MATH-6840
Vignesh Ramakrishnan
Due: Monday March 28, 2022

Problem Set 7

1. (25 pts.) Consider the variable coefficient diffusion equation

$$u_t = (Du_x)_x, \qquad 0 < x < 1, \qquad t > 0$$

where $D > 0$ is a function of x, with initial conditions $u(x,0) = u_0(x)$ and subject to boundary conditions $u_x(0,t) = 0$, $u(1,t) = 0$.

(a) Show that this problem is well-posed by proving the energy estimate

$$\frac{d}{dt}||u||^2 \leq 0,$$

where $||u||^2 = (u,u) = \int_0^1 u^2\, dx$.

$$\begin{aligned}
u_t &= (Du_x)_x \\
u\, u_t &= u\, (Du_x)_x \\
\int_0^1 u\, u_t\, dx &= \int_0^1 u\, (Du_x)_x\, dx \\
\int_0^1 \frac{1}{2}\frac{d||u||^2}{dt}\, dx &= \cancel{u(1)}\,(Du_x(1)) - u(0)\left(D\cancel{u_x(0)}\right) - \int_0^1 u_x D u_x\, dx \\
\int_0^1 \frac{1}{2}\frac{d||u||^2}{dt}\, dx &= -\int_0^1 D\,(u_x)^2\, dx
\end{aligned}$$

Here, $D \geq 0$ and $(u_x)^2 \geq 0$ always and hence,

$$\frac{1}{2}\frac{d||u||^2}{dt} \leq 0$$

Thus, this PDE is well posed.

(b) Propose a computational grid, a semi-discretization of the PDE in space, and a treatment of the boundary conditions for which you can prove a discrete energy estimate of the form $\frac{d}{dt}||u||_h^2 \leq 0$ for an appropriately defined discrete norm $||\cdot||_h$. Prove your discrete energy estimate.

$$\begin{aligned}
\Delta x &= 1/N \\
x_j &= (j - \frac{1}{2})\Delta x,\ j = 0, 1, 2, \ldots\ldots, N
\end{aligned}$$

and here, $D_{-x}v_j = \frac{v_{j+\frac{1}{2}} - v_{j-\frac{1}{2}}}{\Delta x}$

$$\begin{aligned}
\frac{dv_j}{dt} &= D_{+x}\left(\nu_{j-\frac{1}{2}} D_{-x} v_j\right) \\
v_j \frac{dv_j}{dt} &= v_j D_{+x}\left(\nu_{j-\frac{1}{2}} D_{-x} v_j\right) \\
\Delta x \sum_{j=0}^{N-1} \frac{1}{2}\frac{d||v_j||_h^2}{dt} &= \Delta x \sum_{j=0}^{N-1} v_j D_{+x}\left(\nu_{j-\frac{1}{2}} D_{-x} v_j\right) \\
\Delta x \sum_{j=0}^{N-1} \frac{1}{2}\frac{d||v_j||_h^2}{dt} &= \frac{\Delta x}{\Delta x^2} \sum_{j=0}^{N-1} v_j \Delta_{+x}\left(\nu_{j-\frac{1}{2}} \Delta_{-x} v_j\right)
\end{aligned}$$

Using summation by parts, we can simplify and get to the form

$$\Delta x \sum_{j=0}^{N-1} \frac{1}{2}\frac{d||v_j||_h^2}{dt} = \frac{\Delta x}{\Delta x^2}\left(\cancel{v_N}\left(\nu_{N-\frac{1}{2}}\Delta_{-x}v_N\right) - v_0\nu_{-\frac{1}{2}}\Delta_{-x}v_0 - \sum_{j=0}^{N-1}\left(\nu_{j+\frac{1}{2}}\Delta_{-x}v_{j+1}\right)\Delta_{+x}v_j\right)$$

$$\sum_{j=0}^{N-1} \frac{1}{2}\frac{d||v_j||_h^2}{dt} = \left(-\frac{1}{\Delta x}v_0\nu_{-\frac{1}{2}}\cancel{D_{-x}v_0} - \frac{1}{\Delta x^2}\sum_{j=0}^{N-1}\left(\nu_{j+\frac{1}{2}}\left(\Delta_{+x}v_j\right)^2\right)\right)$$

Since $\nu_{j+\frac{1}{2}} \geq 0$ and $(\Delta_{+x}v_j)^2 \geq 0$,

$$\frac{d||v_j||_h^2}{dt} \leq 0$$

(c) Prove a fully discrete energy estimate for Crank-Nicolson integration of the scheme from part (b) above.

$$\frac{v_j^{n+1} - v_j^n}{\Delta t} = D_{+x}\left(\nu_{j-\frac{1}{2}}D_{-x}\right)\frac{v_j^{n+1}+v_j^n}{2}$$

$$\left(v_j^{n+1}\right)^2 - \left(v_j^n\right)^2 = \Delta t D_{+x}\left(\nu_{j-\frac{1}{2}}D_{-x}\right)\frac{v_j^{n+1}+v_j^n}{2}\left(v_j^{n+1}+v_j^n\right)$$

$$\left(v_j^{n+1}\right)^2 - \left(v_j^n\right)^2 = \frac{\Delta t}{4}D_{+x}\left(\nu_{j-\frac{1}{2}}D_{-x}\right)v_j^{n+\frac{1}{2}}v_j^{n+\frac{1}{2}}$$

$$\Delta x\sum_{j=0}^{N-1}\left(v_j^{n+1}\right)^2 - \left(v_j^n\right)^2 = -\Delta x\sum_{j=0}^{N-1}\left(D_{+x}v_j^{n+\frac{1}{2}}, D_{+x}\left(\nu_{j-\frac{1}{2}}v_j^{n+\frac{1}{2}}\right)\right) + + \left[v_j^{n+\frac{1}{2}}D_{+x}v_j^{n+\frac{1}{2}}\right]_{j=0}^{j=N}$$

(d) Write a code to implement the scheme in part (d). Demonstrate second-order convergence using a manufactured solution.

Method of manufactured solutions is used to verify second-order of convergence.

$$u_{ex} = \cos\left(\frac{\pi x}{2}\right)\sin t,\ u_{ex}(x=1,t)=0$$

$$u_t = \cos\left(\frac{\pi x}{2}\right)\cos t$$

$$u_x = -\frac{\pi}{2}\sin\left(\frac{\pi x}{2}\right)\sin t,\ u_x(x=0,t)=0$$

$$u_{xx} = -\frac{\pi^2}{4}\cos\left(\frac{\pi x}{2}\right)\sin t$$

$$u_t - (\nu u_x)_x = u_t - (\nu_x u_x + \nu u_{xx}) = f(x,t)$$

This solution satisfies the boundary conditions specified in the problem. The discrete set of equations for the Crank-Nicholson scheme is,

$$D_{+t}v_j^n = D_{+x}\left(\nu_{j-\frac{1}{2}}D_{-x}\right)\left(\frac{v_j^{n+1}+v_j^n}{2}\right) + \left(\frac{f_j^{n+1}+f_j^n}{2}\right)$$

$$D_{+t}v_j^n = \frac{1}{2}D_{+x}\left(\nu_{j-\frac{1}{2}}v_j^{n+1}\right) + \frac{1}{2}D_{+x}\left(\nu_{j-\frac{1}{2}}v_j^n\right) + \left(\frac{f_j^{n+1}+f_j^n}{2}\right)$$

Setting $\hat{F} = \left(\frac{f_j^{n+1}+f_j^n}{2}\right)$ and $\Gamma = \frac{\Delta t}{\Delta x^2}$, this simplifies to,

$$
\begin{aligned}
-\frac{\Gamma}{2}\nu_{j-\frac{1}{2}}v_{j-1}^{n+1} + \left(1+\frac{\Gamma}{2}\left(\nu_{j+\frac{1}{2}}+\nu_{j-\frac{1}{2}}\right)\right)v_j^{n+1} - \frac{\Gamma}{2}\nu_{j+\frac{1}{2}}v_{j+1}^{n+1} = \\
\frac{\Gamma}{2}\nu_{j-\frac{1}{2}}v_{j-1}^{n} + \left(1-\frac{\Gamma}{2}\left(\nu_{j+\frac{1}{2}}+\nu_{j-\frac{1}{2}}\right)\right)v_j^{n} + \frac{\Gamma}{2}\nu_{j+\frac{1}{2}}v_{j+1}^{n} + \Delta t\hat{F} \\
j = 1, 2, \ldots\ldots, N-1
\end{aligned}
$$

Boundary conditions are given by,

$$
\frac{v_1^{n+1}-v_0^{n+1}}{\Delta x} = 0, \qquad \frac{v_N^{n+1}+v_{N-1}^{n+1}}{2} = 0
$$

The code is shown in Listing 1. The order of convergence is shown in Fig 1.

```
function [err,norm_err,xd] = HeatEqnEnergyCN2(N,nStep,tf,fOption,cOption)

xlim1 = 0;
xlim2 = 1;
tlim1 = 0;
tlim2 = tf;

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


x = (xlim1+dx/2:dx:xlim2-dx/2);
x = [xlim1-dx/2 x xlim2+dx/2];

NTot = length(x);

assert(length(x)==NTot,'Length mismatch');

t = (tlim1:dt:tlim2);

u  = zeros(NTot,1);
nu = zeros(NTot,1);
for j=1:NTot
    u(j)  = getEx(x(j),tlim1,fOption);
    nu(j) = getNu(x(j),cOption);
end

A = zeros(NTot);

Gamma = dt/dx^2;

for j=1:NTot
    if j==1
        A(j,j)   = -1;
        A(j,j+1) = 1;
    elseif j==NTot
        A(j,j)   = 1;
        A(j,j-1) = 1;
    else
        nmh = 0.5*(nu(j-1)+nu(j));
        nph = 0.5*(nu(j)+nu(j+1));
        A(j,j-1) = -(Gamma/2)*nmh;
        A(j,j)   = 1 + (Gamma/2)*(nph+nmh);
```

```
        A(j,j+1) = -(Gamma/2)*nph;
    end
end

b = zeros(NTot,1);
for i=2:length(t)
    uold = u;
    for j=1:NTot
        if j==1
            b(j) = dx*getUx(xlim1,t(i),fOption);
        elseif j==NTot
            b(j) = 2*getEx(xlim2,t(i),fOption);
        else
            nmh  = 0.5*(nu(j-1)+nu(j));
            nph  = 0.5*(nu(j)+nu(j+1));

            fnp  = getF(x(j),t(i),fOption,cOption);
            fn   = getF(x(j),t(i-1),fOption,cOption);
            fhat = 0.5*(fnp+fn);

            b(j) = (Gamma/2)*nmh*uold(j-1)+...
                   (1-Gamma*0.5*(nph+nmh))*uold(j)+...
                   (Gamma/2)*nph*uold(j+1) + dt*fhat;
        end
    end
    u = A\b;
end

uex = zeros(NTot,1);
for j=1:NTot
    uex(j) = getEx(x(j),tlim2,fOption);
end

err = u(2:NTot-1)-uex(2:NTot-1);
norm_err = max(abs(err));


xd = x(2:NTot-1);
end

%% nu functions

function nu = getNu(x,cOption)
if cOption==1
    m   = 1;
    nu0 = 0.5;
    nu  = m*x + nu0;
elseif cOption ==2
    nu = (x+0.5)^2;
else
    nu = 1;
end
end


function nux = getNux(x,cOption)
if cOption==1
    m = 1;
    nux = m*(x^0);
elseif cOption ==2
    nux = 2*(x+0.5);
```

```
else
    nux = 0;
end
end

%% u functions

function uex = getEx(x,t,fOption)
if fOption==1
    uex = cos(pi*x/2)*sin(t);
else
    uex = 0;
end
end

function ut = getUt(x,t,fOption)
if fOption==1
    ut = cos(pi*x/2)*cos(t);
else
    ut = 0;
end
end

function ux = getUx(x,t,fOption)
if fOption==1
    ux = -(pi/2)*sin(pi*x/2)*sin(t);
else
    ux = 0;
end
end

function uxx = getUxx(x,t,fOption)
if fOption==1
    uxx = -(pi^2/4)*cos(pi*x/2)*sin(t);
else
    uxx = 0;
end
end

function f = getF(x,t,fOption,cOption)
if fOption==1
    ut  = getUt(x,t,fOption);
    ux  = getUx(x,t,fOption);
    uxx = getUxx(x,t,fOption);

    nu  = getNu(x,cOption);
    nux = getNux(x,cOption);

    f   = ut - (nux*ux + nu*uxx);
else
end
end
```

Listing 1: Crank-Nicholson scheme - variable co-efficient Heat Equation

2. (25 pts.) Consider a heat conduction problem in an annular section

$$u_t = \nu\left[\frac{1}{r}(ru_r)_r + \frac{1}{r^2}u_{\theta\theta}\right], \quad 1 < r < 2, \quad 0 < \theta < \frac{\pi}{2}, \quad t > 0$$

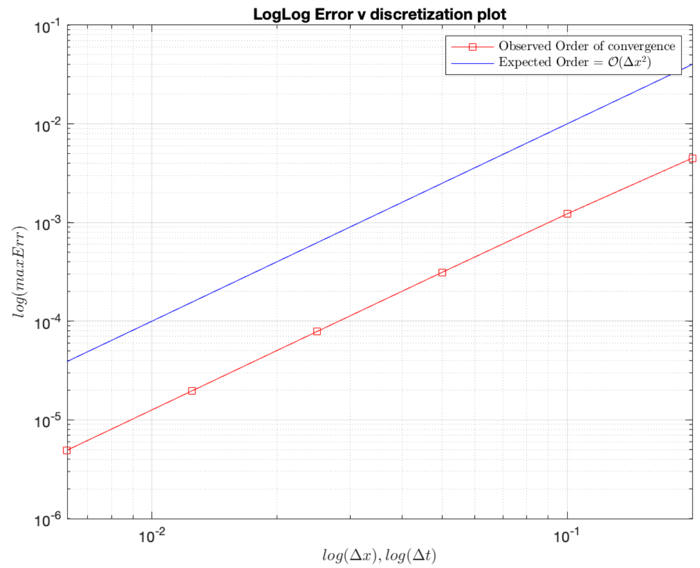


Figure 1: Order of Convergence plot

with initial condition $u(r,\theta,0) = u_0(r,\theta)$, and boundary conditions

$$u_r(1,\theta,t) = \alpha_1(\theta,t), \qquad u_r(2,\theta,t) = \alpha_2(\theta,t)$$
$$u(r,0,t) = \beta_1(r,t) \qquad u(r,\frac{\pi}{2},t) = \beta_2(r,t).$$

(a) Write a second-order accurate code to solve this problem using centered differencing and Crank-Nicolson time integration.

$$\begin{aligned}
-\frac{r_1}{2r_{j,k}} r_{j-\frac{1}{2},k} v_{j-1,k}^{n+1} + \left(1 + \frac{r_1}{2r_{j,k}}\left(r_{j+\frac{1}{2},k} + r_{j-\frac{1}{2},k}\right) + \frac{r_2}{r_{j,k}^2}\right) v_{j,k}^{n+1} - \frac{r_1}{2r_{j,k}} r_{j+\frac{1}{2},k} v_{j+1,k}^{n+1} \\
- \frac{r_2}{2r_{j,k}^2} v_{j,k-1}^{n+1} - \frac{r_2}{2r_{j,k}^2} v_{j,k+1}^{n+1} = \frac{r_1}{2r_{j,k}} r_{j-\frac{1}{2},k} v_{j-1,k}^{n} + \\
\left(1 - \frac{r_1}{2r_{j,k}}\left(r_{j+\frac{1}{2},k} + r_{j-\frac{1}{2},k}\right) - \frac{r_2}{r_{j,k}^2}\right) v_{j,k}^{n} + \\
\frac{r_1}{2r_{j,k}} r_{j+\frac{1}{2},k} v_{j+1,k}^{n} + \frac{r_2}{2r_{j,k}^2} v_{j,k-1}^{n} + \frac{r_2}{2r_{j,k}^2} v_{j,k+1}^{n} + \Delta t \hat{F} \\
j = 0,1,2,\ldots\ldots, N_r - 1, N_r \\
k = 0,1,2,\ldots\ldots, N_\theta - 1, N_\theta
\end{aligned}$$

Boundary conditions are given by,

$$\frac{v_{1,k}^{n+1} - v_{-1,k}^{n+1}}{2\Delta r} = \alpha_1\left(r = 1, \theta, (n+1)\Delta t\right) \qquad \frac{v_{j,-1}^{n+1} + v_{j,1}^{n+1}}{2} = \beta_1\left(r, \theta = 0, (n+1)\Delta t\right)$$

$$\frac{v_{N_r+1,k}^{n+1} - v_{N_r-1,k}^{n+1}}{2\Delta r} = \alpha_2\left(r = 2, \theta, (n+1)\Delta t\right) \quad \frac{v_{j,N_\theta-1}^{n+1} + v_{j,N_\theta+1}^{n+1}}{2} = \beta_2\left(r, \theta = \frac{\pi}{2}, (n+1)\Delta t\right)$$

The code written for this is shown in Listing 2, Listing 3, and Listing 4.

```
function mesh = genMesh(rlim1,rlim2,slim1,slim2,Nr,Ns)

dr = (rlim2-rlim1)/Nr;
ds = (slim2-slim1)/Ns;

ng    = 1;
NrTot = Nr+1+2*ng;
NsTot = Ns+1+2*ng;
jar   = ng+1;
jbr   = NrTot-ng;
jas   = ng+1;
jbs   = NsTot-ng;

r = (rlim1:dr:rlim2);
r = [rlim1-dr r rlim2+dr];
s = (slim1:ds:slim2);
s = [slim1-ds s slim2+ds];

GridFn = cell(NsTot,NrTot);
DOF    = zeros(NsTot,NrTot);
IDX    = cell(NsTot*NrTot,1);
dof = 1;
for k=1:NsTot
    for j=1:NrTot
        GridFn{k,j} = [r(j) s(k)];
        DOF(k,j)    = dof;
        IDX{dof}    = [k,j];
        dof = dof + 1;
    end
end

mesh.grid  = GridFn;
mesh.DOF   = DOF;
mesh.IDX   = IDX;
mesh.NrTot = NrTot;
mesh.NsTot = NsTot;
mesh.ng    = ng;
mesh.jar   = jar;
mesh.jbr   = jbr;
mesh.jas   = jas;
mesh.jbs   = jbs;
mesh.dr    = dr;
mesh.ds    = ds;
end
```

Listing 2: Mesh Generation code

```
function [max_err,ud,uex,err] = HeatEqn2DMapping(Nr,Ns,nStep,tf,iOption)

nu = 1;

rlim1 = 1;
rlim2 = 2;
slim1 = 0;
slim2 = pi/2;
tlim1 = 0;

mesh = genMesh(rlim1,rlim2,slim1,slim2,Nr,Ns);

NrTot = mesh.NrTot;
```

```
NsTot = mesh.NsTot;
ng    = mesh.ng;
jar   = mesh.jar;
jbr   = mesh.jbr;
jas   = mesh.jas;
jbs   = mesh.jbs;
dr    = mesh.dr;
ds    = mesh.ds;
dt    = (tf-tlim1)/nStep;

t  = (tlim1:dt:tf);

r1 = nu*dt/dr^2;
r2 = nu*dt/ds^2;

X      = zeros(NsTot,NrTot);
Y      = zeros(NsTot,NrTot);

for k=1:NsTot
    for j=1:NrTot
        loc  = mesh.grid{k,j};
        xloc = loc(1)*cos(loc(2));
        yloc = loc(1)*sin(loc(2));
        X(k,j)= xloc;
        Y(k,j)= yloc;
    end
end

u = zeros(NsTot,NrTot);
for k=1:NsTot
    for j=1:NrTot
        loc   = mesh.grid{k,j};
        rad   = loc(1);
        theta = loc(2);
        u(k,j)= getEx(rad,theta,tlim1,iOption);
    end
end

A   = zeros(NsTot*NrTot);
b   = zeros(NsTot*NrTot,1);
vnp = zeros(NsTot*NrTot,1);

for k=1:NsTot
    for j=1:NrTot
        row = mesh.DOF(k,j);
        if (k==ng&&j==ng)||(k==1&&j==NrTot)||...
                (k==NsTot&&j==1)||(k==NsTot&&j==NrTot) % corner check
             col = row;
             A(row,col) = 1;
        else
            if k==ng         % bottom boundary (Dirchlet)
                col = mesh.DOF(jas+1,j);
                A(row,row) = 1;
                A(row,col) = 1;
            elseif j==ng     % left boundary (Neumann)
                col = mesh.DOF(k,jar+1);
                A(row,row) = -1;
                A(row,col) = 1;
            elseif j==NrTot % right boundary (Neumann)
                col = mesh.DOF(k,jbr-1);
                A(row,row) = 1;
```

```
                A(row,col) = -1;
            elseif k==NsTot % top boundary (Dirchlet)
                col = mesh.DOF(jbs-1,j);
                A(row,row) = 1;
                A(row,col) = 1;
            else
                rjmk = mesh.grid{k,j-1}(1);
                rjk  = mesh.grid{k,j}(1);
                rjpk = mesh.grid{k,j+1}(1);

                rjmh = 0.5*(rjk+rjmk);
                rjph = 0.5*(rjk+rjpk);

                colm = mesh.DOF(k,j-1);
                colp = mesh.DOF(k,j+1);
                rowm = mesh.DOF(k-1,j);
                rowp = mesh.DOF(k+1,j);

                A(row,colm) = -(r1/(2*rjk))*rjmh;
                A(row,row)  = (1+ (r1/(2*rjk))*(rjmh+rjph) + (r2/rjk^2));
                A(row,colp) = -(r1/(2*rjk))*rjph;
                A(row,rowm) = -(r2/(2*rjk^2));
                A(row,rowp) = -(r2/(2*rjk^2));
            end
        end
    end
end

for i=2:length(t)
    uold = u;
    for k=1:NsTot
        for j=1:NrTot
            row  = mesh.DOF(k,j);
            rjk  = mesh.grid{k,j}(1);
            tjk  = mesh.grid{k,j}(2);
            if (k==ng&&j==ng)||(k==ng&&j==NrTot)||...
                    (k==NsTot&&j==ng)||(k==NsTot&&j==NrTot) % corners
                b(row) = getEx(rjk,tjk,t(i),iOption);
            else
                if k==ng        % bottom Boundary condition
%                     b(row) = 2*getEx(rjk,slim1,t(i),iOption);
                    b(row) = 2*getBC1(rjk,t(i),iOption);
                elseif j==ng    % left Boundary condition
                    b(row) = 2*dr*getUr(rlim1,tjk,t(i),iOption);
                elseif j==NrTot % right Boundary condition
                    b(row) = 2*dr*getUr(rlim2,tjk,t(i),iOption);
                elseif k==NsTot % top Boundary condition
%                     b(row) = 2*getEx(rjk,slim2,t(i),iOption);
                    b(row) = 2*getBC2(rjk,t(i),iOption);
                else
                    rjmk = mesh.grid{k,j-1}(1);
                    rjpk = mesh.grid{k,j+1}(1);

                    rjmh = 0.5*(rjk+rjmk);
                    rjph = 0.5*(rjk+rjpk);

                    fnp  = getF(nu,rjk,tjk,t(i),iOption);
                    fn   = getF(nu,rjk,tjk,t(i-1),iOption);
                    Fhat = 0.5*(fnp+fn);

                    b(row) = (0.5*r1/rjk)*rjmh*uold(k,j-1) + ...
```

```
                                (1- (0.5*r1/rjk)*(rjmh+rjph) - (r2/rjk^2))*uold(k,j) + ...
                                (0.5*r1/rjk)*rjph*uold(k,j+1) + ...
                                (r2/(2*rjk^2))*uold(k-1,j) + ...
                                (r2/(2*rjk^2))*uold(k+1,j) + dt*Fhat;
                end
            end
        end
    end
    vnp = A\b;
    u   = reconstructSol(mesh.IDX,vnp);
end

t = '$t_f=';
t = strcat(t,num2str(tf));
t = strcat(t,'s$');
%
% figure
% surf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),u(jas:jbs,jar:jbr));
% colorbar
% xlabel('$x$','Interpreter','latex');
% ylabel('$y$','Interpreter','latex');
% zlabel('$u(r,\theta,t)$','Interpreter','latex');
% title('$Numerical Solution, \ u(r,\theta,t)$',t,'Interpreter','latex');
%
% figure
% contourf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),u(jas:jbs,jar:jbr));
% colorbar
% xlabel('$x$','Interpreter','latex');
% ylabel('$y$','Interpreter','latex');
% zlabel('$u(r,\theta,t)$','Interpreter','latex');
% title('Numerical Solution, $\ u(r,\theta,t)$',t,'Interpreter','latex');

ud = u(jas:jbs,jar:jbr);

if iOption==1
    uex = zeros(NsTot,NrTot);
    for k=1:NsTot
        for j=1:NrTot
            loc   = mesh.grid{k,j};
            rad   = loc(1);
            theta = loc(2);
            uex(k,j)= getEx(rad,theta,tf,iOption);
        end
    end
    err = u-uex;
    max_err = max(max(abs(err(jas:jbs,jar:jbr))));

%     figure
%     surf(err)

else
    max_err = 0;
    uex = ud;
end

end

%% functions

function uex = getEx(r,theta,t,iOption)
```

```
if iOption==1
    uex = cos(pi*r)*sin(2*theta)*sin(t);
else
    uex = 0;
end
end

function ubc1 = getBC1(r,t,iOption)
if iOption==1
    ubc1 = cos(pi*r)*sin(t)*0;
else
    ubc1 = 0;
end
end

function ubc2 = getBC2(r,t,iOption)
if iOption ==1
    ubc2 = cos(pi*r)*sin(t)*0;
else
    ubc2 = (r-1)^2*(r-2)^2*(t^0);
end
end

function ur = getUr(r,theta,t,iOption)
if iOption==1
    ur = -pi*sin(pi*r)*sin(2*theta)*sin(t);
else
    ur = 0;
end
end

function ut = getUt(r,theta,t,iOption)
if iOption==1
    ut = cos(pi*r)*sin(2*theta)*cos(t);
else
    ut = 0;
end
end

function Vrr = getVrr(r,theta,t,iOption)
if iOption==1
    Vrr = -pi*sin(2*theta)*sin(t)*((sin(pi*r)/r)+pi*cos(pi*r));
else
    Vrr = 0;
end
end

function Vtt = getVtt(r,theta,t,iOption)
if iOption==1
    Vtt = -(4/r^2)*cos(pi*r)*sin(2*theta)*sin(t);
else
    Vtt = 0;
end
end

function f = getF(nu,r,theta,t,iOption)
if iOption==1
    ut  = getUt(r,theta,t,iOption);
    Vrr = getVrr(r,theta,t,iOption);
    Vtt = getVtt(r,theta,t,iOption);

```

```
    f = ut - nu*(Vrr+Vtt);
else
    f = 0;
end
end
```

Listing 3: Heat Equation 2D under domain mapping

```
function u = reconstructSol(IDX,v)

for i=1:length(v)
    idx     = IDX{i};
    k       = idx(1);
    j       = idx(2);
    u(k,j)  = v(i);
end

end
```

Listing 4: Reconstructing solution code

(b) Verify the accuracy of your code using manufactured solutions.

$$
\begin{aligned}
u\left(r,\theta,t\right) &= \cos\left(\pi r\right)\sin 2\theta\sin t,\ r\in\left(1,2\right),\ \theta\in\left(0,\frac{\pi}{2}\right)\\
u_r\left(r,\theta,t\right) &= -\pi\sin\left(\pi r\right)\sin 2\theta\sin t\\
u_t\left(r,\theta,t\right) &= \cos\left(\pi r\right)\sin 2\theta\cos t\\
\frac{1}{r}\left(ru_r\right)_r &= -\pi\sin 2\theta\sin t\left(\frac{\sin\pi r}{r}+\pi\cos\pi r\right)\\
\frac{1}{r^2}\left(u_{\theta\theta}\right) &= -\frac{4}{r^2}\cos\left(\pi r\right)\sin 2\theta\sin t
\end{aligned}
$$

$$
f(r,\theta,t) = u_t - \nu\left(\frac{1}{r}\left(ru_r\right)_r + \frac{1}{r^2}u_{\theta\theta}\right)
$$

Boundary conditions are,

$$
\begin{aligned}
u_r\left(r=1\right) &= 0, & u\left(\theta=0\right) &= 0\\
u_r\left(r=2\right) &= 0, & u\left(\theta=\frac{\pi}{2}\right) &= 0
\end{aligned}
$$

The order of convergence is shown in Fig 2.

(c) Now set

$$
\begin{aligned}
u_0(r,\theta) &= 0\\
\alpha_1(\theta,t) &= 0\\
\alpha_2(\theta,t) &= 0\\
\beta_1(r,t) &= 0\\
\beta_2(r,t) &= (r-1)^2(r-2)^2.
\end{aligned}
$$

Using $\nu = 1$ and 40 grid lines in both the radial and angular coordinate directions, compute numerical solutions to this problem at $t = 0, .1, .5, 1.5$. Create surface plots of the solution for each time.
The solutions are plotted in Fig 3 and Fig 4.

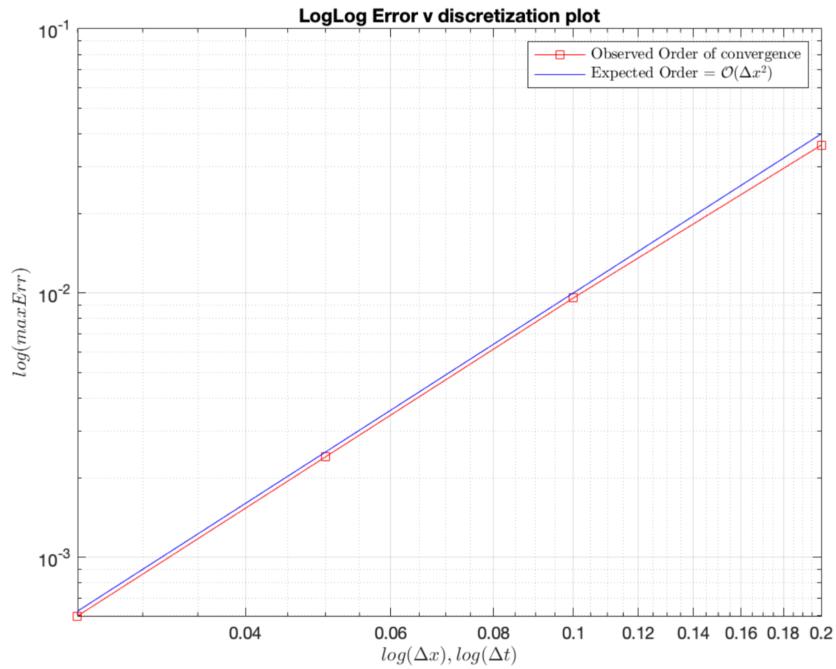


Figure 2: Order of Convergence

(d) Create a single line plot with four curves showing the solutions from part (c) along the inner radius ($r = 1$), as a function of θ for all four times $t = 0, .1, .5, 1.5$.
The 1-D slice of the function is shown in Fig 5 and Fig 6.

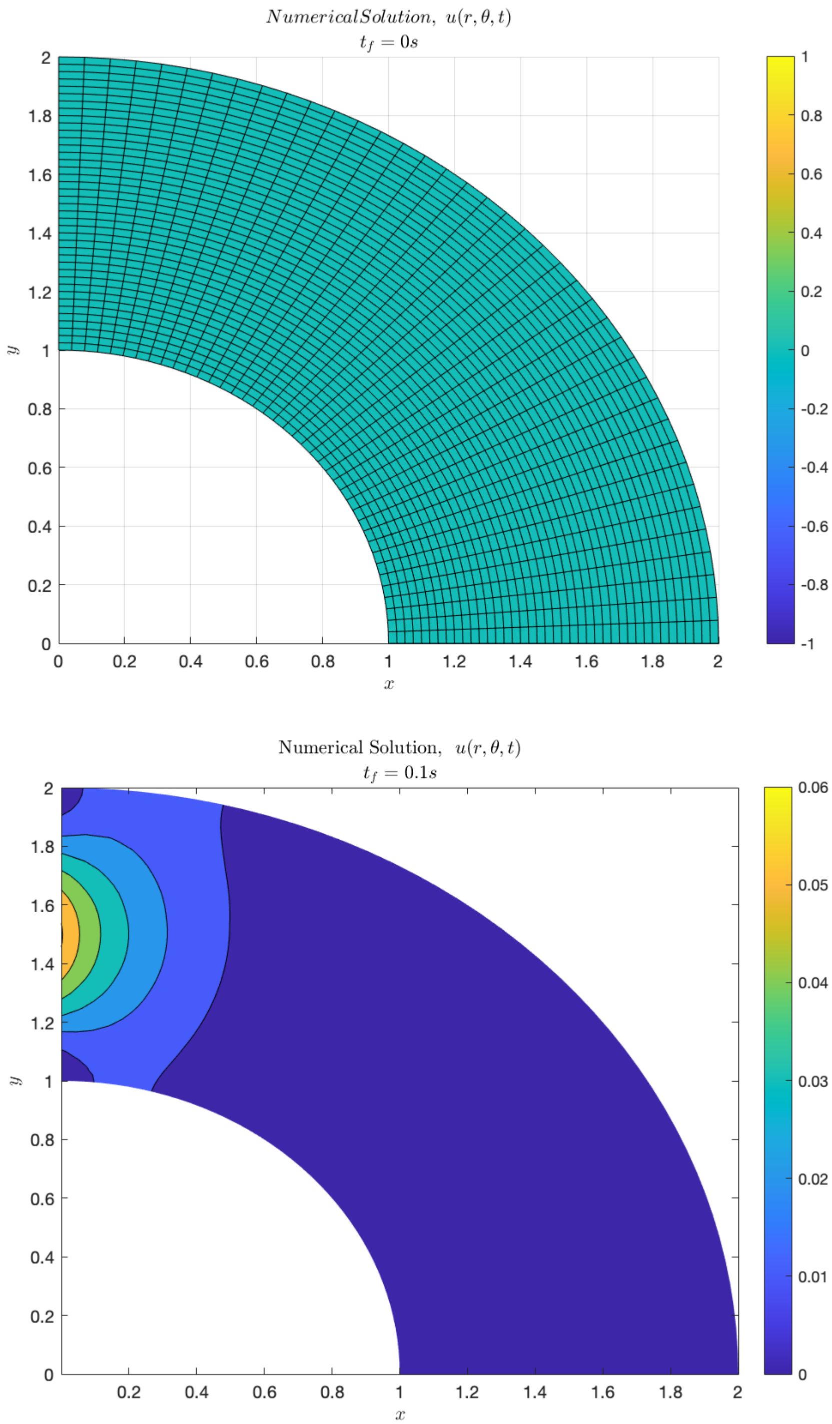


Figure 3: Solution at different final times

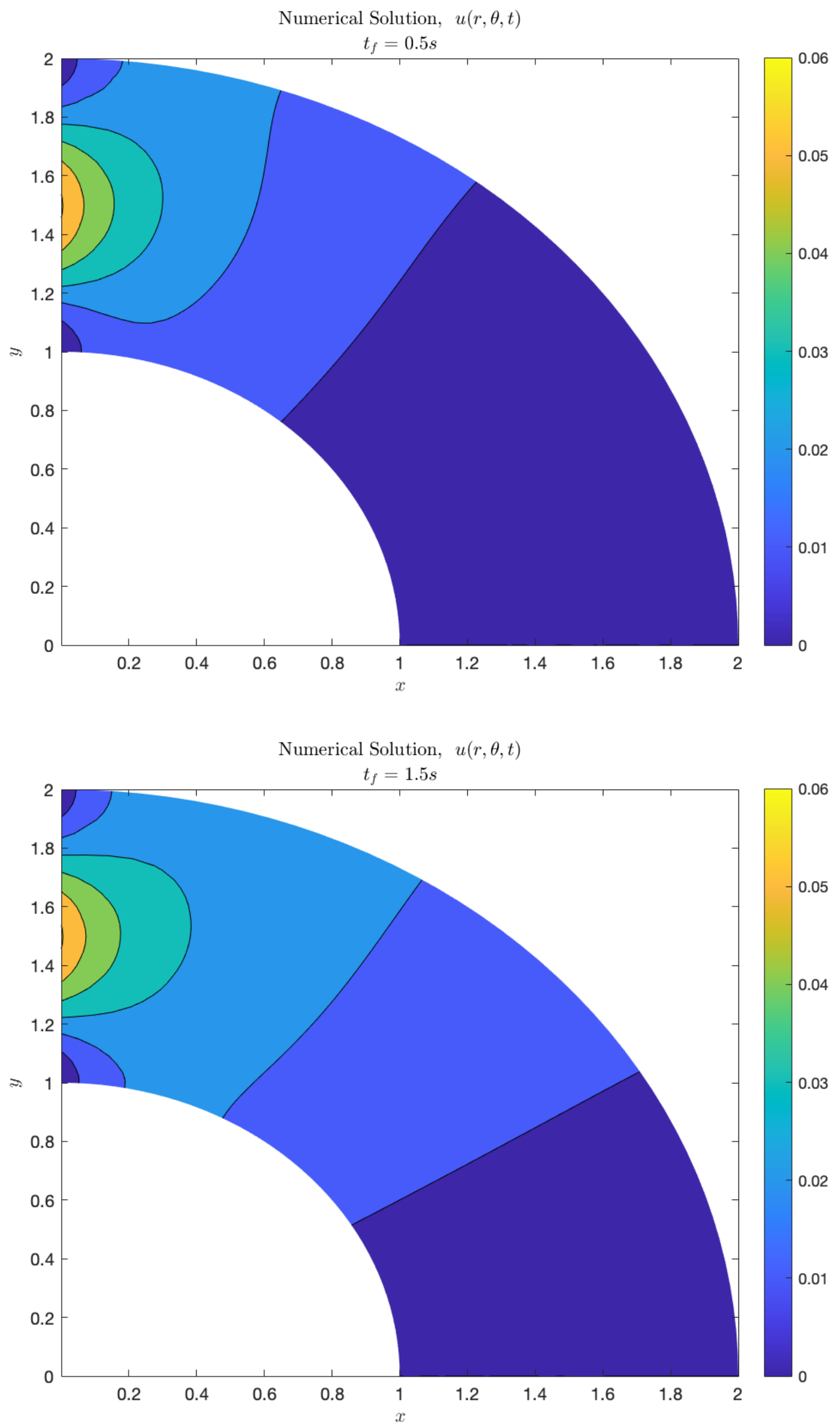


Figure 4: Solution at different final times

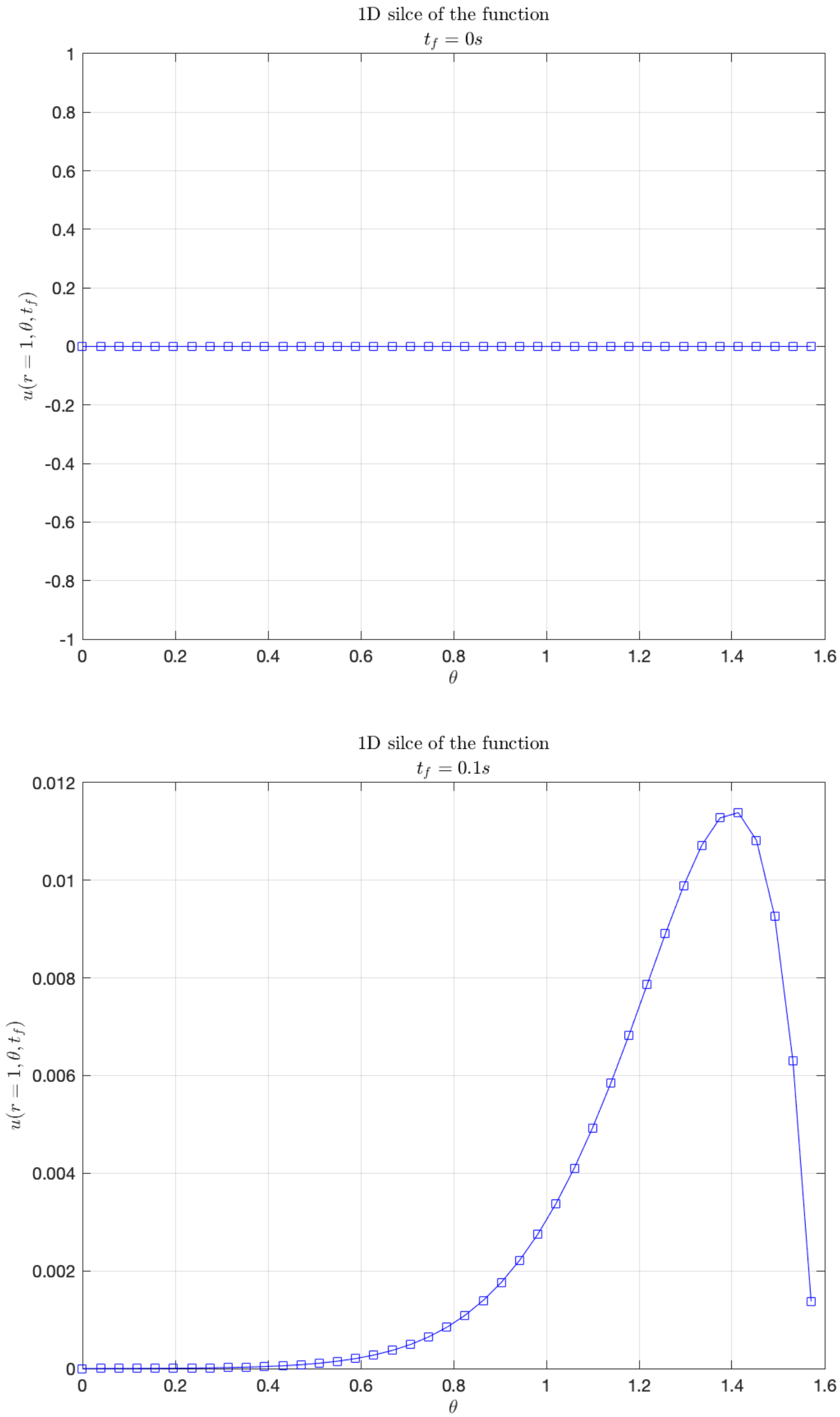
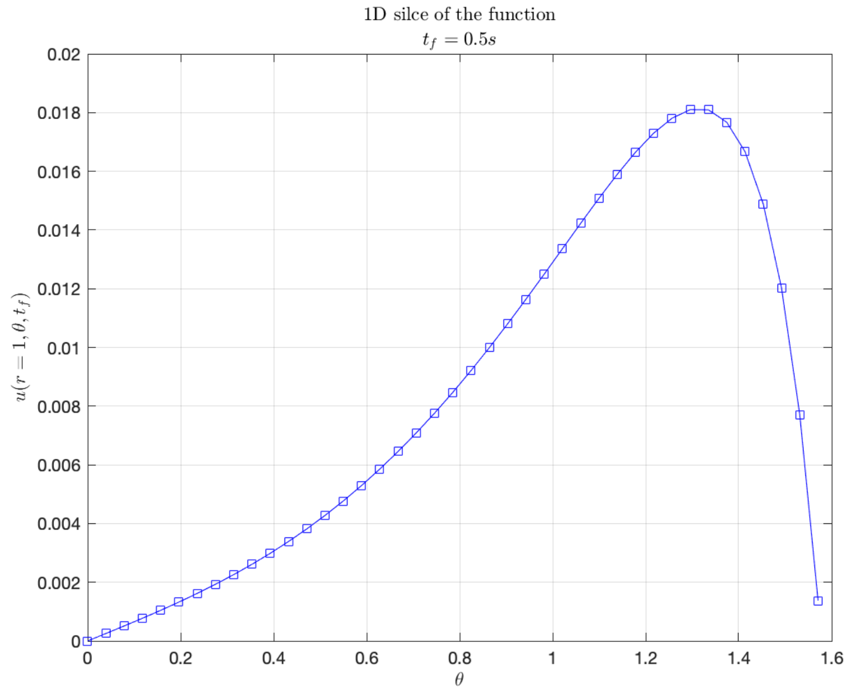


Figure 5: 1D slice of solution at final time - 1



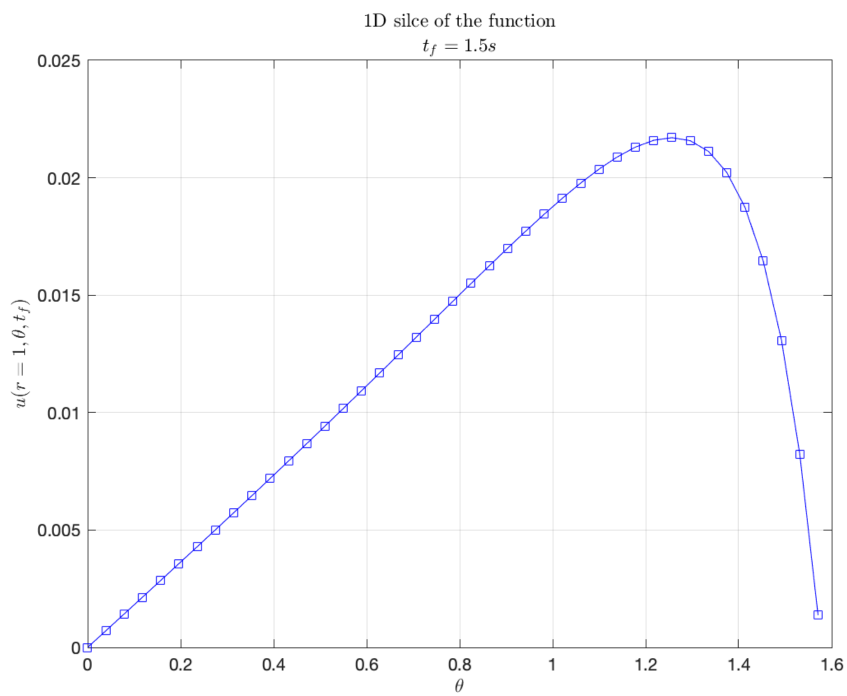


Figure 6: 1D slice of solution at final time - 2