matrices of both half-steps

% Specify the coefficients of the 2 three diagonal matrices used in the ADI scheme

*define a1*

*define b1*

*define c1*

*define d1*

*For* Initiate time for

*Hei=sur(x,y,u)* initiate graph

*For j=2:nyy* for loop visiting columns of the fisrt half of the time step matrix

*Populate d1*

*Populate last element of 2*

*For i=2:Nxx initiate loop visit rows of the second half matrix*

*Populate d*

*%Calculate the first half using the tridiagonal function*

*%Use new right hand side based on the results of the tridiagonal*

*For J= 2:Nyy*

*Populate d*

*%based on the newly calculated right hand calculate final u matrix*

*Impalement Neuman condition*

*Calculate u using tridiagonal*

**Technical Specifications of Computer Used**

*Sony Vaio Flip 13*

* The Intel Core i5-4200U, Haswell. released in August 2013.:
* Speed: At a clock speed of 1.60 GHz,.
* Cores: It is a dual-core processor,
* Cache: Its 3MB cache
* RAM: 8GB
* Hard Drive:125GB SSD
* OS: Microsoft Windows 10
* System Type: 64-bit

**Results**

The following results demonstrate all of the calculation and data collected when running both the explicit and ADI codes to solve for the heat distribution over a period of time in a square place.

**Explicit**

From figure below it is evident that the explicit numerical solution was able to map the heat distribution over time somewhat accurately. The blue in the figure represents lower temperature while higher temperatures are represented by yellow

****

****

**The following table shows the data collected from the performance of the explicit code**

|  |  |  |
| --- | --- | --- |
| Mesh size (N) | Number of Iterations | Running Time |
| 4 | 22 | 0.022970 |
| 8 | 102 | 0.082787 |
| 16 | 432 | 0.315920 |
| 32 | 1,956 | 2.568503 |
| 64 | 7,890 | 13.475713 |
| 128 | 35,745 | 70.703948 |
| 256 | 123,659 | 575.944671 |
| 512 | - | Excessive Time required |

**ADI**

From figure below it is evident that the ADI numerical solution was able to map the heat distribution in a more defined way compared to the explicit . The blue in the figure represents lower temperature while higher temperatures are represented by yellow

****