MECH 587 PROJECT 2

Submitted to: Dr. Rajeev Jaiman

Christian Rowsell (40131393)

Contents

Table of Figures

Part 1: Unsteady Advection Equation

1. Exact Solution

a) Consider 1D advection equation $\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0$. How to classify it? How the solution behaves for this class of PDE?

This PDE will be classified as a hyperbolic equation. Therefore, this PDE will have a wave-like behaviour for the solution.

a) Plot the initial contour of ϕ and velocity vector (u, v) with Matlab. Then convert the velocity vector from Cartesian coordinate system (u, v) to polar coordinate system (vr, vθ). Together with the answer of 1(a), can you write down the exact solution of the current problem?

The following plots were generated using Python with the Matplotlib package. This package allows plots to be generated in a very similar manner to MATLAB. 4 Plots were generated. The first and second plots show the velocity contours for u, and v respectively, in Cartesian coordinates. The 3rd plot shows the total magnitude of the velocity vector in Cartesian coordinates. Finally, the last plot shows the magnitude of the velocity vector using the polar coordinate system. The following code was used to generate these plots.

```
# Import Packages
import numpy as np
import matplotlib.pyplot as plt
# Initialize
num_div = 100  # Number of divisions
x_list = np.linspace(-1.0, 1.0, num_div)  # x coordinates
y_list = np.linspace(-1.0, 1.0, num_div)  # y coordinates
# Polar coordinates
x, y = np.meshgrid(x_list, y_list)  # Mesh
phi = 5.0 * np.exp(-1500.0 * ((x - 0.25) ** 2 + y ** 2))  # Initial phi
# Magnitude
u = y
v = -x
r = np.sqrt(u ** 2 + v ** 2)
theta = np.arctan(v/u)
# plt.contourf(x_list, y_list, u)
# plt.contourf(x_list, y_list, v)
# plt.contourf(x_list, y_list, r)
plt.contourf(x_list, y_list, theta)
# Adding
velocity = -x + y
# plt.contourf(x_list, y_list, velocity)

# Show plot
plt.xlabel('X')
plt.ylabel('Y')
plt.title('theta Velocity')
plt.colorbar()
plt.show()
```

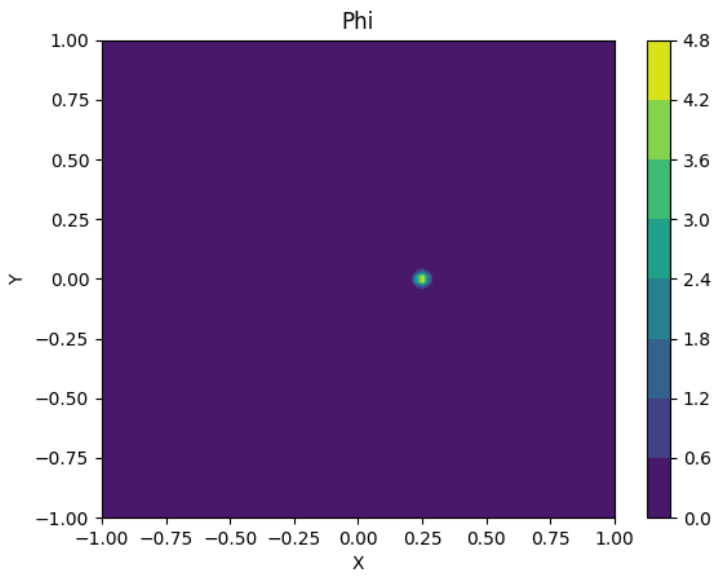


Figure 1: Phi Initial Condition Contour

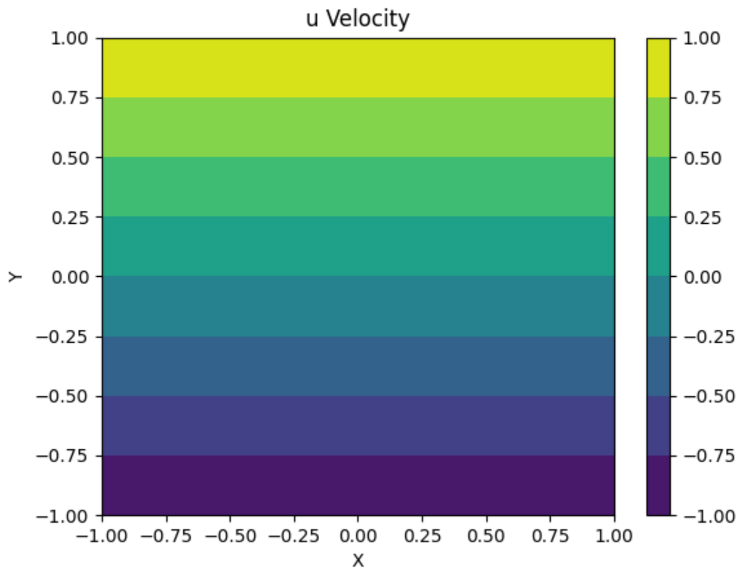


Figure 2: u Velocity Contour

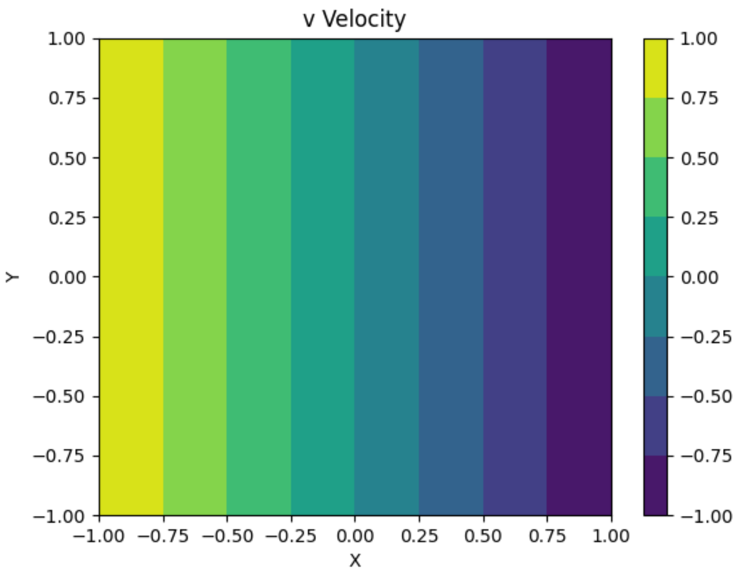


Figure 3: v Velocity Contour

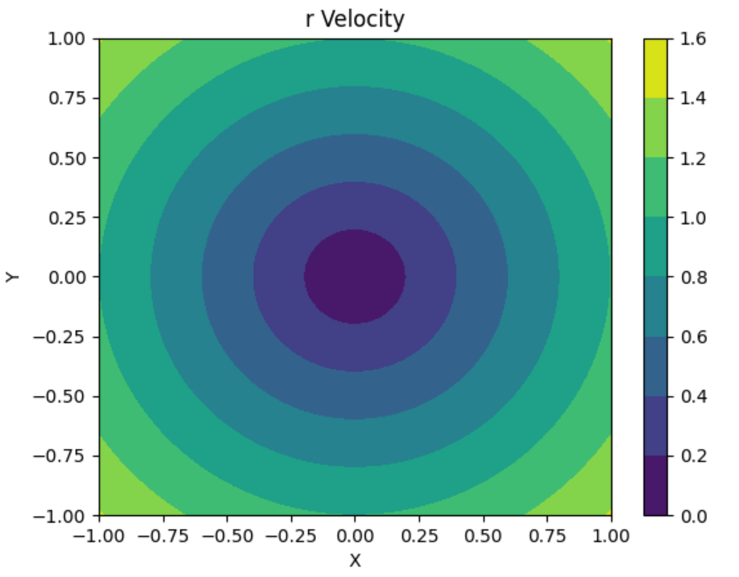


Figure 4: r Velocity Contour

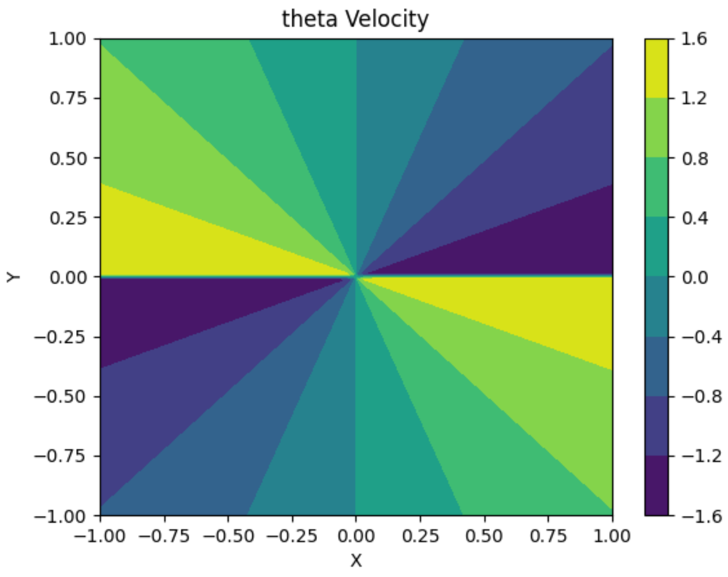


Figure 5: theta Velocity Contour

Looking at the above plots we can see that the velocity is going in a circle. Therefore, the best guess for the exact solution is that it will repeat a certain time has passed.

2. First-order Spatial and Temporal Discretization

a) Write down the first-order upwind discretization for the convection term. Complete the corresponding function which takes the solution vector Φ and calculate the vector $-u\frac{\partial\Phi}{\partial x} - v\frac{\partial\Phi}{\partial y}$

The first order upwind convection scheme derivation can be found in the attached handwritten appendix. The function to calculate this vector is as follows:

```
void FOU(Vector &fc_Curr, const Vector &phi, const Vector &u, const Vector
&v, const Grid &G)
{
   unsigned long i, j;
   const double dx = G.dx();
   const double dy = G.dy();

    // Describe interior nodes
   for(i = 1; i < G.Nx() - 1; i++)
      for(j = 1; j < G.Ny() - 1; j++)
      {
         // This one doesnt work because abs returns an int
         /*fc_Curr(i, j) = (u(i, j) / (2 * dx)) * (phi(i + 1, j) - phi(i - 1,
j)) -
                   (abs(u(i, j)) / (2 * dx)) * (phi(i + 1, j) - 2 * phi(i, j) +
phi(i - 1, j)) +
                   (v(i, j) / (2 * dy)) * (phi(i, j + 1) - phi(i, j - 1)) -
                   (abs(v(i, j)) / 2 * dy) * (phi(i, j + 1) - 2 * phi(i, j) +
phi(i, j - 1));
          */
          //This one works because abs returns an int value
          fc_Curr(i, j) = (-0.5 * sqrt(u(i, j) * u(i, j)) / dx) * (phi(i + 1,
j) - 2 * phi(i, j) + phi(i - 1, j)) +
                        (0.5 * u(i, j) / dx) * (phi(i + 1, j) - phi(i - 1, j)) -
                        (0.5 * sqrt(v(i, j) * v(i, j)) / dy) * (phi(i, j + 1) -
2 * phi(i, j) + phi(i, j - 1)) +
                        (0.5 * v(i, j) / dy) * (phi(i, j + 1) - phi(i, j - 1));

       }
}
```

b) Use the forward Euler method to perform time integration. Calculate the solution at t=2π with an isotropic Cartesian grid Δx=Δy=h, with 201 grid points in along each direction. What is the largest possible time step you can use? Report the CFL number. (Take convective velocity as c=1)

The code to implement for forward Euler is as follows:

```
void eulerExp(Vector &phi, const Vector &fc_Curr, double &dt)
{
   phi = phi - (dt * fc_Curr);
}
```

A for loop was added to the main function in the code, which allowed batches of simulations to be run. This allowed for numerical experimentation to find the maximum timestep allowable. Simulations were run for CFL numbers ranging from 0.2 to 0.8. The solution diverged at a maximum timestep of 0.00793, with a CFL number of 0.793. At CFL 0.792, with a timestep of 0.00792, the solution converged.

c) Plot the contour of Φ.

The following contour was generated using 201 grid points, and a timestep of 0.00792.

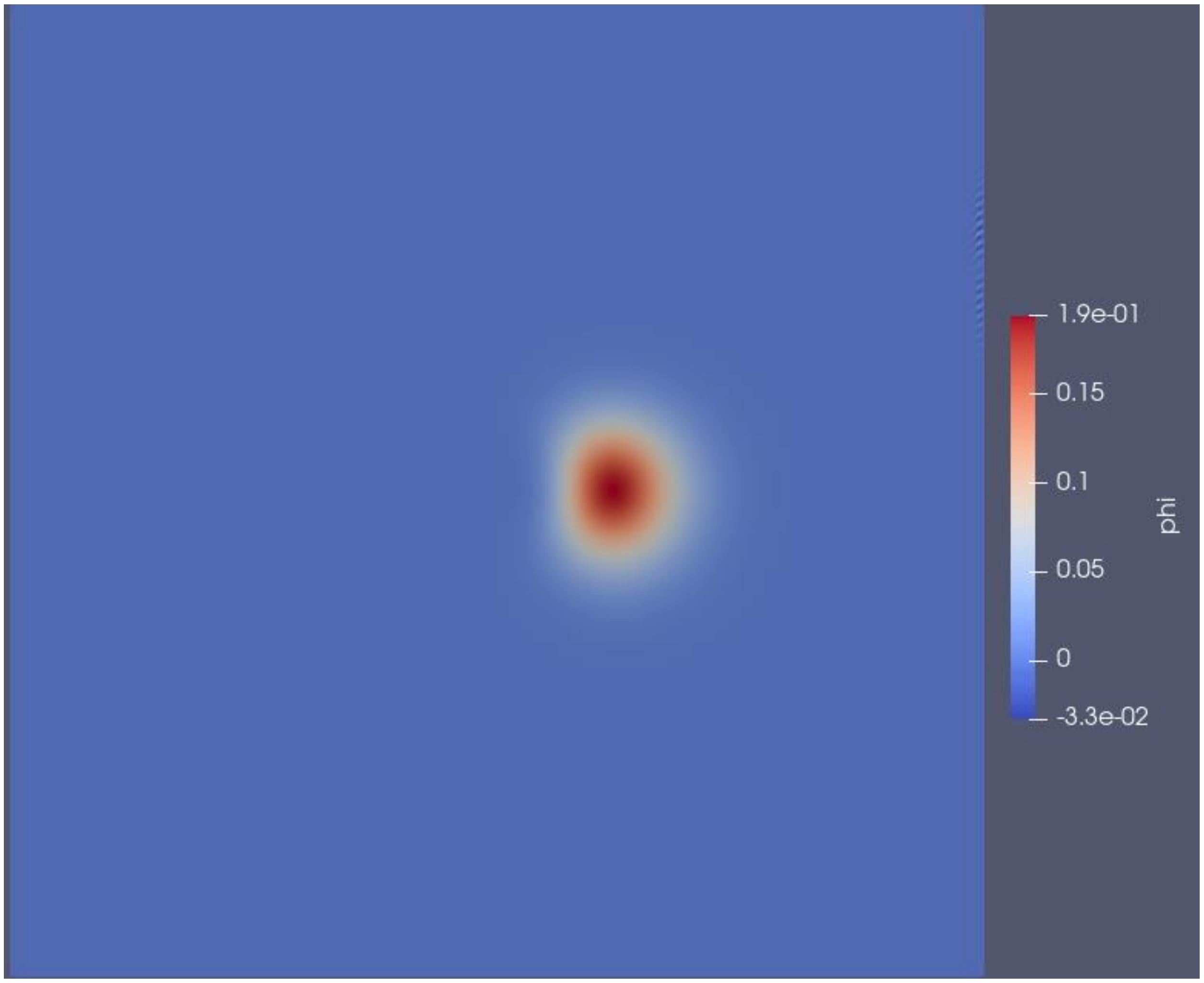


Figure 6: First-order Spatial and Temporal Contour

3. Higher-Order Discretization

a) Discretize the convection term with 2nd-order upwind scheme. For the points close to the boundary, use the central difference method.

The derivation for the 2nd-order upwind scheme can be found in the appendix with the handwritten derivations. The function to calculate this scheme is as follows:

```
void
SOU(Vector & fc_Curr, const
Vector & phi, const
Vector & u, const
Vector & v, const
Grid & G)
void SOU(Vector & fc_Curr, const Vector & phi, const Vector & u, const Vector
& v, const Grid & G)
{
unsigned long i, j;
const double dx = G.dx();
const double dy = G.dy();
// Interior points only
for (i = 1; i < G.Nx() - 1; i++)
    for (j = 1; j < G.Ny() - 1; j++)
    {
        // Central Difference for points near boundary
        if (i == 1 | | i == G.Nx() - 2 | | j == 1 | | j == G.Ny() - 2)
            {
                fc_Curr(i, j) = (0.5 * u(i, j) / dx) * (phi(i + 1, j) - phi(i
- 1, j)) +
                (0.5 * v(i, j) / dx) * (phi(i, j + 1) - phi(i, j - 1));
            }
        // Second Order Upwind for Interior Points
        else
            {
                fc_Curr(i, j) = (0.25 * u(i, j) / dx) * (-4 * phi(i - 1, j) +
phi(i - 2, j) + 4 * phi(i + 1, j) - phi(i + 2, j)) +
                (0.25 * sqrt(((u(i, j)) / dx) * ((u(i, j)) / dx))) * (6 *
phi(i, j) - 4 * phi(i - 1, j) + phi(i - 2, j) - 4 * phi(i + 1, j) + phi(i +
2, j)) +
                (0.25 * v(i, j) / dy) * (-4 * phi(i, j - 1) + phi(i, j - 2) +
4 * phi(i, j + 1) - phi(i, j + 2)) +
                (0.25 * sqrt(((v(i, j)) / dy) * ((v(i, j)) / dy))) * (6 *
phi(i, j) - 4 * phi(i, j - 1) + phi(i, j - 2) - 4 * phi(i, j + 1) + phi(i, j
+ 2));
            }
    }
}
```

b) Use the second order Adams-Bashforth method to perform time integration. Calculate the solution at t=2π with an isotropic Cartesian grid Δx=Δy=h, with 201 grid points in along each direction. What is the largest possible time step you can use? Report the CFL number. (Take convective velocity as c=1)

The code to implement the second order Adams-Bashforth method is as follows:

```
void abs2Exp(Vector &phi, const Vector &fc_Curr, const Vector &fc_Prev,
double &dt)
{
   phi = phi - dt * (1.5 * fc_Curr - 0.5 * fc_Prev);
}
```

Again, a for loop was added to the main function to allow batches of simulations to be ran. This allowed for easier determination of the maximum timestep using numerical experimentation. Simulations were ran for CFL numbers corresponding from 0.01 to 0.3. The solution began to diverge at a CFL number of 0.195, or a timestep of 0.00195. The highest simulation that converged was a CFL of 0.19, which corresponds to a timestep of 0.0019.

c) Plot the contour of Φ.

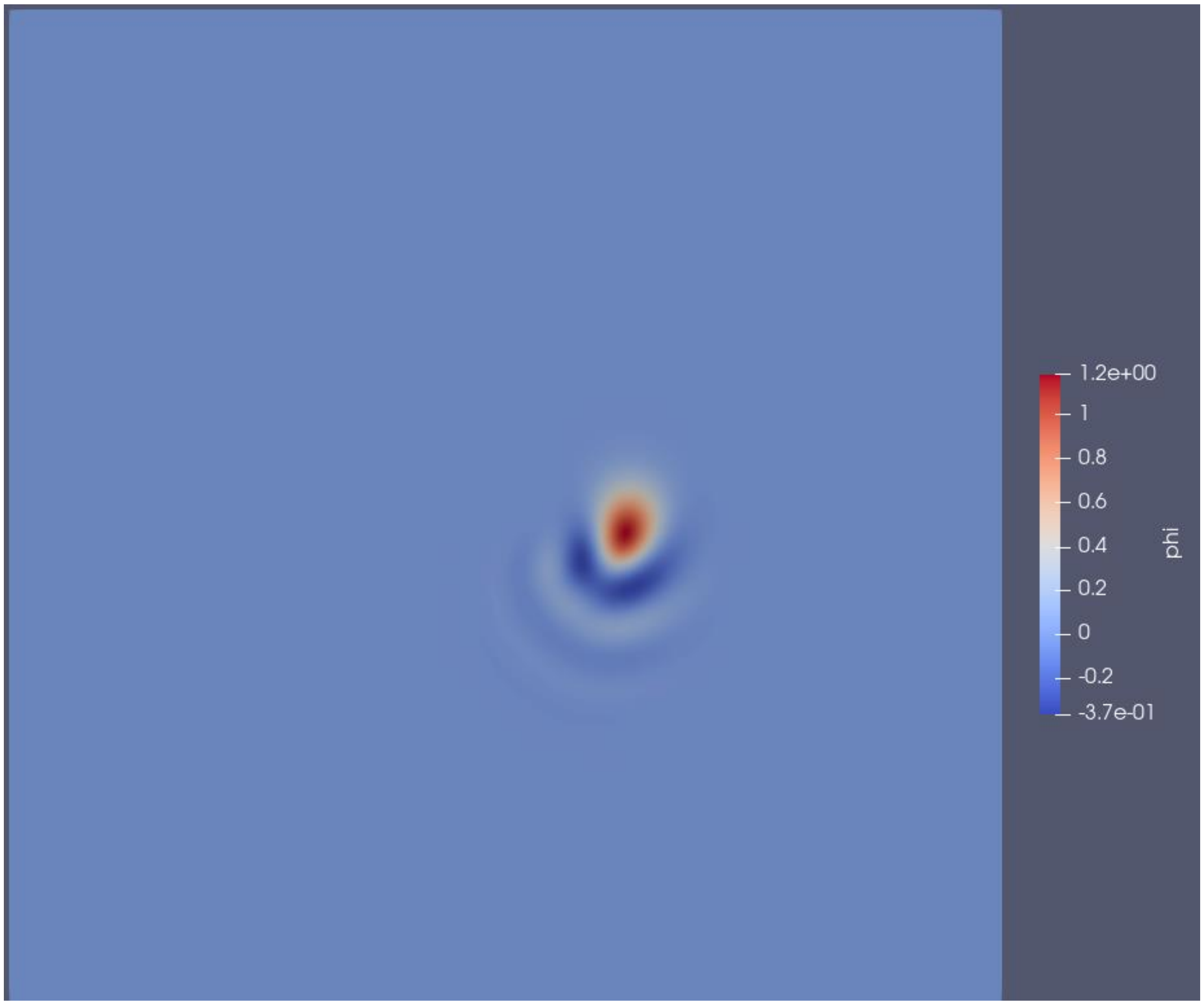


Figure 7: Higher-order Spatial and Temporal Contour

4. Comparison of Schemes

a) Compare the first order upwind + FE and second order upwind + FE. Which one allows larger Δt? Which one gives you a better solution in terms of accuracy (ϕ_{max} and ϕ_{min})? Plot the contours of ϕ. Which one is more diffusive thus giving less oscillation? Comment based on numerical stability, accuracy and diffusion. (No derivation is needed)

When running the second order upwind scheme with Forward Euler, the solution began to diverge at a CFL number of 0.25. The highest CFL tested which didn't diverge was 0.249. The following plot was obtained.

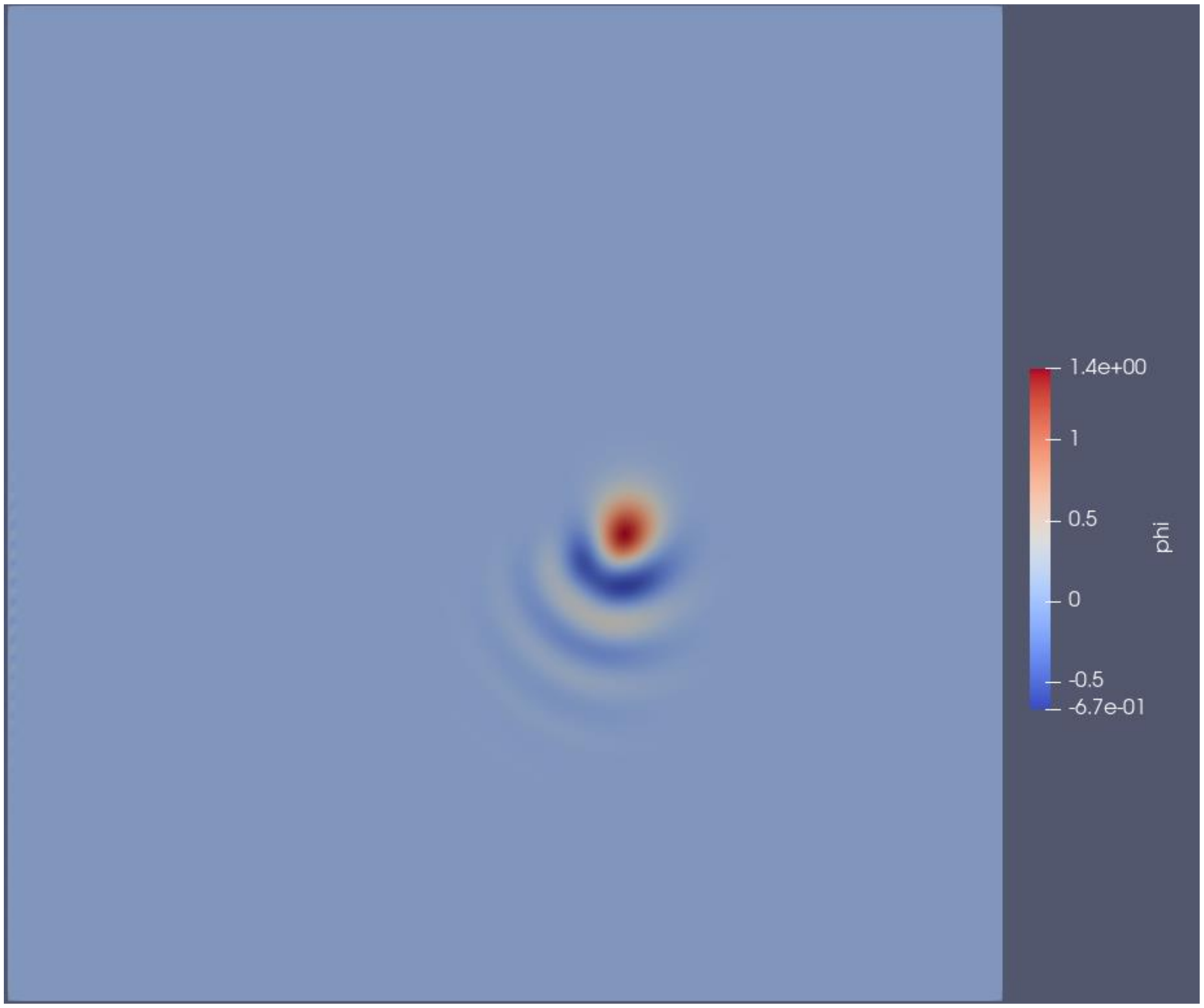


Figure 8: 2nd Order Upwind with FE Contour

The first order upwind scheme converges with a larger timestep, corresponding to a CFL number of 0.792, than the second order upwind which had a timestep corresponding with CFL number 0.249. However, the second order upwind solution provides a more accurate solution. This can be seen by comparing Figure 8 with Figure 6. In Figure 6, the solution is diffused more, and the wave-like solution is not picked up. Instead, a very large cone, that is diffused out very far can be found. However, in Figure 8, the solution appears to be diffused, however the wave-like part can be seen diffusing away from the initial cone.

b) According to what you learnt from class, comment on the stability of the central difference method for convection + forward Euler for time integration (No derivation is needed). Try the central difference method for convection + AB2. Is it stable or not? What is the largest possible time step you can use? Report the CFL number. Which method you have learnt from class might be helpful for explaining this? Just name one or few and explain why.

The central difference method for convection using forward Euler for time integration is unstable unless the CFL number is very small. CFL numbers from 0.01 to 0.5 were checked, and the central difference method did not diverge for CFL 0.05, however the solution was very dispersed. The following contour can be seen for this.

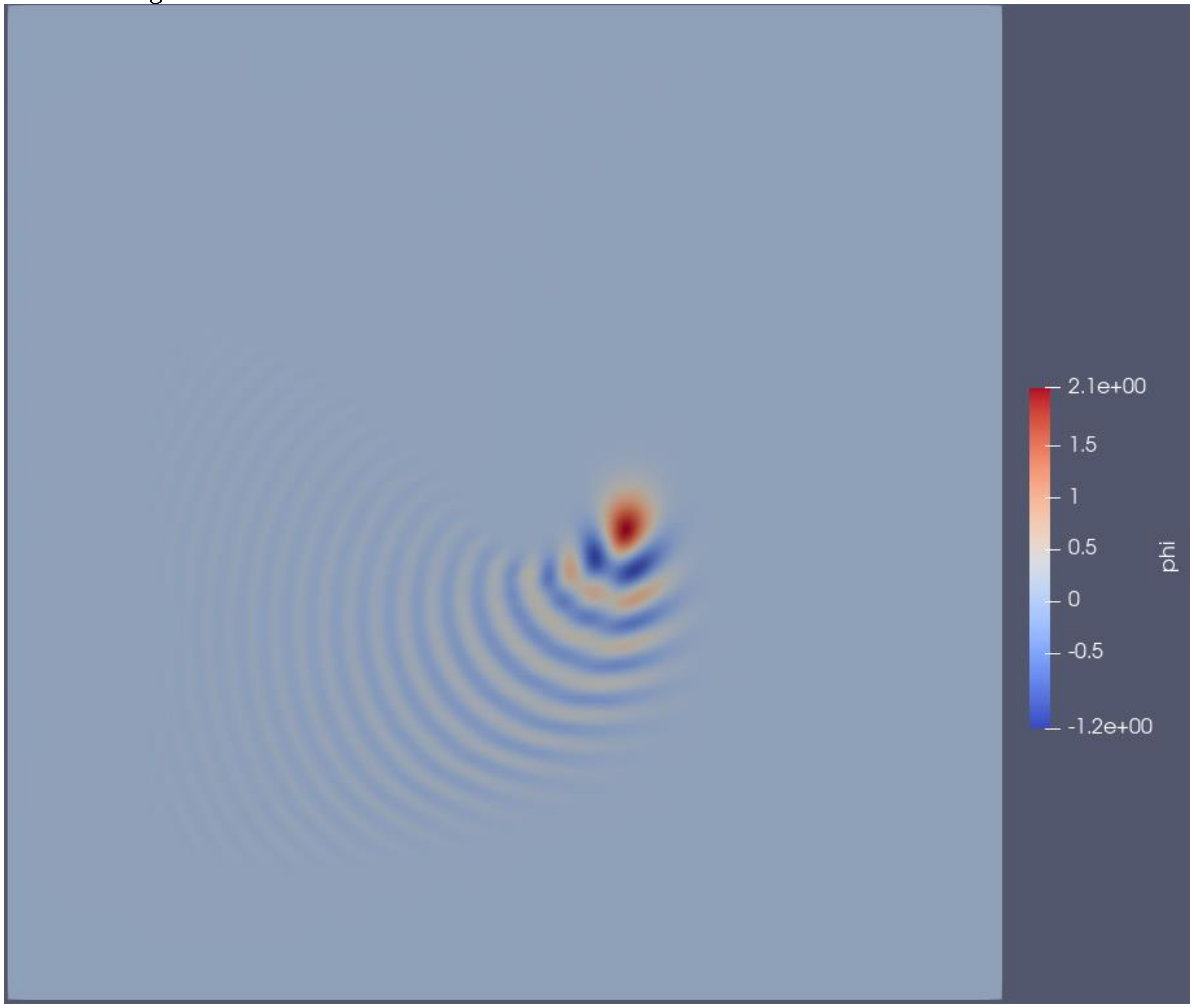


Figure 9: Central Difference with Forward Euler Contour

The solution is very smeared out, and dispersed, and overall, not accurate compared to what can be seen in Figure 8. This makes sense as the central difference scheme will take information from points that the signal has not reached yet. That is, assuming C>0, the information will travel from left to right, however the central difference scheme will take information from points that are to the right of the point of interest, where the information has not yet reached.

For Central difference with Adams-Bashforth method, the highest tested CFL number that convergence occurred was 0.2, which corresponds to a timestep of 0.002. The following contour was seen for this timestep.

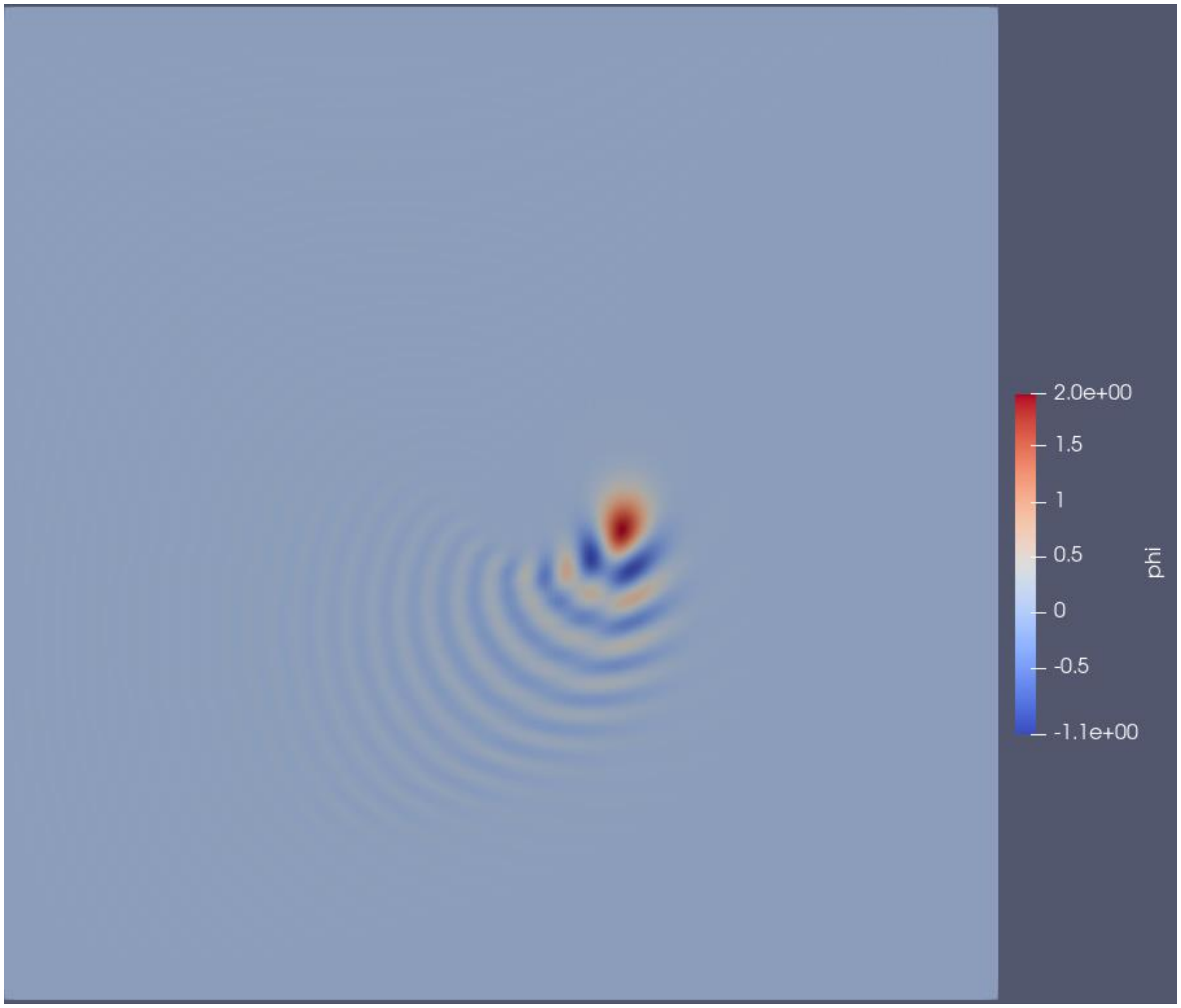


Figure 10: Central Difference with Adams Bashforth Contour

This contour shows that the solution is even more dissipated and smeared out than the forward Euler. This is for the same reasoning as the forward Euler. This can be explained by doing a Von Neumann stability analysis on the discretization. With the upwind scheme, a numerical viscosity term is added, while with the central difference scheme this is not added. Therefore very small timesteps need to be taken, in order for the information to not pass through an entire cell before the solution can be found for stability to be taken.

Part 2: Unsteady Advection-Diffusion Equation

1. Derive the error-residual form.

The error-residual derivation can be seen in the appendix handwritten derivations. As the convection equation is solved using an explicit method, the error residual for the entire equation basically becomes the equivalent of the error residual derived for the unsteady diffusion equation in part 1, combined with the solution found for the advection term using the Adams-Bashforth method. The residual code can be seen below:

```
R = id - 1.5 * fc_Curr + 0.5 * fc_Prev;  // Update residual to subtract
convective solution
```

Where id is the derived residual for the diffusion term, fc_curr is the current solution for the convective term, and fc_prev is the previous timestep solution for the convective term.

2. Determine the largest timestep for each grid size. Express your answer as CFL number.

A variety of simulations were run, in order to numerically determine the largest timestep at which convergence can be found. The following timesteps were found to be the largest at which convergence occurs.

Grid Size	17x17	65x65	257x257
Timestep	0.05625	0.058125	0.056441
CFL	0.9	3.72	14.4

These CFL numbers are over 1, which can be explained by 2 possibilities. The first possibility is that there is still a small bug in the code somewhere. The second possibility is that the discretization scheme used is a hybrid scheme. That is the, solution for the diffusion is found implicitly, while the solution for the advection term is found explicitly. Hybrid schemes can make it possible to overcome the CFL imposed stability limit, and allow for a convergent solution with a much higher timestep. [1] It is also important to note that the simulation was programmed to stop once the time has reached 2π. It is possible that the solution did not converge within a time of 2π, and if the simulation were left to run longer, than convergence can be achieved. If this was allowed, the CFL numbers may be closer for each grid size.

3. Integrate the equation to steady state, when $||\phi n+1 - \phi n||L2 < 1e - 8$.

a) Plot the contours of the steady state solution Φ. Compare the numerical solution with the exact solution along the lines x=0.5, y=0.5, and x=y. Can you claim the convergence of the solution.

The following contour was found for the exact solution of the equation.

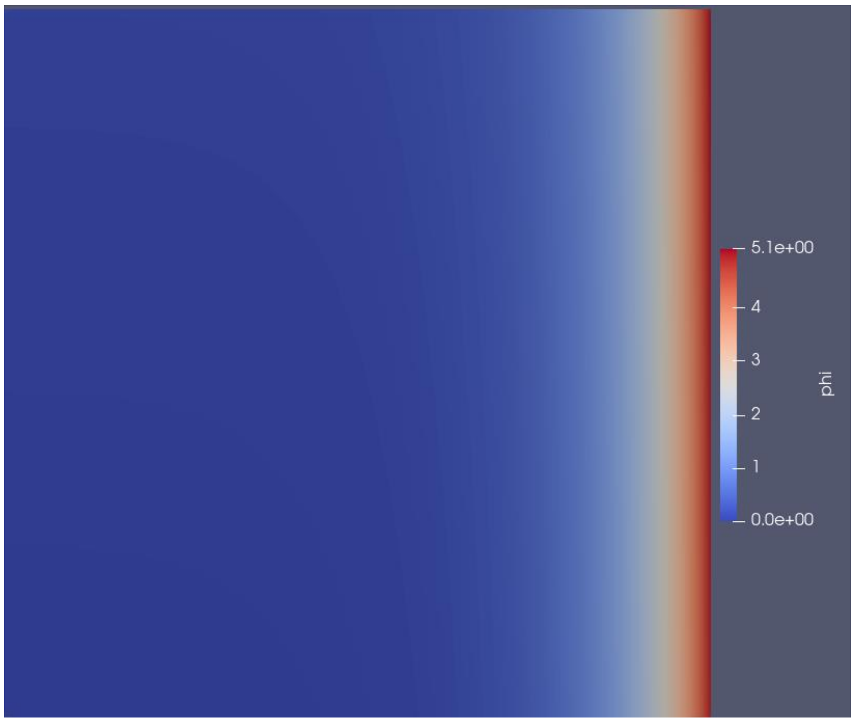


Figure 11: Exact Solution Contour

The following contours were found for each grid size, 17, 65, and 256, respectively.

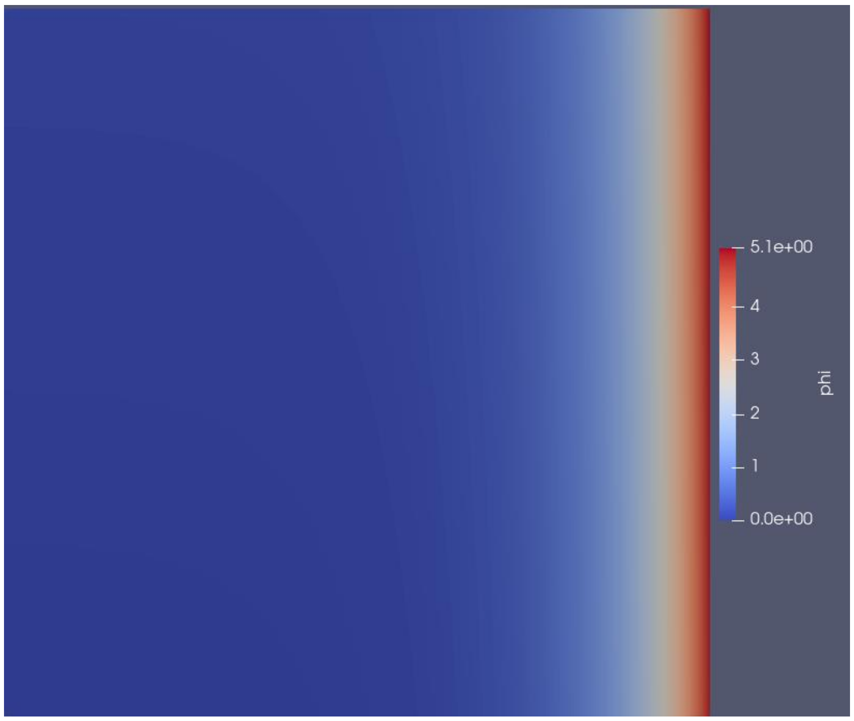


Figure 12: Converged Solution for 17 Grid Points

The following plots were generated which compare the exact solution to the numerical solution along the lines Y=X, Y=0.5, and X=0.5.

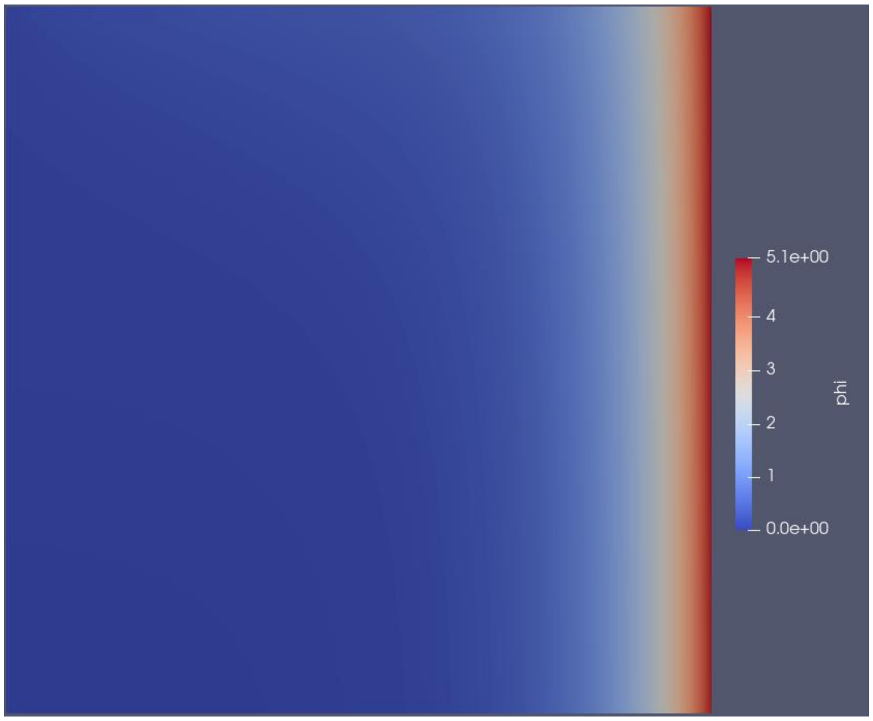


Figure 13: Converged Solution for 65 Grid Points

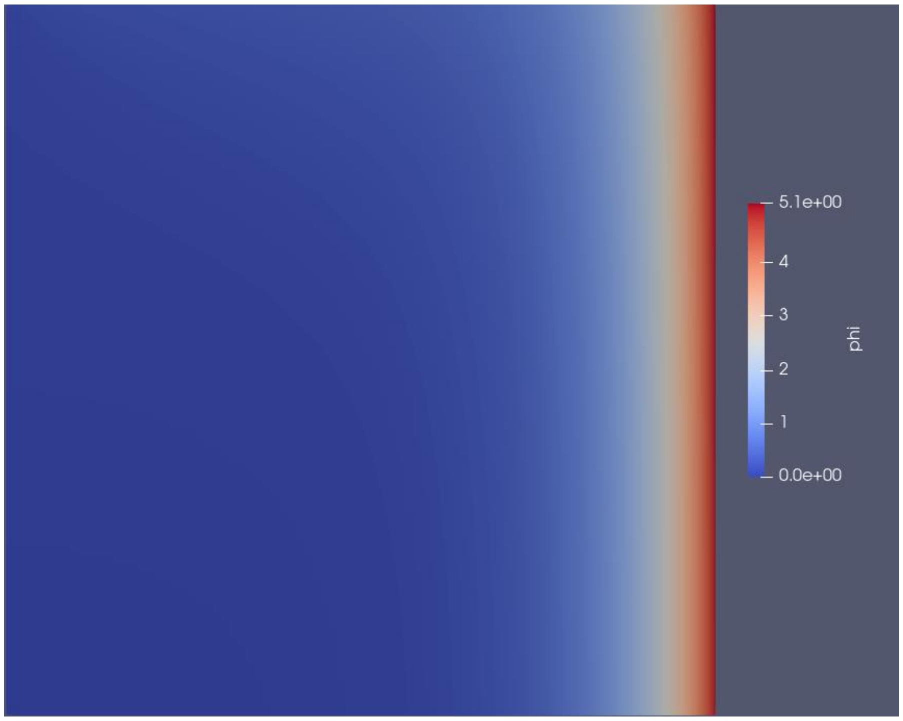


Figure 14: Converged Solution for 257 Grid Points

The following plots were generated which compare the exact solution to the numerical solution along the lines Y=X, Y=0.5, and X=0.5.

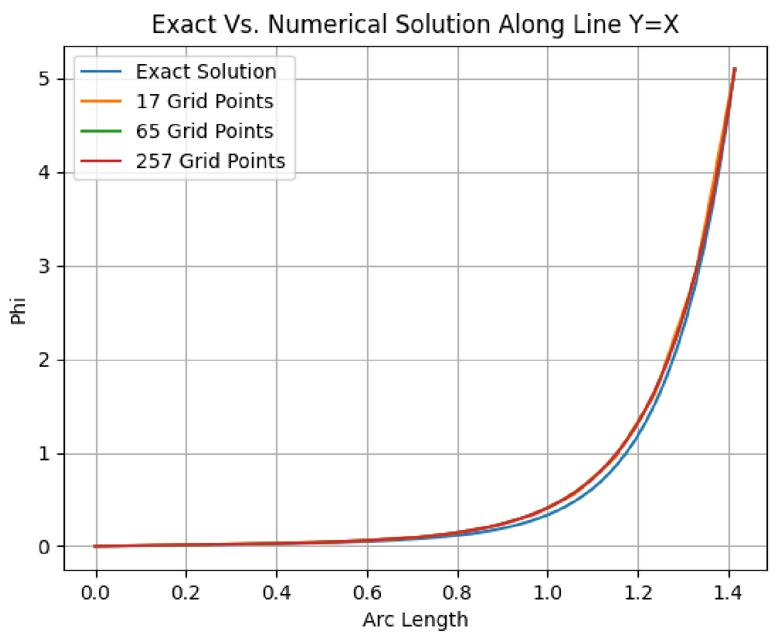


Figure 15: Exact Vs. Numerical Solution Along Y=X

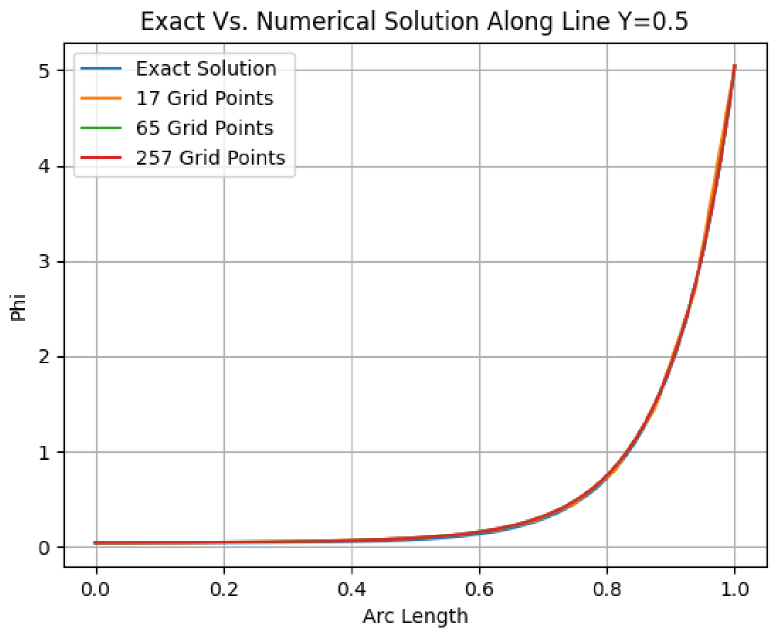


Figure 16: Exact Vs. Numerical Solution Along Y=0.5

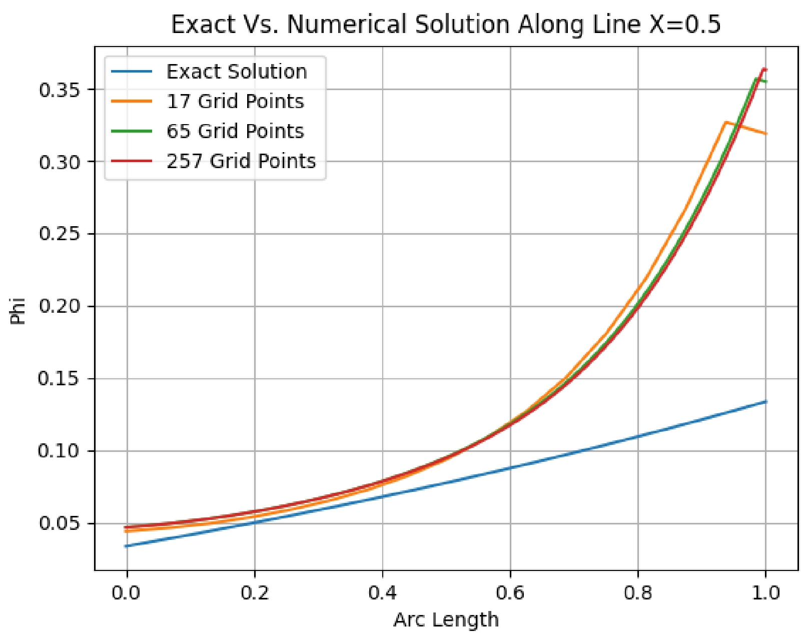


Figure 17: Exact Vs. Numerical Solution Along Y=0.5

The plots match very well along the lines Y=X, and Y=0.5. However, along the line X=0.5, the plot does not match well. As such, it is possible that although the code reached convergence, the solution is not well approximating the exact solution. This could be due to bugs in the code which were not yet found, or due to the inaccuracies that come with the numerical discretization of the differential equation. This solution was taken at the highest timestep that convergence was reached for each grid size. It is possible that at a lower timestep the numerical solution may better approximate the exact solution.

b) Determine $||\phi - \phi e||_{L2}$, and show the log-log plot of the L2 norm of error and grid-size. Calculate the slope of the line obtained.

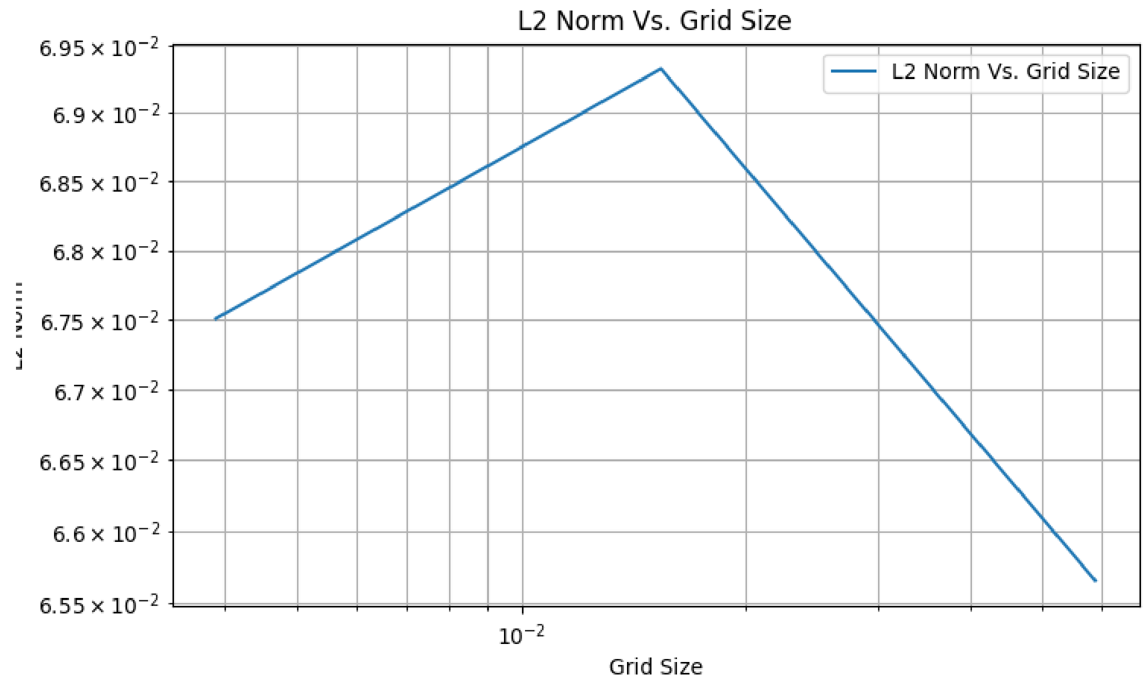


Figure 18: Error Norm Vs. Grid Size

Figure 18 shows $||\phi - \phi e||_{L2}$ compared to grid size. The same timestep of 0.05 was used to generate each of these points. This graph indicates that there is most likely a bug in the code, as the results are not expected. Instead, a straight line with a slope of 2 is expected. This is because the central difference method is accurate to an order of $O(\Delta x^2)$. The error residuals are all very close in this case.

4. Modify $\alpha = 0.0001$ and repeat steps 2. Do you have to modify Δt obtained in step 2?

When you change the diffusivity constant to 0.0001, the code blows up. This is due to the fact that in the initialization function for phi, there are terms that contain $e^{-v/\alpha}$. In this case, v is always 1. Therefore, when plugging this in, the term becomes $e^{-10\,000}$. The entries for the phi vector are contained with a double data type, which can contain values from 1.7e-308 to 1.7e308. However, e^{-10000} has a value of approximately 1.1e-4343 (according to WolframAlpha). Therefore, the value will be read as 0, causing the code to blow up to infinity due to a divide by 0 error.

1.2 a) $\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = 0$

$$\frac{\partial \phi}{\partial t} = \underbrace{-u\frac{\partial \phi}{\partial x} - v\frac{\partial \phi}{\partial y}}_{\text{convective term}}$$

BS - $\frac{u^n_{i,j} - u^n_{i-1,j}}{\Delta x} + \frac{u^n_{i,j} - u^n_{i,j-1}}{\Delta y}$

FS - $\frac{u^n_{i+1,j} - u^n_{i,j}}{\Delta x} + \frac{u^n_{i,j} - u^n_{i,j-1}}{\Delta y}$

Upwind

$-\frac{u\Delta t}{\Delta x}(u^n_{i,j} - u^n_{i-1,j}) - \frac{v\Delta t}{\Delta y}(u^n_{i,j} - u^n_{i,j-1}),\ u>0, v>0$ (1)

$-\frac{u\Delta t}{\Delta x}(u^n_{i+1,j} - u^n_{i,j}) - \frac{v\Delta t}{\Delta y}(u^n_{i,j+1} - u^n_{i,j}),\ u<0, v<0$ (2)

$-\frac{u\Delta t}{\Delta x}(u^n_{i,j} - u^n_{i-1,j}) - \frac{v\Delta t}{\Delta y}(u^n_{i,j+1} - u^n_{i,j}),\ u>0, v<0$ (3)

$-\frac{u\Delta t}{\Delta x}(u^n_{i+1,j} - u^n_{i,j}) - \frac{v\Delta t}{\Delta y}(u^n_{i,j} - u^n_{i,j-1}),\ u<0, v>0$ (4)

Define

$u^+ = \frac{1}{2}(u + |u|),\ u^+ = u$ if $u>0$, 0 if $u<0$ (5)

$u^- = \frac{1}{2}(u - |u|),\ u^- = u$ if $u<0$, 0 if $u>0$ (6)

$v^+ = \frac{1}{2}(v + |v|),\ v^+ = v$ if $v>0$, 0 if $v<0$ (7)

$v^- = \frac{1}{2}(v - |v|),\ v^- = v$ if $v<0$, 0 if $v>0$ (8)

Combine the previous to get a general 2D upwind discretization

$$= -\frac{\Delta t}{\Delta x}\left[u^+\left(u^n_{i,j} - u^n_{i-1,j}\right) + u^-\left(u^n_{i+1,j} - u^n_{i,j}\right)\right]$$
$$-\frac{\Delta t}{\Delta y}\left[v^+\left(u^n_{i,j} - u^n_{i,j-1}\right) + v^-\left(u^n_{i,j+1} - u^n_{i,j}\right)\right]$$

$$= -\frac{\Delta t}{\Delta x}\left[\frac{1}{2}(u+|u|)\left(u^n_{i,j} - u^n_{i-1,j}\right) + \frac{1}{2}(u-|u|)\left(u^n_{i+1,j} - u^n_{i,j}\right)\right]$$
$$-\frac{\Delta t}{\Delta y}\left[\frac{1}{2}(v+|v|)\left(u^n_{i,j} - u^n_{i,j-1}\right) + \frac{1}{2}(v-|v|)\left(u^n_{i,j+1} - u^n_{i,j}\right)\right]$$

$$= -\frac{\Delta t}{\Delta x}\Big[\frac{1}{2}\left(u\,u^n_{i,j} - u\,u^n_{i-1,j} + |u|\,u^n_{i,j} - |u|\,u^n_{i-1,j}\right)$$
$$+ \frac{1}{2}\left(u\,u^n_{i+1,j} - u\,u^n_{i,j} - |u|\,u^n_{i+1,j} + |u|\,u^n_{i,j}\right)\Big]$$
$$-\frac{\Delta t}{\Delta y}\Big[\frac{1}{2}\left(v\,u^n_{i,j} - v\,u^n_{i,j-1} + |v|\,u^n_{i,j} - |v|\,u^n_{i,j-1}\right) +$$
$$\frac{1}{2}\left(v\,u^n_{i,j+1} - v\,u^n_{i,j} - |v|\,u^n_{i,j+1} + |v|\,u^n_{i,j}\right)\Big]$$

$$= -\frac{\Delta t}{2\Delta x}\Big[\cancel{u\,u^n_{i,j}} - u\,u^n_{i-1,j} + \overbrace{|u|\,u^n_{i,j}}^{2} - |u|\,u^n_{i-1,j} +$$
$$u\,u_{i+1,j} - \cancel{u\,u^n_{i,j}} - |u|\,u^n_{i+1,j} + \cancel{|u|\,u^n_{i,j}}\Big]$$
$$-\frac{\Delta t}{2\Delta y}\Big[\cancel{v\,u^n_{i,j}} - v\,u^n_{i,j-1} + \overbrace{|v|\,u^n_{i,j}}^{3} - |v|\,u^n_{i,j-1} +$$
$$v\,u^n_{i,j+1} - \cancel{v\,u^n_{i,j}} - |v|\,u^n_{i,j+1} + \cancel{|v|\,u^n_{i,j}}\Big]$$

$$= -\frac{\Delta t}{2\Delta x}\left[u\left(u^n_{i+1,j} - u^n_{i-1,j}\right) + |u|\left(2u^n_{i,j} - u^n_{i-1,j} - u^n_{i+1,j}\right)\right]$$

$$-\frac{\Delta t}{2\Delta y}\left[v\left(u^n_{i,j+1} - u^n_{i,j-1}\right) + |v|\left(2u^n_{i,j} - u^n_{i,j-1} - u^n_{i,j+1}\right)\right]$$

$$\approx -\frac{u\Delta t}{2\Delta x}\left(u^n_{i+1,j} - u^n_{i-1,j}\right) + \frac{|u|\Delta t}{2\Delta x}\left(u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}\right)$$

$$-\frac{v\Delta t}{2\Delta y}\left(u^n_{i,j+1} - u^n_{i,j-1}\right) + \frac{|v|\Delta t}{2\Delta y}\left(u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}\right)$$

1.2b) $f\,T \; - \; \frac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t}$

Complete upwind

$$u^{n+1}_{i,j} = u^n_{i,j} - \frac{u\Delta t}{2\Delta x}\left(u^n_{i+1,j} - u^n_{i-1,j}\right) + \frac{|u|\Delta t}{2\Delta \alpha}\left(u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}\right)$$

$$-\frac{v\Delta t}{2\Delta y}\left(u^n_{i,j+1} - u^n_{i,j-1}\right) + \frac{|v|\Delta t}{2\Delta y}\left(u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}\right)$$

1.3a)

2nd Order Backward $\quad$ 2nd Order forward

$$\frac{3u_i^n - 4u_{i-1}^n + u_{i-2}^n}{2\Delta x} \qquad \frac{-u_{i+2}^n + 4u_{i+1}^n - 3u_{i}^n}{2\Delta x}$$

Define

$u^+ = \frac{1}{2}(u+|u|)$, $u^+ = u$ if $u>0$, 0 if $u<0$

$u^- = \frac{1}{2}(u-|u|)$, $u^- = u$ if $u<0$, 0 if $u>0$

$v^+ = \frac{1}{2}(v+|v|)$, $v^+ = v$ if $v>0$, 0 if $v<0$

$v^- = \frac{1}{2}(v-|v|)$, $v^- = v$ if $v<0$, 0 if $v>0$

2nd order Upwind

$$\approx -\frac{\Delta t}{2\Delta x}\left[u^+(3u_{i,j}^n - 4u_{i-1,j}^n + u_{i-2,j}^n) + u^-(-3u_{i,j}^n + 4u_{i+1,j}^n - u_{i+2,j}^n)\right]$$
$$-\frac{\Delta t}{2\Delta y}\left[v^+(3u_{i,j}^n - 4u_{i,j-1}^n + u_{i,j-2}^n) + v^-(-3u_{i,j}^n + 4u_{i,j+1}^n - u_{i,j+2}^n)\right]$$

$$= -\frac{\Delta t}{2\Delta x}\left[\frac{1}{2}(u+|u|)(3u_{i,j}^n - 4u_{i-1,j}^n + u_{i-2,j}^n) + \frac{1}{2}(u-|u|)(-3u_{i,j}^n + 4u_{i+1,j}^n - u_{i+2,j}^n)\right]$$
$$-\frac{\Delta t}{2\Delta y}\left[\frac{1}{2}(v+|v|)(3u_{i,j}^n - 4u_{i,j-1}^n + u_{i,j-2}^n) + \frac{1}{2}(v-|v|)(-3u_{i,j}^n + 4u_{i,j+1}^n - u_{i,j+2}^n)\right]$$

$$= -\frac{\Delta t}{4\Delta x}\Big[u(\cancel{3u_{i,j}^n} - 4u_{i-1,j}^n + u_{i-2,j}^n - \cancel{3u_{i,j}^n} + 4u_{i+1,j}^n - u_{i+2,j}^n)$$
$$+ |u|(\overset{6}{\cancel{3}}u_{i,j}^n - 4u_{i-1,j}^n + u_{i-2,j}^n + \cancel{3u_{i,j}^n} - 4u_{i+1,j}^n + u_{i+2,j}^n)\Big]$$
$$-\frac{\Delta t}{4\Delta y}\Big[v(\cancel{3u_{i,j}^n} - 4u_{i,j-1}^n + u_{i,j-2}^n - \cancel{3u_{i,j}^n} + 4u_{i,j+1}^n - 4u_{i,j+2}^n)$$
$$+ |v|(\overset{6}{\cancel{3}}u_{i,j}^n - 4u_{i,j-1}^n + u_{i,j-2}^n + \cancel{3u_{i,j}^n} - 4u_{i,j+1}^n + u_{i,j+2}^n)\Big]$$

$$= -\frac{u\Delta t}{4\Delta x}\left[u^n_{i-2,j} - 4u^n_{i-1,j} + 4u^n_{i+1,j} - u^n_{i+2,j}\right]$$

$$- \frac{|u|\Delta t}{4\Delta x}\left[u^n_{i-2,j} - 4u^n_{i-1,j} + 6u^n_{i,j} - 4u^n_{i+1,j} + u^n_{i+2,j}\right]$$

$$- \frac{v\Delta t}{4\Delta y}\left[u^n_{i,j-2} - 4u^n_{i,j-1} + 4u^n_{i,j+1} - u^n_{i,j+2}\right]$$

$$- \frac{|v|\Delta t}{4\Delta y}\left[u^n_{i,j-2} - 4u^n_{i,j-1} + 6u^n_{i,j} - 4u^n_{i,j+1} + u^n_{i,j+2}\right]$$

1.3b) 2nd Order Adams Bashforth method

$$u^{n+1} = u^n + \Delta t\left(\frac{3}{2}g(t^n, u^n) - \frac{1}{2}g(t^{n-1}, u^{n-1})\right)$$

Taking

$$- \frac{u\Delta t}{4\Delta x}\left[u^n_{i-2,j} - 4u^n_{i-1,j} + 4u^n_{i+1,j} - u^n_{i+2,j}\right]$$

$$- \frac{|u|\Delta t}{4\Delta x}\left[u^n_{i-2,j} - 4u^n_{i-1,j} + 6u^n_{i,j} - 4u^n_{i+1,j} + u^n_{i+2,j}\right]$$

$$- \frac{v\Delta t}{4\Delta y}\left[u^n_{i,j-2} - 4u^n_{i,j-1} + 4u^n_{i,j+1} - u^n_{i,j+2}\right]$$

$$- \frac{|v|\Delta t}{4\Delta y}\left[u^n_{i,j-2} - 4u^n_{i,j-1} + 6u^n_{i,j} - 4u^n_{i+1,j} + u^n_{i+2,j}\right]$$

as $g(t^n, u^n)$ (spatial derivatives) we obtain the following

$$
\begin{aligned}
u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t \Bigg[\frac{3}{2} \Bigg(& -\frac{u\Delta t}{4\Delta x}\left[u_{i-2,j}^{n} - 4u_{i-1,j}^{n} + 4u_{i+1,j}^{n} - u_{i+2,j}^{n}\right] \\
& - \frac{|u|\Delta t}{4\Delta x}\left[u_{i-2,j}^{n} - 4u_{i-1,j}^{n} + 6u_{i,j}^{n} - 4u_{i+1,j}^{n} + u_{i+2,j}^{n}\right] \\
& - \frac{v\Delta t}{4\Delta y}\left[u_{i,j-2}^{n} - 4u_{i,j-1}^{n} + 4u_{i+1,j}^{n} - u_{i+2,j}^{n}\right] \\
& - \frac{|v|\Delta t}{4\Delta y}\left[u_{i,j-2}^{n} - 4u_{i,j-1}^{n} + 6u_{i,j}^{n} - 4u_{i+1,j}^{n} + 4_{i+2,j}^{n}\right] \Bigg) \\
- \frac{1}{2} \Bigg(& -\frac{u\Delta t}{4\Delta x}\left[u_{i-2,j}^{n-1} - 4u_{i-1,j}^{n-1} + 4u_{i+1,j}^{n-1} - u_{i+2,j}^{n-1}\right] \\
& - \frac{|u|\Delta t}{4\Delta x}\left[u_{i-2,j}^{n-1} - 4u_{i-1,j}^{n-1} + 6u_{i,j}^{n-1} - 4u_{i+1,j}^{n-1} + u_{i+2,j}^{n-1}\right] \\
& - \frac{v\Delta t}{4\Delta y}\left[u_{i,j-2}^{n-1} - 4u_{i,j-1}^{n-1} + 4u_{i+1,j}^{n-1} - u_{i+2,j}^{n-1}\right] \\
& - \frac{|v|\Delta t}{4\Delta y}\left[u_{i,j-2}^{n-1} - 4u_{i,j-1}^{n-1} + 6u_{i,j}^{n-1} - 4u_{i+1,j}^{n-1} + 4_{i+2,j}^{n-1}\right] \Bigg)
\end{aligned}
$$

Central difference - Advection

$$\left(u\frac{\partial\phi}{\partial x}+v\frac{\partial\phi}{\partial y}\right)\approx u\left(\frac{\phi^n_{i+1,j}-\phi^n_{i-1,j}}{2\Delta x}\right)+v\left(\frac{\phi^n_{i,j+1}-\phi^n_{i,j-1}}{2\Delta y}\right)$$

Central difference - Diffusion

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\approx\frac{\phi^n_{i+1,j}-2\phi^n_{i,j}+\phi^n_{i-1,j}}{\Delta x^2}+\frac{\phi^n_{i,j+1}-2\phi^n_{i,j}+\phi^n_{i,j-1}}{\Delta y^2}$$

2.1) $\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = \alpha\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right)$

We can let $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \nabla^2 \phi$

Newton Raphson

- Start with guess ϕ_0
- update $\phi_{k+1} = \phi_k + \delta\phi_k$
- Hope $\nabla^2 \phi_{k+1} = 0$

If $\nabla^2 \phi_{k+1} = 0 \rightarrow \nabla^2(\phi_k + \delta\phi_k) = 0$
Only doing for diffusion $\rightarrow$ Linear
$\nabla^2(\phi_k + \delta\phi_k) = \nabla^2\phi_k + \nabla^2(\delta\phi_k) \rightarrow \nabla^2(\delta\phi_k) = -\nabla^2\phi_k$
Update the solution $\phi_{k+1} = \phi_k + \delta\phi_k$ until $\|\delta\phi_k\|_2 < tol$

We have $\nabla^2(\delta\phi_k) = -\nabla^2\phi_k$

True solution is $\nabla^2\phi = 0$ so we have $-\nabla\phi_k$ as our residual

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\delta\phi_k = -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\phi_k$$

We discretize ϕ using Central difference,

$$A\delta\phi_k = -R$$

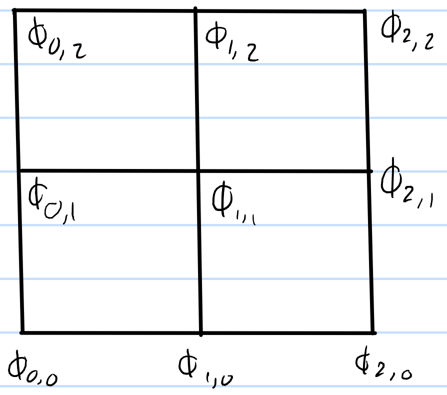
Discretizing

$$\left(\frac{\partial^2}{\partial x^2}\delta\phi\right)_{i,j} = \frac{\delta\phi_{i-1,j} - 2\delta\phi_{i,j} + \delta\phi_{i+1,j}}{\Delta x^2}$$

$$\left(\frac{\partial^2}{\partial y^2}\delta\phi\right)_{i,j} = \frac{\delta\phi_{i,j-1} - 2\delta\phi_{i,j} + \delta\phi_{i,j+1}}{\Delta y^2}$$

so $\left(\frac{\partial^2 \delta\phi}{\partial x^2} + \frac{\partial^2 \delta\phi}{\partial y^2}\right)_{i,j} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2}$

Take 3x3 matrix



We can discretize $\phi_{1,1}$ to be

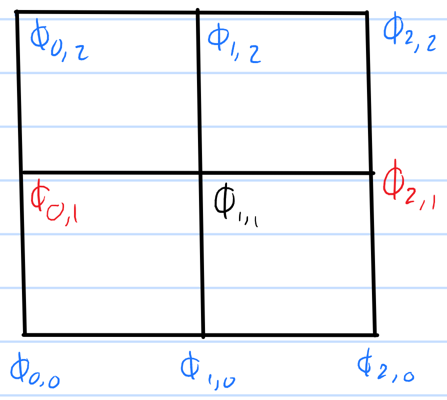
$$\frac{\phi_{0,1}-2\phi_{1,1}+\phi_{2,1}}{\Delta x^2}+\frac{\phi_{1,0}-2\phi_{1,1}+\phi_{1,2}}{\Delta y^2}$$

Isolate all ϕ

$$\delta\phi_{1,1}=\frac{1}{\Delta x^2}\delta\phi_{0,1}+\frac{1}{\Delta y^2}\delta\phi_{1,0}+\left(-\frac{2}{\Delta x^2}-\frac{2}{\Delta y^2}\right)\delta\phi_{1,1}+\frac{1}{\Delta y^2}\delta\phi_{1,2}+\frac{1}{\Delta x^2}\delta\phi_{2,1}$$

At boundary nodes - If Neumann we need to update $\delta\phi$

- If Dirichlet, value is known so $\delta\phi=0$



- In this case we have Neumann boundary on top & bottom

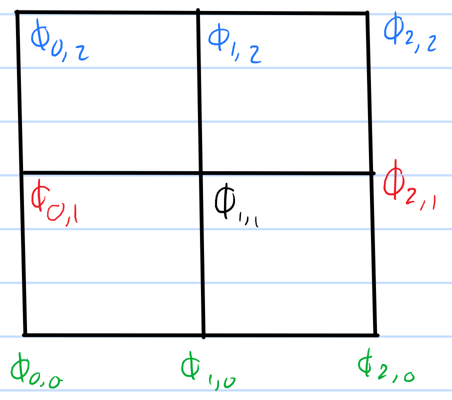
- Dirichlet Boundary
- Neumann Boundary

At Neumann boundaries we obtain $\frac{\partial\phi}{\partial n}=h$, where h is the Prescribed Neumann value

We will apply forward difference in y on bottom boundary & top boundary on top. We get the following

$$\frac{\delta\phi_{0,1}^{k+1}-\delta\phi_{0,0}^{k+1}}{\Delta y}=h\rightarrow\frac{(\phi_{1,1}^{k}+\delta\phi_{1,1}^{k})-(\phi_{1,0}^{k}+\delta\phi_{1,0}^{k})}{\Delta y}=h$$

$$\frac{1}{\Delta y}\delta\phi_{0,1}-\frac{1}{\Delta y}\delta\phi_{0,0}=h-\frac{\phi_{0,1}-\phi_{0,0}}{\Delta y}\longleftarrow \text{Point } 0,0$$

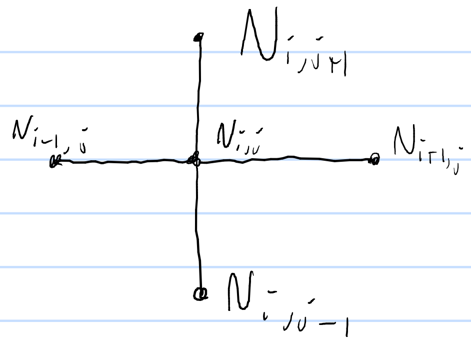


$\delta\phi_{0,1} = 0$ $\delta\phi_{0,0} =$

$\delta\phi_{2,1} = 0$

r, dPhi, phi, V, G,

α



for bottom Boundary

$$\underbrace{\frac{1}{\Delta y}\delta\phi^{k}_{i,1} - \frac{1}{\Delta y}\delta\phi^{k}_{i,0}} = \underbrace{h - \frac{\phi^{k}_{i,1} - \phi^{k}_{i,0}}{\Delta y}}$$

for Top boundary

$$\frac{1}{\Delta y}\delta\phi_{i,N_y-1} - \frac{1}{\Delta y}\delta\phi_{i,N_y-2} = h - \frac{\phi_{i,N_y-2} - \phi_{i,N_y-1}}{\Delta y}$$

Matrix form

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta x^2} & 0 & \frac{1}{\Delta y^2} & \left(-\frac{2}{\Delta x^2}-\frac{2}{\Delta y^2}\right) & \frac{1}{\Delta y^2} & 0 & \frac{1}{\Delta x^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta\phi_{0,0} \\ \delta\phi_{0,1} \\ \delta\phi_{0,2} \\ \delta\phi_{1,0} \\ \delta\phi_{1,1} \\ \delta\phi_{1,2} \\ \delta\phi_{2,0} \\ \delta\phi_{2,1} \\ \delta\phi_{2,2} \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

$$-\frac{1}{\Delta y}\delta\phi_{i,0} = h - \frac{\phi_{i,1}-\phi_{i,0}}{\Delta y} - \frac{1}{\Delta y}\delta\phi_{i,1}$$

$$\delta\phi_{i,0} = (\phi_{i,1}-\phi_{i,0}) + \delta\phi_{i,1} - h\Delta y$$

for Top boundary

$$\frac{1}{\Delta y}\delta\phi_{i,N_y-2} - \frac{1}{\Delta y}\delta\phi_{i,N_y-1} = h - \frac{\phi_{i,N_y-2}-\phi_{i,N_y-1}}{\Delta y}$$