Name: Swithin Stephen

Class: MECE 5397

Diffusion Equation Project

Dr. Prosperetti

Dr. Amritkar

Abstract

The project deals with discretizing a diffusion equation https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial%20u%7D%7B%5Cpartial%20t%7D%20%3D%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20x%5E2%7D&plus;%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20y%5E2%7D. To discretize this equation, a process called finite difference method is employed. There are different schemes that could be used to discretize this equation, for example: explicit method, implicit method, Runge-Kutta , ADI (Alternating direction implicit method) , and Crank- Nicholson method. In this project the explicit and implicit are used to discretize this equation and will be written in MATLAB. The explicit or FTCS (Forward in time and centered in space) have parameters that are calculated based on previous levels, and ADI (Alternating direction implicit method) where it is explicitly solved for the y-axis and implicitly solved for the x-axis and for the second step it is explicit for the x-axis and implicit for the y-axis. Along this project, there will several steps that will done to optimize the overall efficiency of the code such as vectorization of the code, array pre-allocation etc. There also will be debugging steps such as check pointing that will be used in this project.

Mathematical statement of the problem

Given a diffusion equation;

https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial%20u%7D%7B%5Cpartial%20t%7D%20%3D%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20x%5E2%7D&plus;%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20y%5E2%7D

Domain of interest of rectangle;

https://latex.codecogs.com/gif.latex?a_%7Bx%7D%3Cx%3Cb_%7Bx%7D https://latex.codecogs.com/gif.latex?a_y%3Cy%3Cb_y

With boundary condtions;

https://latex.codecogs.com/gif.latex?u%28x%2C%20y%20%3D%20b_%7By%7D%29%3Df_%7Bb%7D%28x%29 https://latex.codecogs.com/gif.latex?u%28x%2C%20y%20%3D%20a_%7By%7D%29%3Dg_%7Bb%7D%28x%29

https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial%20u%7D%7B%5Cpartial%20x%7D%20%5Cmid%20x%3Db_%7Bx%20%3D%200%7D https://latex.codecogs.com/gif.latex?u%28x%2Cy%3D%20a_%7By%7D%29%3D%20g_%7Bb%7D%28a_%7Bx%7D%29&plus;%5Cfrac%7By-a_%7By%7D%7D%7Bb_%7By%7D-a_%7By%7D%7D%5Bf_%7Bb%7D%28a_%7Bx%7D%29-g_%7Bb%7D%28a_%7Bx%7D%29%5D

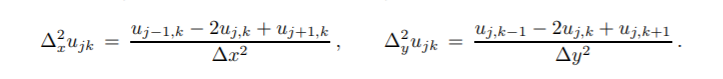
https://latex.codecogs.com/gif.latex?a_%7Bx%7D%3Da_%7By%7D%3D-%5Cpi https://latex.codecogs.com/gif.latex?b_%7Bx%7D%3Db_%7By%7D%3D-%5Cpi

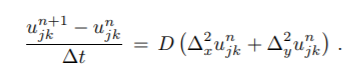
https://latex.codecogs.com/gif.latex?g_b%28x%29%3D%28b_x-x%29%5E2cos%5Cfrac%7B%5Cpi%20x%7D%7Bb_x%7D https://latex.codecogs.com/gif.latex?f_b%28x%29%3Dx%28b_x-x%29%5E2

Discretization of the equations

The diffusion equation is discretized explicitly by;

https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial%20u%7D%7B%5Cpartial%20t%7D%20%3D%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20x%5E2%7D&plus;%20%5Cfrac%7B%5Cpartial%5E2%20u%7D%7B%5Cpartial%20y%5E2%7D





Where D=1 in our case, Ujkn+1 is the one we are solving for.

Pseudocode:

Uj,kn+1= Uj,kn + ∆t\*D\*((Uj,k-1n + Uj-1,kn – 4Uj,kn + Uj+1,kn + Uj,k+1n)/(h^2))

while error>Tol

Tc= Tn;

t = t+dt;

for i=2:Nx-1

for j= 2:Ny-1

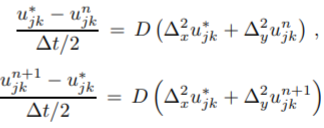
Tn(j,i)=Tc(j,i)+ dt\*((Tc(j,i+1)+Tc(j+1,i)-4\*Tc(j,i)+Tc(j,i-1))+Tc(j-1,i))/dx/dx;

%dx/dx is faster than dx^2 which is an optimization method

end

end

ADI scheme is employed using this discretization;



ADI is solved using a tri-diagonal solver where it is implicitly solves in one grid direction and for the next step it is done the same way in another grid direction.

Pseudocode:

for i=2:nx-1;

for j=1:nx;

% r(i) is the explicit solution in x for the second half time step

r2(j)=a\*U(i-1,j)+c\*U(i,j)+a\*U(i+1,j);

end

% The implicit solution in y is determined using the tridiagonal equations

% solver in which the subdiagonal, diagonal, and superdiagonal where

% determined form discretization and the Neumann boundary conditions for

% y

x2=tridiag(e2,f2,g2,r2);

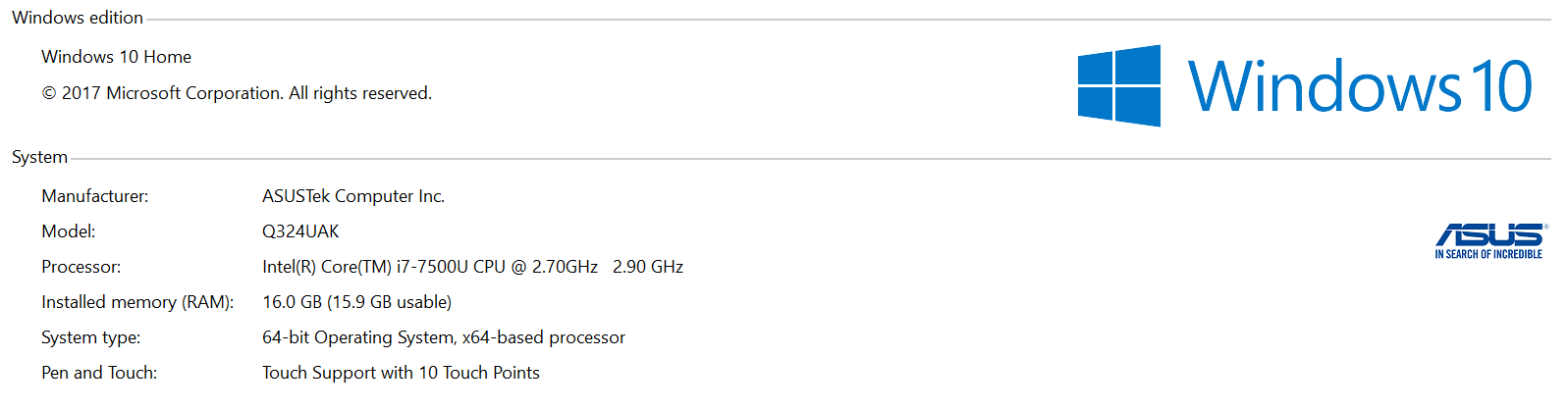
% Finally u(n+1) is determined by the combination of the explicit

% solution in x and the implicit solution in x

U(i,:)=x2;

End

Technical specification of computer



Results

The two methods used in the project was Explicit method and ADI method. Both methods used employs the usage of diffusive CFL in its code. The effect of CFL number determines the stability of numerical scheme. As common, if they reach some critical limit, the numerical solution begins to oscillate in space and time with grid period. That oscillatory solution is non-physical and grows rapidly providing the numerical overflow. The parameters used in the code are the boundary conditions, the initial conditions and CFL number that was implemented in the code, D or diffusivity constant was taken to be 1. For the verification process, the explicit method is solved up-to an error of 1\*10-6 which can be seen in Figure1.

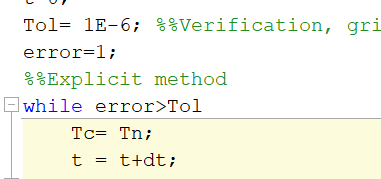


Figure 1: Error verification

The ADI scheme is verified by using a straight-line convergence test which is shown in Figure 2

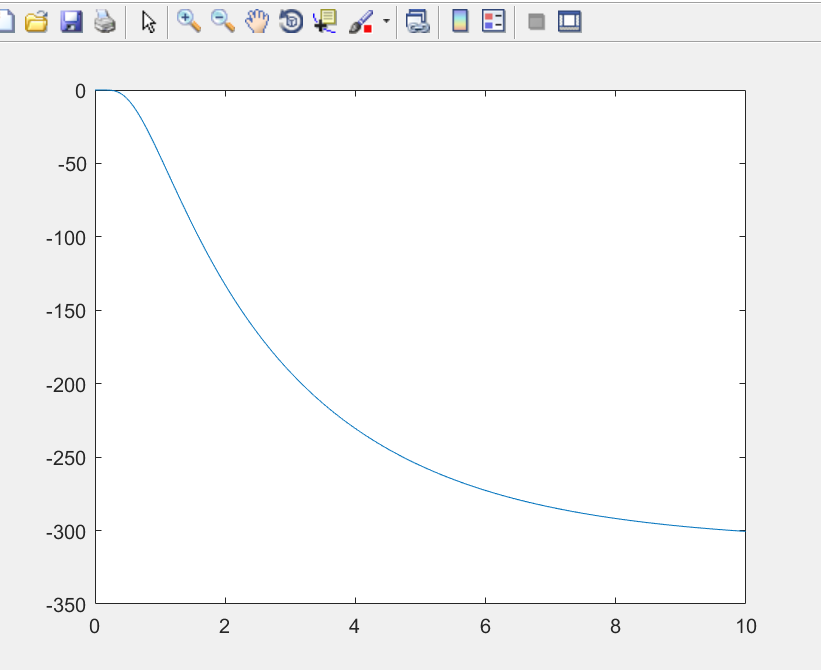


Figure 2: Straight Line Test for Convergence (ADI scheme)

The number of points used in the code helps it to create a finer mesh to output the results. In this project for the explicit scheme the number of points used was N= 50 to get a better mesh and for the ADI scheme the number of points used was also 50 to output the same results.

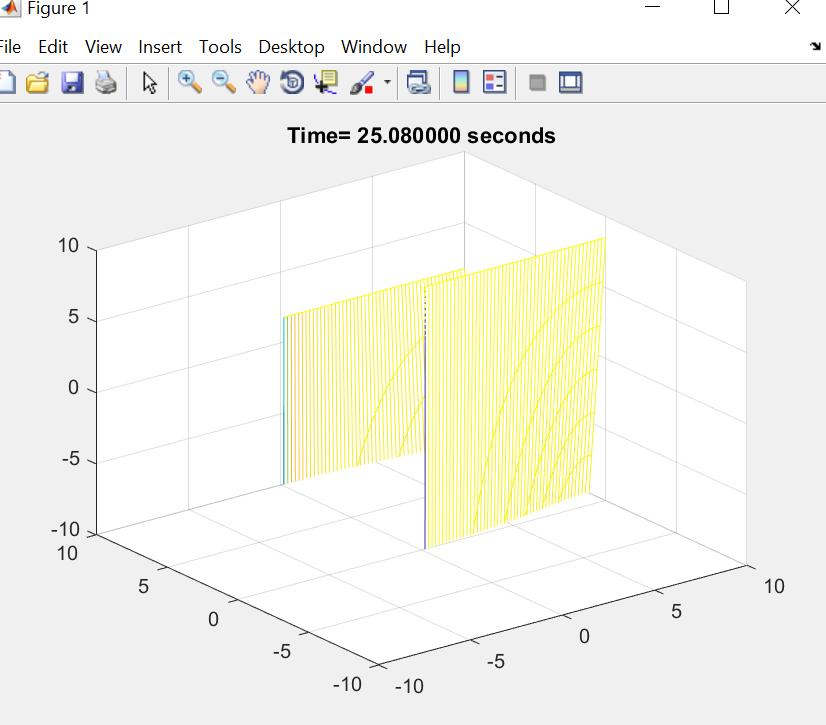


Figure 3: Diffusion using Explicit Method

The figure above shows the amount of diffusion when it reaches 25.08 seconds. For the explicit scheme, it converged at ~ 98 seconds and Figure 3 shows the steady state as the temperature changes with respect to x and y as it converges.

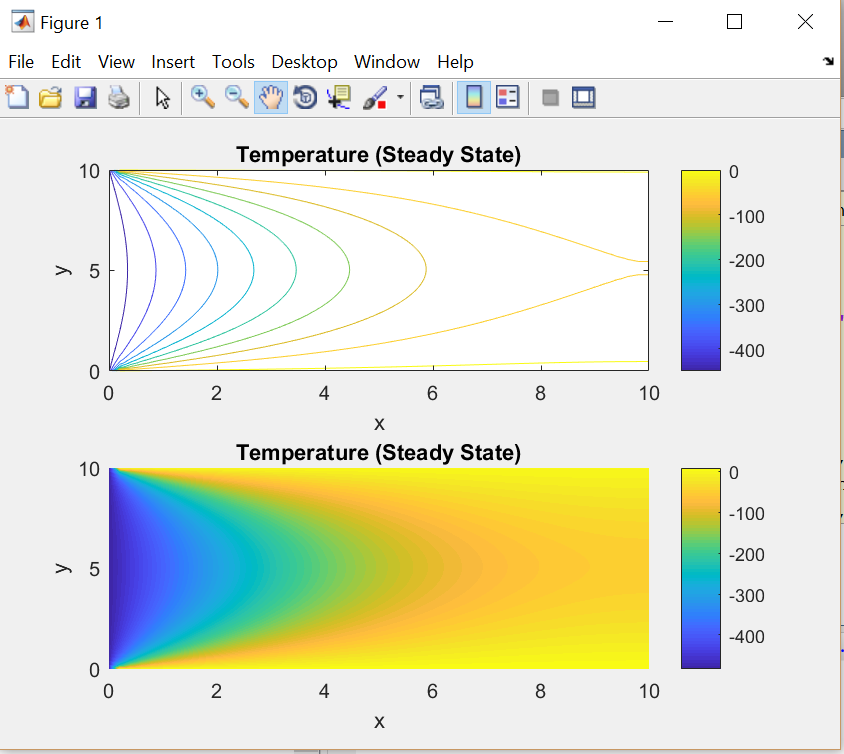


Figure 4: Steady state using Explicit method

It was seen in these codes that the expected behavior for fine meshes have roundoff errors while discretization error was for large step sizes. In this project the implementation of Neumann conditions in the ADI scheme was hard to implement. Due to this reason, Figure 5 has a lot of variance from the diffusion plot of the explicit scheme.

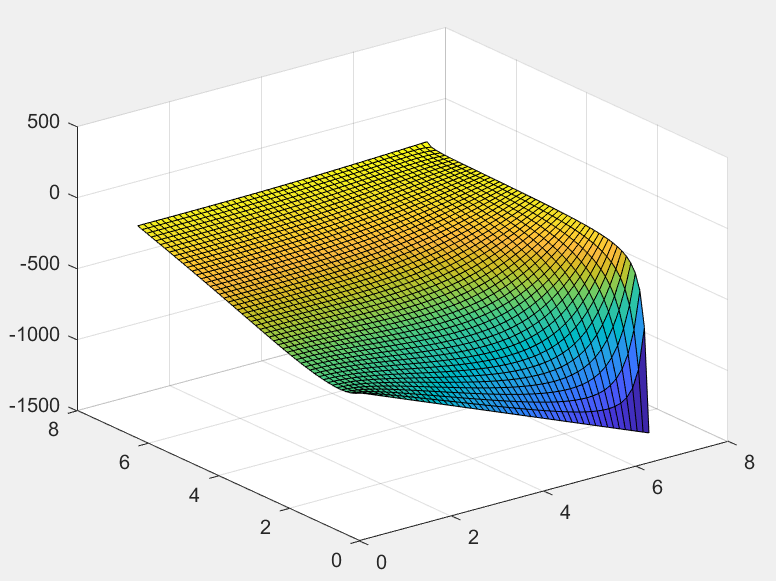


Figure 5: Diffusion using ADI scheme