MATH-6840
Vignesh Ramakrishnan
RIN: 662028006
Due: Monday March 21, 2022

Problem Set 6

1. (25 pts.) Consider the advection-diffusion equation

$$\begin{aligned} u_t + au_x - \nu u_{xx} &= f(x,t), \qquad x \in (0,1), \qquad 0 < t \le T_f \\ u(x,0) &= u_0(x), \qquad x \in (0,1) \\ u(0,t) &= \alpha(t), \\ u_x(1,t) &= \beta(t), \qquad t \ge 0 \qquad t \ge 0. \end{aligned}$$

(a) Determine $f(x,t)$, $u_0(x)$, $\alpha(t)$, and $\beta(t)$ so that the exact solution to the problem is $u(x,t) = 2\cos(3x)\cos(t)$.

$$\begin{aligned} u_{ex}(x,t) &= 2\cos(3x)\cos(t) \\ u_t(x,t) &= -2\cos(3x)\sin(t) \\ u_x(x,t) &= -6\sin(3x)\cos(t) \\ u_{xx}(x,t) &= -18\cos(3x)\cos(t) \\ u_t + au_x - \nu u_{xx} &= \cos(3x)\left[18\nu\cos(t) - 2\sin(t)\right] - 6a\sin(3x)\cos(t) = f(x,t) \\ u(x,t=0) &= 2\cos(3x) = u_0(x,t=0) \\ u(x=0,t) &= 2\cos(t) = \alpha(t) \\ u_x(x=1,t) &= -6\sin(3)\cos(t) = \beta(t) \end{aligned}$$

(b) Now using the computational grid defined by $x_j = j\Delta x$, $-1, 0, \ldots, N+1$, with $\Delta x = 1/N$ (note there is a ghost cell at left and right), define a discrete treatment of the boundary conditions that is at least second-order accurate.

$$x_j = j\Delta x, \; j = -1, 0, 1, 2, \ldots\ldots, N_x, N_x + 1$$

$$v_0^{n+1} = \frac{v_{-1}^{n+1} + v_1^{n+1}}{2} = 2\cos(t^{n+1})$$

$$-6\sin(3)\cos(t^{n+1}) = D_{0x} v_{N_x}^{n+1} = \frac{v_{N_x+1}^{n+1} - v_{N_x-1}^{n+1}}{2\Delta x}$$

(c) Write a code to solve this problem using the Crank-Nicolson scheme

$$D_{+t} v_j^n = \left(-aD_{0x} + \nu D_{+x} D_{-x}\right) \frac{v_j^{n+1} + v_j^n}{2} + \frac{f_j^{n+1} + f_j^n}{2}.$$

for all interior j (exact values may depend on your discrete BCs), along with the BCs you defined in part (b) above.

When the scheme is worked out, the final discretization has the form:

$$\begin{aligned} -\left(\frac{r}{2} + \frac{\sigma}{4}\right) v_{j-1}^{n+1} + (1+r)\, v_j^{n+1} - \left(\frac{r}{2} - \frac{\sigma}{4}\right) v_{j+1}^{n+1} = \\ \left(\frac{r}{2} + \frac{\sigma}{4}\right) v_{j+1}^{n} + (1-r)\, v_j^{n} + \left(\frac{r}{2} - \frac{\sigma}{4}\right) v_{j+1}^{n} + \frac{f_j^{n+1} + f_j^n}{2} \\ , \; j = 0, 1, 2, 3, \ldots\ldots N_x \end{aligned}$$

where, $r = \frac{\nu\Delta t}{\Delta x^2}$ and $\sigma = \frac{a\Delta t}{\Delta x}$. The code is attached below in Listing 1.

```
function [e,max_err,xd] = ADCrankNicholson(tlim2,Nx,nStep)
a  = 1;
nu = 1;
xlim1 = 0;
xlim2 = 1;
tlim1 = 0;

dx = (xlim2-xlim1)/Nx;
dt = (tlim2-tlim1)/nStep;

r  = nu*dt/dx^2;
s  = a*dt/dx;

ng = 1;
NP = Nx+1+2*ng;
ja = ng+1;
jb = NP-ng;

A  = zeros(NP);
b  = zeros(NP,1);
u  = zeros(NP,1);


x = (xlim1:dx:xlim2);
x = [xlim1-dx x xlim2+dx];
t = (tlim1:dt:tlim2);

% set IC
for j=1:length(x)
    u(j) = getEX(x(j),t(1));
end
% u(ng) = -u(ja+1) + 2*getEX(xlim1,tlim1);
% u(NP) = u(jb-1) + 2*dx*getUx(tlim1);
%plot(x,u)

for j=ng:NP
    if j==ng
        A(j,j)     = 1;
        A(j,ja+1)  = 1;
    elseif j==NP
        A(j,jb-1)  = -1;
        A(j,NP)    = 1;
    else
        A(j,j-1)   = -(r/2 + s/4);
        A(j,j)     = (1+r);
        A(j,j+1)   = -(r/2 - s/4);
    end
end

for i=2:length(t)
    uold = u;
    for j=ng:NP
        if j==ng
            b(j) = 2*getEX(xlim1,t(i));
        elseif j==NP
            b(j) = 2*dx*getUx(t(i));
        else
            f_avg  = 0.5*(getF(x(j),t(i),nu,a) + getF(x(j),t(i-1),nu,a));
            b(j)   = (r/2 + s/4)*uold(j-1) + (1-r)*uold(j) + (r/2-s/4)*
```

```
    uold(j+1) + dt*f_avg;
            end
        end
        u = A\b;
end

uex = zeros(NP,1);
for j=1:NP
    uex(j,1) = getEX(x(j),tlim2);
end

max_err = max(abs(uex(ja:jb)-u(ja:jb)));
e       = (uex(ja:jb)-u(ja:jb));
xd      = x(ja:jb);

% figure
% plot(x,u)

end

%%
function uex = getEX(x,t)
uex = 2*cos(3*x)*cos(t);
end

function uxBC = getUx(t)
uxBC = -6*sin(3)*cos(t);
end

function f = getF(x,t,nu,a)
f = cos(3*x)*(18*nu*cos(t)-2*sin(t))-6*a*sin(3*x)*cos(t);
end
```

Listing 1: Crank-Nicholson scheme - AD Equation

(d) Perform a grid refinement study with $a = 1$, $\nu = 1$, and $\Delta t = \Delta x$. Present results for the maximum error in the approximation at $t = 1$. Discuss the observed order-of-accuracy.
The scheme developed initially converges at an order $\approx \mathcal{O}(\Delta x^2)$.

(e) In your computations, you should have observed stability for $\Delta t = \Delta x$. Perform a stability analysis for the Cauchy problem (i.e. the infinite domain problem with no BCs) to partially explain this.
Taking a DFT on the Cauchy problem gives,

$$\frac{r}{2}\left(e^{i\xi} - 2 + e^{-i\xi}\right)\hat{V}^{n+1} + \frac{\sigma}{4}\left(e^{i\xi} - e^{-i\xi}\right)\hat{V}^{n+1} + \hat{V}^{n+1} = \hat{V}^{n} - \frac{\sigma}{4}\left(e^{i\xi} - e^{-i\xi}\right)\hat{V}^{n} + \frac{r}{2}\left(e^{i\xi} - 2 + e^{-i\xi}\right)\hat{V}^{n}$$

This simplifies to,

$$\left((1 + r(1 - \cos\xi)) + i\frac{\sigma}{2}\sin\xi\right)\hat{V}^{n+1} = \left((1 - r(1 - \cos\xi)) - i\frac{\sigma}{2}\sin\xi\right)\hat{V}^{n}$$
$$\hat{V}^{n+1} = a(\xi)v_j^n$$

where $|a(\xi)| \leq 1$ or in other words, $|Nr|^2 \leq |Dr|^2$ for this $a(\xi)$

$$(1 - r(1 - \cos\xi))^2 + \left(\frac{\sigma}{2}\sin\xi\right)^2 \leq (1 + r(1 - \cos\xi))^2 + \left(\frac{\sigma}{2}\sin\xi\right)^2$$

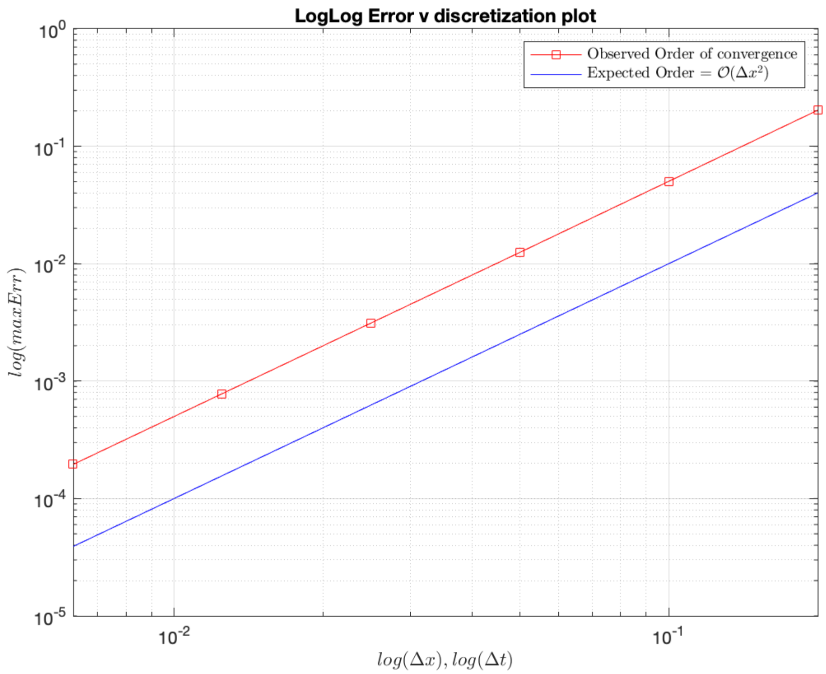

Figure 1: Grid Refinement - LogLog plot

This simplifies to

$$4r\,(1 - \cos\xi) \geq 0$$

This is always true since $r \geq 0$ and $(1 - \cos\xi) \geq 0$. Hence this is unconditionally stable.

2. (25 pts.) Consider the initial-boundary value problem

$$u_t = \nu(u_{xx} + u_{yy}), \quad 0 < x < \pi, \quad 0 < y < \pi, \quad t > 0$$

with initial condition $u(x, y, t = 0) = u_0(x, y)$, and boundary conditions

$$u(0, y, t) = u(\pi, y, t) = 0$$
$$u_y(x, 0, t) = u_y(x, \pi, t) = 0.$$

(a) Define a computational grid and second-order accurate discrete BCs. You can use ghost cells or not, as you see fit.
My preferred choice for this problem was to have one ghost point on either side of the x direction and have 1 ghost point on either side of the y direction.

$$x_j = j\Delta x,\ j = -1, 0, 1, 2, 3, \ldots\ldots N_x, N_x + 1$$
$$y_k = k\Delta y,\ k = -1, 0, 1, 2, \ldots\ldots N_y, N_y + 1$$

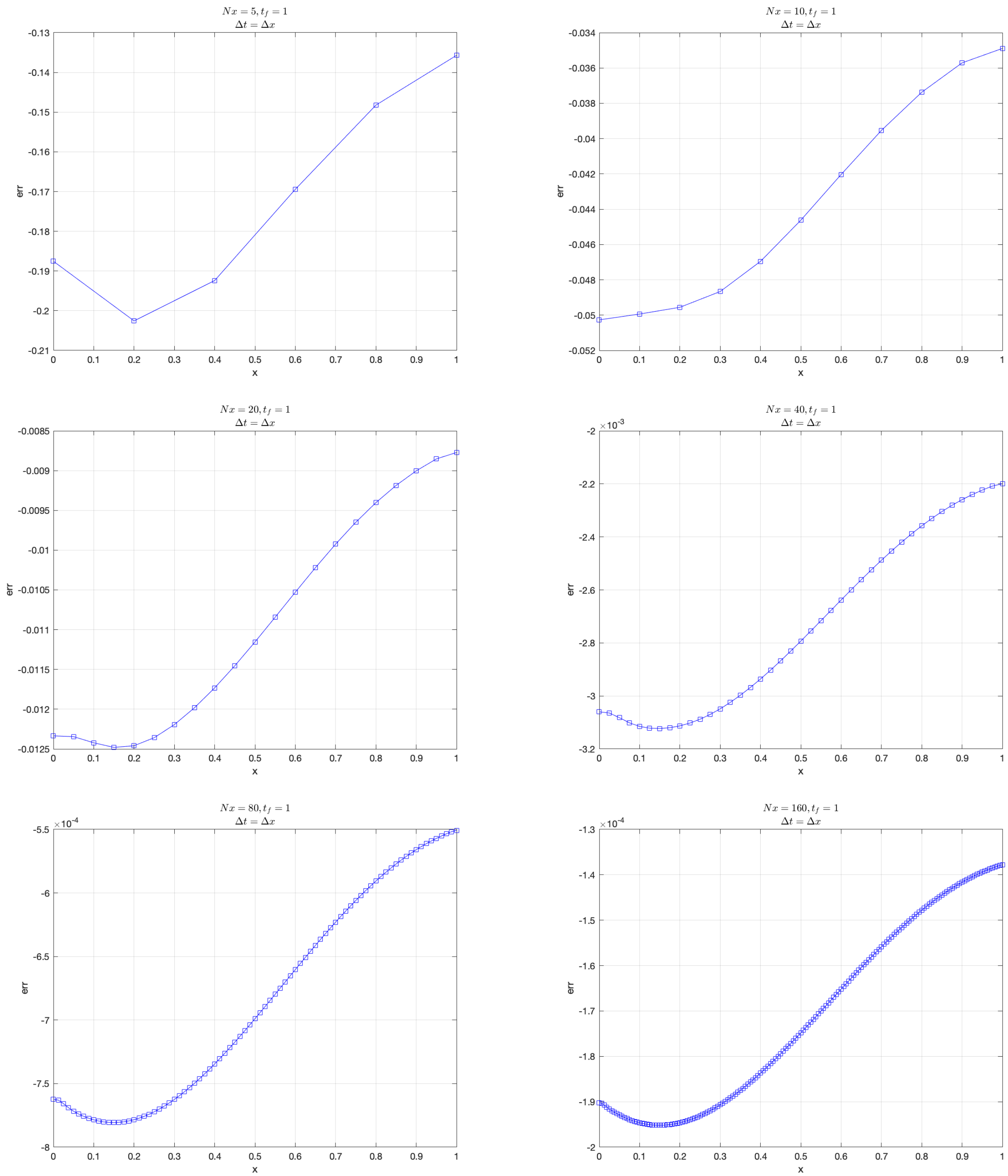

Figure 2: Error Plot v x

(b) Write a code to solve this problem using the ADI scheme of Peaceman and Rachford. The discretization for this scheme is presented as

$$-\frac{r_x}{2}v_{j-1,k}^{n+\frac{1}{2}}+(1+r_x)\,v_{j,k}^{n+\frac{1}{2}}-\frac{r_x}{2}v_{j+1,k}^{n+\frac{1}{2}}=$$
$$\frac{r_y}{2}v_{j,k-1}^{n}+(1-r_y)\,v_{j,k}^{n}+\frac{r_y}{2}v_{j,k+1}^{n},$$
$$(k=0,1,2,3,\ldots\ldots\ldots N_y)$$

$$-\frac{r_y}{2}v_{j,k-1}^{n+1}+(1+r_y)\,v_{j,k}^{n+1}-\frac{r_y}{2}v_{j,k+1}^{n+1}=$$
$$\frac{r_x}{2}v_{j-1,k}^{n+\frac{1}{2}}+(1-r_x)\,v_{j,k}^{n+\frac{1}{2}}+\frac{r_x}{2}v_{j+1,k}^{n+\frac{1}{2}}$$
$$j=0,1,2,3,\ldots\ldots\ldots N_x$$

The Boundary conditions used for this problem are:

$$\frac{v_{-1,k}^{n+1}+v_{1,k}^{n+1}}{2}=0 \text{ (Left Boundary condition)}$$
$$\frac{v_{N_x+1,k}^{n+1}+v_{N_x-1,k}^{n+1}}{2}=0 \text{ (Right Boundary condition)}$$
$$\frac{v_{j,1}^{n+1}-v_{j,-1}^{n+1}}{2\Delta y}=0 \text{ (Bottom Boundary condition)}$$
$$\frac{v_{j,N_y+1}^{n+1}-v_{j,N_y-1}^{n+1}}{2\Delta y}=0 \text{ (Top Boundary condition)}$$

The ADI scheme of Peaceman and Rachford is attached as a listing below in Listing 2

```
function [x,y,e,u,uex,max_err] = HeatEqnADI2(tlim2,Nx,Ny,nStep,iOption)

xlim1 = 0;
xlim2 = pi;
ylim1 = 0;
ylim2 = pi;
tlim1 = 0;

nu = 1;

% Define dx,dy,dt
dx = (xlim2-xlim1)/Nx;
dy = (ylim2-ylim1)/Ny;
dt = (tlim2-tlim1)/nStep;

% Define Data Structures
ng_x   = 1;
ng_y   = 1;
NxTot  = Nx + 1 + 2*ng_x;
NyTot  = Ny + 1 + 2*ng_y;
jax    = ng_x + 1;        % Index of X-Boundary (x=xlim1)
jbx    = NxTot- ng_x;     % Index of X-Boundary (x=xlim2)
jay    = ng_y + 1;        % Index of Y-Boundary (y=ylim1)
jby    = NyTot- ng_y;     % Index of Y-Boundary (y=ylim2)
```

```
% Define Domain
x = (xlim1:dx:xlim2);
x = [xlim1-dx x xlim2+dx];

y = (ylim1:dy:ylim2);
y = [ylim1-dy y ylim2+dy];


rx = nu*dt/(dx^2);
ry = nu*dt/(dy^2);

% Solution variable
u  = zeros(NxTot,NyTot);

A1halfStep = zeros(NxTot);
q1halfStep = zeros(NxTot,1);
A2fullStep = zeros(NyTot);
q2fullStep = zeros(NyTot,1);

% Define Intial conditions and nu at all locations
for k=1:NyTot
    for j=1:NxTot
        u(j,k)  = getIC(x(j),y(k),iOption);
    end
end

% Time marching to find 'u' at tf
for i=1:nStep

    % 1st half step
    uold = u;
    for k=jay:jby  % looping through y axis
        for j=ng_x:NxTot
            if j== ng_x

                A1halfStep(j,j)     = 1;
                A1halfStep(j,jax+1) = 1;
                q1halfStep(j,1)     = 2*0; % BC1

            elseif j==NxTot

                A1halfStep(j,j)     = 1;
                A1halfStep(j,jbx-1) = 1;
                q1halfStep(j,1)     = 2*0; %BC2

            else

                A1halfStep(j,j-1)   = -rx/2;
                A1halfStep(j,j)     = 1+rx;
                A1halfStep(j,j+1)   = -rx/2;
                q1halfStep(j,1)     = 0.5*ry*uold(j,k-1) + (1-ry)*uold(j,
   k)...
                                      + 0.5*ry*uold(j,k+1);
            end
        end
        u(:,k) = A1halfStep\q1halfStep;
    end

```

```
    % 2nd Half Step implicit
    uold = u;
    for j=jax:jbx       % looping through x-axis
        for k=ng_y:NyTot
            if k==ng_y

                A2fullStep(k,k)      = -1;
                A2fullStep(k,jay+1)  = 1;
                q2fullStep(k,1)      = 0;

            elseif k==NyTot

                A2fullStep(k,k)      = 1;
                A2fullStep(k,jby-1)  = -1;
                q2fullStep(k,1)      = 0;

            else

                A2fullStep(k,k-1)    = -ry/2;
                A2fullStep(k,k)      = 1+ry;
                A2fullStep(k,k+1)    = -ry/2;
                q2fullStep(k,1)      = 0.5*rx*uold(j-1,k)+(1-rx)*uold(j,k)
    +...
                                       0.5*rx*uold(j+1,k);
            end
        end
        u(j,:) = A2fullStep\q2fullStep;
    end
end

uex = zeros(NxTot,NyTot);

for k=1:NyTot
    for j=1:NxTot
        uex(j,k)  = getEX(x(j),y(k),tlim2,nu,iOption);
    end
end
e       = u-uex;
max_err = max(max(abs(e(jax:jbx,jay:jby))));
end

%%
function uex = getEX(x,y,t,nu,iOption)
if(iOption==1)
    uex = sin(x)*cos(y)*exp(-2*nu*t) - 3*sin(x)*cos(2*y)*exp(-5*nu*t);
else
    uex = 0;
end
end
```

Listing 2: ADI scheme - 2D Heat Equation

(c) Setting $u_0(x,y) = \sin(x)(\cos(y) - 3\cos(2y))$, compare the numerical and exact solutions at $t = 1$.

Set $u = \hat{u}e^{ik_1x}e^{ik_2y}e^{-i\omega t}$ and performing dispersion analysis,

$$\begin{aligned} u_t =& -i\omega u \\ u_{xx} =& -k_1^2 u \\ u_{yy} =& -k_2^2 u \\ \omega =& -i\nu(k_1^2 + k_2^2) \\ u =& \hat{u}e^{ik_1x}e^{ik_2y}e^{-\nu\left(k_1^2+k_2^2\right)t} \end{aligned}$$

When $t = 0$, the initial condition is given by $u_0(x, y) = \sin(x)\left(\cos(y) - 3\cos(2y)\right)$. Hence the solution is written as the linear combination of these two terms present

$$u(x, y, t = 0) = \sin(x)\cos(y)e^{-i\omega_1(0)} - 3\sin(x)\cos(2y)e^{-i\omega_2(0)}$$

where $\omega_1 = -i\nu(1^2 + 1^2)$ and $\omega_2 = -i\nu(1^2 + 2^2)$. Therefore the solution is written as,

$$u(x, y, t) = \sin(x)\cos(y)e^{-2\nu t} - 3\sin(x)\cos(2y)e^{-5\nu t}$$

The comparison between the numerical and the exact solution is shown in Fig 3.

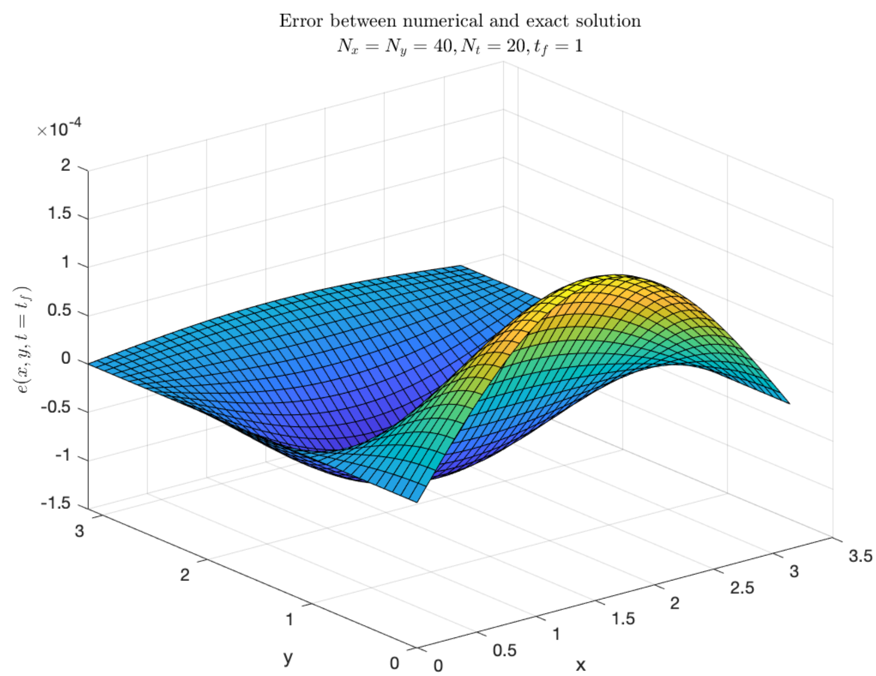

Figure 3: Comparison between numerical and exact solution - ADI Scheme

(d) Perform a grid refinement study to verify second-order convergence in both space and time.
A grid refinement study was performed and the spatial and temporal order of convergence plots are attached in Fig 4.

(e) Find a numerical solution using 40 grid lines in both physical dimensions for the case when

$$u_0(x, y) = \begin{cases} 1 & \text{if} \quad (x - \frac{\pi}{2})^2 + (y - \frac{\pi}{2})^2 < \frac{1}{2} \\ 0 & \text{else.} \end{cases}$$

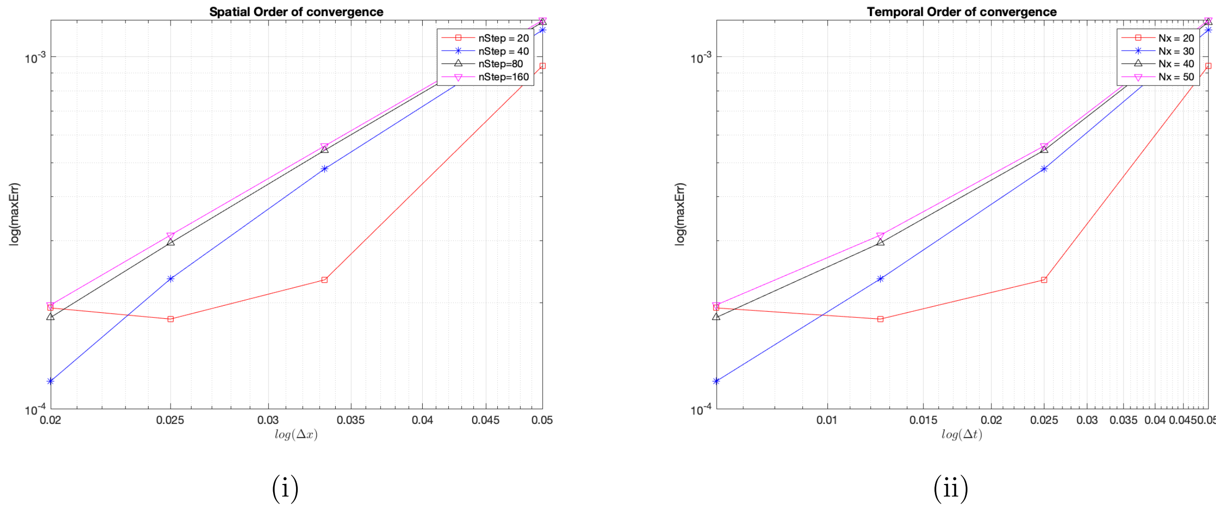

Figure 4: Order of Convergence - ADI scheme

Plot your results at $t = 0$, $t = .1$, and $t = .5$
For this initial function, the results are presented in Fig 5

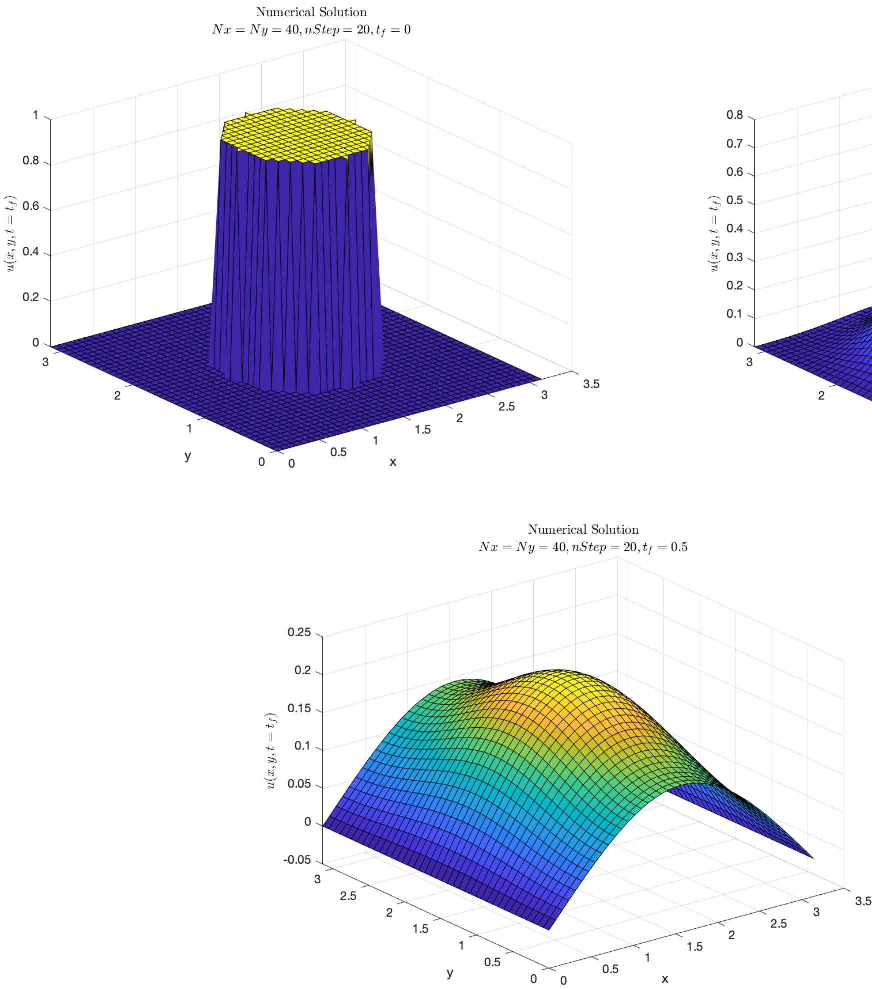

Figure 5: Numerical Solution at different end times