MATH-6840
Vignesh Ramakrishnan
Due: Thursday April 7, 2022

Problem Set 8

1. (25 pts.) Consider the wave equation

$$u_{tt} = c^2 u_{xx}, \qquad t > 0$$

with initial conditions $u(x,0) = f(x)$ and $u_t(x,0) = g(x)$ (boundary conditions will be added later).

(a) Derive a sixth-order accurate (in space and time) discretization of this equation using centered spatial differences and the 3-level modified equation time stepper discussed in class. Hint: Useful discretizations may be found on the 2nd page of this document.

$$u_{tt} = c^2 u_{xx}$$
$$u_{tttt} = c^2 (u_{xx})_{tt} = c^2 (u_{tt})_{xx} = c^4 u_{xxxx}$$

This can be generalized and written as, $u_{(2j)t} = c^{(2j)} u_{(2j)x}$ where $j = 1, 2, 3, ...$

$$u_{tt} = D_{+t}D_{-t}u_j^n - \left(\frac{\Delta t^2}{12}u_{tttt} + \frac{\Delta t^4}{360}u_{6t}\right) = c^2 u_{xx}$$

$$D_{+t}D_{-t}v_j^n - \frac{\Delta t^2}{12}u_{tttt} - \frac{\Delta t^4}{360}u_{6t} = c^2 \frac{2v_{j+3}^n - 27v_{j+2}^n + 270v_{j+1}^n - 490v_j^n + 270v_{j-1}^n - 27v_{j-2}^n + 2v_{j-3}^n}{180\Delta x^2}$$

Simplifying this with $\sigma = \frac{c\Delta t}{\Delta x}$ becomes,

$$\begin{aligned}
v_j^{n+1} - 2v_j^n + v_j^{n-1} &= \frac{\sigma^2}{180}\left(2v_{j+3}^n - 27v_{j+2}^n + 270v_{j+1}^n - 490v_j^n + 270v_{j-1}^n - 27v_{j-2}^n + 2v_{j-3}^n\right) \\
&+ \frac{\Delta t^2}{12}c^4 u_{xxxx} + \frac{\Delta t^4}{360}c^6 u_{6x} \\
v_j^{n+1} - 2v_j^n + v_j^{n-1} &= \frac{\sigma^2}{180}\left(2v_{j+3}^n - 27v_{j+2}^n + 270v_{j+1}^n - 490v_j^n + 270v_{j-1}^n - 27v_{j-2}^n + 2v_{j-3}^n\right) \\
&+ \frac{\sigma^4}{72}\left(-v_{j+3}^n + 12v_{j+2}^n - 39v_{j+1}^n + 56v_j^n - 39v_{j-1}^n + 12v_{j-2}^n - v_{j-3}^n\right) \\
&+ \frac{\sigma^6}{360}\left(v_{j+3}^n - 6v_{j+2}^n + 15v_{j+1}^n - 20v_j^n + 15v_{j-1}^n - 6v_{j-2}^n + v_{j-3}^n\right) \\
&j = 0, 1, 2, \ldots\ldots N
\end{aligned}$$

(b) Using Fourier mode analysis, derive an expression for the amplification factors. Create a surface plot of the magnitude of each of the two roots for $\sigma = c\Delta t/\Delta x \in [-1.1, 1.1]$ and for the discrete wave number $\xi \in [-\pi, \pi]$.
Let $v_j^n = a^n e^{ikxj}$. Then,

$$\begin{aligned}
a - 2 + \frac{1}{a} &= \frac{\sigma^2}{180}\left(2e^{3ikx} - 27e^{2ikx} + 270e^{ikx} - 490 + 270e^{-ikx} - 27e^{-2ikx} + 2e^{-3ikx}\right) \\
&+ \frac{\sigma^4}{72}\left(-e^{3ikx} + 12e^{2ikx} - 39e^{ikx} + 56 - 39e^{-ikx} + 12e^{-2ikx} - e^{-3ikx}\right) \\
&+ \frac{\sigma^6}{360}\left(e^{3ikx} - 6e^{2ikx} + 15e^{ikx} - 20 + 15e^{-ikx} - 6e^{-2ikx} + e^{-3ikx}\right)
\end{aligned}$$

$$
\begin{aligned}
\frac{a^2-2a+1}{a} &= \frac{\sigma^2}{180}\left(4\cos(3kx)-54\cos(2kx)+540\cos(kx)-490\right)\\
&+\frac{\sigma^4}{72}\left(-2\cos(3kx)+24\cos(2kx)-78\cos(kx)+56\right)\\
&+\frac{\sigma^6}{360}\left(2\cos(3kx)-12\cos(2kx)+30\cos(kx)-20\right)
\end{aligned}
$$

Simplifying this equation it becomes,

$$
\begin{aligned}
a^2-2a+1 = 2a\left\{\left(\frac{\sigma^2}{90}-\frac{\sigma^4}{72}+\frac{\sigma^6}{360}\right)\cos(3\xi)+\left(\frac{-3\sigma^2}{20}+\frac{12\sigma^4}{72}-\frac{6\sigma^6}{360}\right)\cos(2\xi)\right\}+\\
2a\left\{\left(\frac{270\sigma^2}{180}-\frac{39\sigma^4}{72}+\frac{15\sigma^6}{360}\right)\cos(\xi)+\left(\frac{-245\sigma^2}{180}+\frac{28\sigma^4}{72}-\frac{\sigma^6}{36}\right)\right\}
\end{aligned}
$$

Here, $\xi = kx$ and this equation can be clubbed and written as $a^2-2b+1=0$ and $a = b \pm \sqrt{b^2-1}$. The plots of the magnitude of the amplitude $|a|$ can be seen in Fig

(c) Now restrict consideration to the finite domain $x \in [-1,1]$ with boundary conditions $u_x(-1,t)=\alpha(t)$, $u(1,t)=\beta(t)$. Using the computational grid defined by $x_j = -1 + j\Delta x$, $0 \le j \le N$, $\Delta x = 2/N$, introduce ghost cells as needed and define appropriate compatibility boundary conditions suitable for 6th order accuracy.

If 3 ghost points are introduced at either ends of the boundary, then the boundary condition at the right is given as,

$$
\begin{aligned}
u(1,t) &= \beta(t)\\
u_{tt}(1,t) &= \beta_{tt}(t)\\
c^2u_{xx}(1,t) &= \beta_{tt}(t) && \text{Equation 1}\\
c^4u_{xxxx}(1,t) &= \beta_{tttt}(t) && \text{Equation 2}\\
c^6u_{6x}(1,t) &= \beta_{6t}(t) && \text{Equation 3}
\end{aligned}
$$

This using the discretization available can be written as,

$$
c^2\frac{2v_{N+3}^n-27v_{N+2}^n+270v_{N+1}^n-490v_N^n+270v_{N-1}^n-27v_{N-2}^n+2v_{N-3}^n}{180\Delta x^2} = \beta_{tt}(t) \quad (1)
$$

$$
c^4\frac{-v_{N+3}^n+12v_{N+2}^n-39v_{N+1}^n+56v_N^n-39v_{N-1}^n+12v_{N-2}^n-v_{N-3}^n}{6\Delta x^4} = \beta_{tttt}(t) \quad (2)
$$

$$
c^6\frac{v_{N+3}^n-6v_{N+2}^n+15v_{N+1}^n-20v_N^n+15v_{N-1}^n-6v_{N-2}^n+v_{N-3}^n}{\Delta x^6} = \beta_{6t}(t) \quad (3)
$$

This leads to a system of Linear equations to solve for $v_{N+1}^n, v_{N+2}^n, v_{N+3}^n$.

$$
\begin{bmatrix} 2 & -27 & 270\\ -1 & 12 & -39\\ 1 & -6 & 15 \end{bmatrix}
\begin{bmatrix} v_{N+3}^n\\ v_{N+2}^n\\ v_{N+1}^n \end{bmatrix}
=
\begin{bmatrix}
\frac{180\Delta x^2}{c^2}\beta_{tt}(t)+490v_N^n-270v_{N-1}^n+27v_{N-2}^n+2v_{N-3}^n\\
\frac{6\Delta x^4}{c^4}\beta_{4t}(t)-56v_N^n+39v_{N-1}^n-12v_{N-2}^n+v_{N-3}^n\\
\frac{\Delta x^6}{c^6}\beta_{6t}(t)+20v_N^n-15v_{N-1}^n+6v_{N-2}^n-v_{N-3}^n
\end{bmatrix}
$$

Now similarly for the left boundary condition,

$$
\begin{aligned}
u_x(-1,t) &= \alpha(t) && \text{Equation 4}\\
(u_{tt})_x(-1,t) = c^2u_{xxx}(-1,t) &= \alpha_{tt}(t) && \text{Equation 5}\\
(u_{4t})_x(-1,t) = c^4u_{5x}(-1,t) &= \alpha_{5t}(t) && \text{Equation 6}
\end{aligned}
$$

Using the discretization available, it can be rewritten as,

$$\frac{v_3^n - 9v_2^n + 45v_1^n - 45v_{-1}^n + 9v_{-2}^n - v_{-3}^n}{60\Delta x} = \alpha(t) \tag{4}$$

$$\frac{-v_3^n + 8v_2^n - 13v_1^n + 13v_{-1}^n - 8v_{-2}^n + v_{-3}^n}{8\Delta x^3} = \alpha_{tt}(t) \tag{5}$$

$$\frac{v_3^n - 4v_2^n + 5v_1^n - 5v_{-1}^n + 4v_{-2}^n - v_{-3}^n}{2\Delta x^5} = \alpha_{4t}(t) \tag{6}$$

This is a system of linear equations which can be written as,

$$\begin{bmatrix} -45 & 9 & -1 \\ 13 & -8 & 1 \\ -5 & 4 & -1 \end{bmatrix} \begin{bmatrix} v_{-1}^n \\ v_{-2}^n \\ v_{-3}^n \end{bmatrix} = \begin{bmatrix} 60\Delta x\alpha(t) - v_3^n + 9v_2^n - 45v_1^n \\ \frac{8\Delta x^3}{c^2}\alpha_{tt}(t) + v_3^n - 8v_2^n + 13v_1^n \\ \frac{2\Delta x^5}{c^4}\alpha_{4t}(t) - v_3^n + 4v_2^n - 5v_1^n \end{bmatrix}$$

(d) Write a code implementing the sixth-order method. Perform a convergence study using the exact solution $u(x,t) = \sin(5(x-ct)) + \cos(2(x+ct))$ with $c = .9$.
It is important to find forcing first for this solution.

$$u_{tt} = c^2 u_{xx} + F(x,t)$$

But for this exact solution Forcing $F(x,t) = 0$. For $t = 0$,

$$\begin{aligned} u(x,0) &= \sin 5x + \cos 2x = f(x) \\ u_t(x,0) &= -5c\cos(5x) - 2c\sin 2x = g(x) \\ u_x(-1,t) &= 5\cos(-5-5ct) - 2\sin(-2+2ct) = \alpha(t) \\ u(1,t) &= \sin(5-5ct) + \cos(2+2ct) = \beta(t) \end{aligned}$$

Now, $\alpha_{tt}(t), \alpha_{4t}(t), \beta_{tt}(t), \beta_{4t}(t), \beta_{6t}(t)$ are all computed analytically and then the boundary conditions are set.

```
function max_err = WaveEqn1DOrder6(N,s,tf,c,fOption,mtd)

% defined constants
xlim1 = -1;
xlim2 = 1;
tlim1 = 0;
tlim2 = tf;

% calculated parameters
dx = (xlim2-xlim1)/N;
dt = s*dx/c;

ng   = 3;
NTot = N+1+2*ng;
ja   = ng+1;
jb   = NTot-ng;

x  = (xlim1-ng*dx:dx:xlim2+ng*dx);
t  = (tlim1:dt:tlim2);

% set up solution arrays
unp1 = zeros(NTot,1);
```

```
un    = zeros(NTot,1);
unm1 = zeros(NTot,1);

% set Initial conditions
for j=ja:jb
    f0      = f(x(j),c,fOption);
    unm1(j) = f0;

    % set values at un
    fm3     = f(x(j-3),c,fOption);
    fm2     = f(x(j-2),c,fOption);
    fm1     = f(x(j-1),c,fOption);
    fp1     = f(x(j+1),c,fOption);
    fp2     = f(x(j+2),c,fOption);
    fp3     = f(x(j+3),c,fOption);

    gm3     = g(x(j-3),c,fOption);
    gm2     = g(x(j-2),c,fOption);
    gm1     = g(x(j-1),c,fOption);
    g0      = g(x(j),c,fOption);
    gp1     = g(x(j+1),c,fOption);
    gp2     = g(x(j+2),c,fOption);
    gp3     = g(x(j+3),c,fOption);

    ut      = g0;
    utt     = c^2*cUxx_Order6(fm3,fm2,fm1,f0,fp1,fp2,fp3,dx);
    uttt    = c^2*cUxx_Order6(gm3,gm2,gm1,g0,gp1,gp2,gp3,dx);
    u4t     = c^4*cU4x_Order4(fm3,fm2,fm1,f0,fp1,fp2,fp3,dx);
    u5t     = c^4*cU4x_Order4(gm3,gm2,gm1,g0,gp1,gp2,gp3,dx);
    u6t     = c^6*cU6x_Order2(fm3,fm2,fm1,f0,fp1,fp2,fp3,dx);

    un(j)   = f0+ (dt/1)*ut + (dt^2/2)*utt + (dt^3/6)*uttt + ...
              (dt^4/24)*u4t + (dt^5/120)*u5t + (dt^6/720)*u6t;
end

unm1 = setBC(unm1,t(1),c,ja,jb,dx,fOption);
un   = setBC(un,t(2),c,ja,jb,dx,fOption);

% find solution at new time level

for i=3:length(t)
    for j=ja:jb
        if mtd==1
            fn = getF(x(j),t(i),c,fOption);
            unp1(j) = 2*un(j)-unm1(j) + s^2*(un(j+1)-2*un(j)+un(j-1))...
                      -(s^2/12)*(1-s^2)*(un(j+2)-4*un(j+1)+6*un(j)-4*un(j-1)+un(j-2))...
                      -(s^2/360)*(1-s^4)*(un(j+3)-6*un(j+2)+15*un(j+1)...
                                          -20*un(j)+15*un(j-1)-6*un(j-2)+un(j-3)) + dt^2*fn;
        elseif mtd==2
            fn = getF(x(j),t(i),c,fOption);
            unp1(j) = 2*un(j)-unm1(j)+...
                (s^2/180)*(2*un(j+3)-27*un(j+2)+270*un(j+1)-490*un(j)+270*un(j-1)-27*un(j-2)+2*un(j-3))+...
                (s^4/72)*(-un(j+3)+12*un(j+2)-39*un(j+1)+56*un(j)-39*un(j-1)+12*un(j-2)-un(j-3))+...
                (s^6/360)*(un(j+3)-6*un(j+2)+15*un(j+1)-20*un(j)+15*un(j-1)-6*un(j-2)+un(j-3)) + dt^2*fn;
```

```
        end
    end
    unp1 = setBC(unp1,t(i),c,ja,jb,dx,fOption);

    unm1 = un;
    un   = unp1;

end


uex = zeros(NTot,1);
err = zeros(NTot,1);
for j=ja:jb
    uex(j,1) = getEx(x(j),t(end),c,fOption);
end
uex = setBC(uex,t(end),c,ja,jb,dx,fOption);

err(ja:jb) = unp1(ja:jb)-uex(ja:jb);

max_err = max(abs(err));

end

%% setting Boundary conditions
function uout = setBC(uin,t,c,ja,jb,dx,fOption)

uout = uin;
% Left Boundary condition
[a,att,a4t] = alpha(t,c,fOption);

A = [-45 9 -1;13 -8 1;-5 4 -1];
b = [(60*dx*a)-uin(ja+3)+9*uin(ja+2)-45*uin(ja+1);...
     (8*dx^3*att/c^2)+uin(ja+3)-8*uin(ja+2)+13*uin(ja+1);...
     (2*dx^5*a4t/c^4)-uin(ja+3)+4*uin(ja+2)-5*uin(ja+1)];
v = A\b;
uout(ja-1) = v(1);
uout(ja-2) = v(2);
uout(ja-3) = v(3);

% Right Boundary condition
[b,btt,b4t,b6t] = beta(t,c,fOption);
uin(jb)    = b;
uout(jb)   = uin(jb);
A = [2 -27 270;-1 12 -39;1 -6 15];
b = [(180*dx^2*btt/c^2)+490*uin(jb)-270*uin(jb-1)+27*uin(jb-2)-2*uin(jb
    -3);...
     (6*dx^4*b4t/c^4)-56*uin(jb)+39*uin(jb-1)-12*uin(jb-2)+uin(jb-3);...
     (dx^6*b6t/c^6)+20*uin(jb)-15*uin(jb-1)+6*uin(jb-2)-uin(jb-3)];
v = A\b;
uout(jb+3) = v(1);
uout(jb+2) = v(2);
uout(jb+1) = v(3);

end

%% functions

function uex = getEx(x,t,c,fOption)
if fOption==1
```

```
    uex = sin(5*(x-c*t)) + cos(2*(x+c*t));
else
    uex = 0;
end
end

function force = getF(x,t,c,fOption)
if fOption==1
    force = (c^2-c^2)*( 25*sin(5*(x-c*t)) + 4*cos(2*(x+c*t)) );
else
    force = 0;
end
end

function v = f(x,c,fOption)
if fOption == 1
    v = sin(5*x)+cos(2*x);
else
    v = 0*c;
end
end

function v = g(x,c,fOption)
if fOption==1
    v = -5*c*cos(5*x) - 2*c*sin(2*x);
else
    v = 0;
end
end

function [v,vtt,v4t] = alpha(t,c,fOption)
if fOption==1
    v   = 5*cos(-5-5*c*t) -2*sin(-2+2*c*t);
    vtt = -125*c^2*cos(-5-5*c*t) + 8*c^2*sin(-2+2*c*t);
    v4t = (5^5)*(c^4)*cos(-5-5*c*t) - (2^5)*(c^4)*sin(-2+2*c*t);
else
    v   = 0;
    vtt = 0;
    v4t = 0;
end
end

function [v,vtt,v4t,v6t] = beta(t,c,fOption)
if fOption==1
    A   = 5-5*c*t;
    B   = 2+2*c*t;
    a   = -5*c;
    b   = 2*c;
    v   = sin(A) + cos(B);
    vtt = -sin(A)*a^2 - cos(B)*b^2;
    v4t = sin(A)*a^4 + cos(B)*b^4;
    v6t = -sin(A)*a^6 - cos(B)*b^6;
else
    v   = 0;
    vtt = 0;
    v4t = 0;
    v6t = 0;
end
end
```

```
%% derivatives
function ux = cUx_Order6(um3,um2,um1,up1,up2,up3,dx)
ux = (up3-9*up2+45*up1-45*um1+9*um2-um3)/(60*dx);
end

function uxx = cUxx_Order6(um3,um2,um1,u,up1,up2,up3,dx)
uxx = (2*up3-27*up2+270*up1-490*u+270*um1-27*um2+2*um3)/(180*dx^2);
end

function uxxx = cUxxx_Order4(um3,um2,um1,up1,up2,up3,dx)
uxxx = (-up3+8*up2-13*up1+13*um1-8*um2+um3)/(8*dx^3);
end

function u4x = cU4x_Order4(um3,um2,um1,u,up1,up2,up3,dx)
u4x = (-up3+12*up2-39*up1+56*u-39*um1+12*um2-um3)/(6*dx^4);
end

function u5x = cU5x_Order2(um3,um2,um1,up1,up2,up3,dx)
u5x = (up3-4*up2+5*up1-5*um1+4*um2-um3)/(2*dx^5);
end

function u6x = cU6x_Order2(um3,um2,um1,u,up1,up2,up3,dx)
u6x = (up3-6*up2+15*up1-20*u+15*um1-6*um2+um3)/(dx^6);
end
```

Listing 1: Wave Equation - 6th order scheme

The convergence analysis of this scheme is displayed in Figure 1.

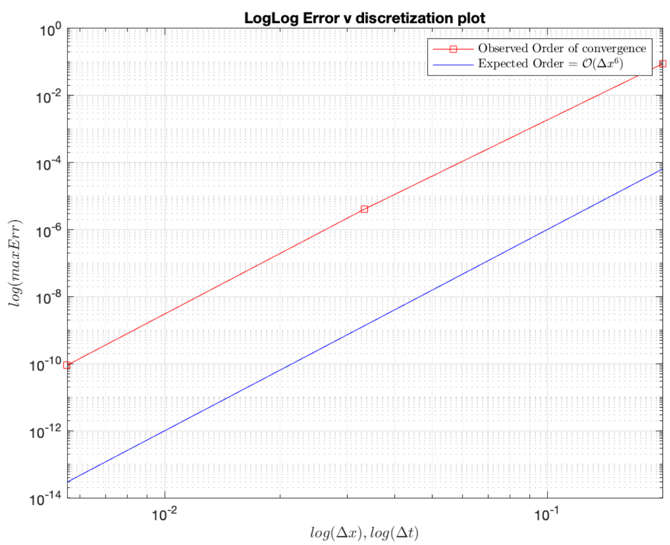

Figure 1: Order of Convergence

2. (25 pts.) Consider the wave propagation problem in an annular section

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

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

$$u\left(\frac{1}{2},\theta,t\right) = 0, \qquad u_r\left(1,\theta,t\right) = 0$$
$$u\left(r,-\frac{\pi}{2},t\right) = 0 \qquad u_\theta\left(r,\frac{\pi}{2},t\right) = 0.$$

(a) Write a second-order accurate code to solve this problem using centered differencing and the 3-level modified equation time stepper discussed in class. That is to say you must to treat this as a variable coefficient operator rather than performing a chain rule (e.g. you must discretize $(ru_r)_r$ as it sits and **not** convert it to $u_r + ru_{rr}$). Note this code will have a maximal stable time step and your code will need to be constructed to satisfy this constraint.

If a rectangular discretization of the domain is considered, then,

$$u_{tt} = c^2\left(u_{xx} + u_{yy}\right)$$
$$v_{j,k}^{n+1} = 2v_{j,k}^n - v_{j,k}^{n-1} + \frac{c^2\Delta t^2}{\Delta x^2}\left(v_{j+1,k}^n - 2v_{j,k}^n + v_{j-1,k}^n\right) + \frac{c^2\Delta t^2}{\Delta y^2}\left(v_{j,k+1}^n - 2v_{j,k}^n + v_{j,k-1}^n\right)$$

Let $v_{j,k}^n = a^n e^{ik_1 xj} e^{ik_2 yk}$ and substituting this above results in,

$$a - 2 + \frac{1}{a} = \sigma_1^2\left(e^{ik_1x} - 2 + e^{-ik_1x}\right) + \sigma_2^2\left(e^{ik_2y} - 2 + e^{-ik_2y}\right)$$
$$\frac{a^2 - 2a + 1}{a} = 2\sigma_1^2\left(\cos\xi - 1\right) + 2\sigma_2^2\left(\cos\eta - 1\right)$$

Where, $\sigma_1 = \frac{c\Delta t}{\Delta x}, \sigma_2 = \frac{c\Delta t}{\Delta y}, \xi = k_1x, \eta = k_2y$.

$$a^2 - 2\left(1 - \sigma_1^2\left(1-\cos\xi\right) - \sigma_2^2\left(1 - \cos\eta\right)\right)a + 1 = 0$$

Let $b = 1 - \sigma_1^2\left(1-\cos\xi\right) - \sigma_2^2\left(1-\cos\eta\right)$.
For stability $b^2 \leq 1$

$$\begin{aligned}1 + \sigma_1^4(1-\cos\xi)^2 + &\sigma_2^4(1-\cos\eta)^2 - \\ &2\sigma_1^2(1-\cos\xi) - 2\sigma_2^2(1-\cos\eta) + \\ &2\sigma_1^2\sigma_2^2(1-\cos\xi)(1-\cos\eta) \leq 1\end{aligned}$$

This can be further simplified into

$$\left(\sigma_1^2(1-\cos\xi) + \sigma_2^2(1-\cos\eta)\right)^2 \leq 2\left(\sigma_1^2(1-\cos\xi) + \sigma_2^2(1-\cos\eta)\right)$$

The maximum value of this inequality occurs at $\cos\xi = -1$ and at $\cos\eta = -1$. Then it reduces to,

$$4\left(\sigma_1^2 + \sigma_2^2\right)^2 \leq 4\left(\sigma_1^2 + \sigma_2^2\right)$$
$$c^2\Delta t^2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \leq 1$$
$$\Delta t^2 \leq \frac{1}{c^2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}$$

If $\Delta x \le \Delta y$, then

$$\Delta t = \frac{\Delta x}{c\sqrt{2}}$$

In our case, we choose which ever is smaller between Δr and $r_1\Delta\theta$ and then choose $\Delta t = \frac{\Delta X}{c\sqrt{2.5}}$ where $(\Delta X = min\,(\Delta r, r_1\Delta\theta))$.

(b) Verify the accuracy of your code using a manufactured solution. Here you should use $N_\theta = 3N_r$ so that in physical space the grids are approximately square. Note that you will likely need to consider non-homogeneous boundary conditions since your exact solution may not satisfy the given BCs.

Let the exact solution be, $u(r,\theta,t) = \sin(\pi r)\cos\theta\sin(t)$. Then

$$F(r,\theta,t) = u_{tt} - c^2\left(\frac{1}{r}\left(ru_r\right)_r + \frac{1}{r^2}u_{\theta\theta}\right)$$

Boundary conditions are

$$\begin{aligned}
u(\frac{1}{2},\theta,t) &= \cos\theta\sin(t) = \alpha_1(\theta,t)\\
u_r(1,\theta,t) &= \pi\cos\pi\cos\theta\sin t = \alpha_2(\theta,t)\\
u(r,-\frac{\pi}{2},t) &= \sin(\pi r)\cos\frac{-\pi}{2}\sin t = \alpha_3(r,t)\\
u_\theta(r,\frac{\pi}{2},t) &= -\sin(\pi r)\sin\frac{\pi}{2}\sin t = \alpha_4(r,t)
\end{aligned}$$

The discretization is now written as,

$$\begin{aligned}
v_{j,k}^{n+1} = &\; 2v_{j,k}^n - v_{j,k}^{n-1} + \frac{\sigma_1^2}{r_{j,k}}\left(r_{j+\frac{1}{2},k}v_{j+1,k}^n - \left(r_{j+\frac{1}{2},k} + r_{j-\frac{1}{2},k}\right)v_{j,k}^n + r_{j-\frac{1}{2},k}v_{j-1,k}^n\right)\\
&+ \frac{\sigma_2^2}{r_{j,k}^2}\left(v_{j,k+1}^n - 2v_{j,k}^n + v_{j,k-1}^n\right)\\
&j = 0,1,2,....,N_r\\
&k = 0,1,2,....,N_\theta
\end{aligned}$$

$$\begin{aligned}
\frac{v_{-1,k}^n + v_{1,k}^n}{2} &= \alpha_1(\theta,t)\\
\frac{v_{N_r+1,k}^n - v_{N_r-1,k}^n}{2\Delta r} &= \alpha_2(\theta,t)\\
\frac{v_{j,-1}^n + v_{j,1}^n}{2} &= \alpha_3(r,t)\\
\frac{v_{j,N_\theta+1}^n - v_{j,N_\theta-1}^n}{2\Delta\theta} &= \alpha_4(r,t)
\end{aligned}$$

```
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

```
function [max_err,u,uexd,uini] = WaveEqn2DMapping(Nr,Ns,tf,c,fOption)

rlim1 = 0.5;
rlim2 = 1;
slim1 = -pi/2;
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;
```

```
% smalest dx
if dr< rlim1*ds
    dG = dr;
else
    dG = rlim1*ds;
end

dT = dG/(c*sqrt(2.5));

t  = (tlim1:dT:tf);

s1 = c*dT/dr;
s2 = c*dT/ds;

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

unm1 = zeros(NsTot,NrTot);
un   = zeros(NsTot,NrTot);
unp1 = zeros(NsTot,NrTot);

% set Initial conditions
for k=jas:jbs
    for j=jar:jbr
        rjk  = mesh.grid{k,j}(1);
        rjp = mesh.grid{k,j+1}(1);
        rjm = mesh.grid{k,j-1}(1);

        sk  = mesh.grid{k,j}(2);
        skp = mesh.grid{k+1,j}(2);
        skm = mesh.grid{k-1,j}(2);

        rjph = 0.5*(rjk+rjp);
        rjmh = 0.5*(rjk+rjm);

        fjk  = getIC1(rjk,sk,c,fOption);
        fjpk = getIC1(rjp,sk,c,fOption);
        fjmk = getIC1(rjm,sk,c,fOption);
        fjkp = getIC1(rjk,skp,c,fOption);
        fjkm = getIC1(rjk,skm,c,fOption);

        gjk  = getIC2(rjk,sk,c,fOption);

        Fjk  = getF(rjk,sk,t(1),c,fOption);

        utt  = c^2*((1/rjk)*(1/dr^2)*(rjph*fjpk-(rjph+rjmh)*fjk+rjmh*fjmk
    )+...
```

```
                (1/rjk^2)*(1/ds^2)*(fjkp-2*fjk+fjkm)) + Fjk;

        unm1(k,j) = fjk;
        un(k,j)   = fjk+dT*gjk+(dT^2/2)*utt;
    end
end
% set BCs
unm1 = setBC(unm1,t(1),c,mesh,fOption);
un   = setBC(un,t(2),c,mesh,fOption);

uini = unm1(jas:jbs,jar:jbr);

figure
surf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),unm1(jas:jbs,jar:jbr));
colorbar
shading interp
xlabel('$x$','Interpreter','latex');
ylabel('$y$','Interpreter','latex');
zlabel('$u(r,\theta,t)$','Interpreter','latex');
title('$Numerical Solution, \ u(r,\theta,t)$','Interpreter','latex');

% figure
for i=3:length(t)
    for k=jas:jbs
        for j=jar:jbr
            rjk  = mesh.grid{k,j}(1);
            sjk  = mesh.grid{k,j}(2);

            rjpk = mesh.grid{k,j+1}(1);
            rjmk = mesh.grid{k,j-1}(1);

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

            Fjk  = getF(rjk,sjk,t(i),c,fOption);

            unp1(k,j) = 2*un(k,j)-unm1(k,j)+...
                        (s1^2/rjk)*(rjph*un(k,j+1)-(rjph+rjmh)*un(k,j)+rjmh*un(k,j-1))+...
                        (s2^2/rjk^2)*(un(k+1,j)-2*un(k,j)+un(k-1,j))+dT^2*Fjk;
        end
    end
    unp1 = setBC(unp1,t(i),c,mesh,fOption);

    unm1 = un;
    un   = unp1;

%     surf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),unp1(jas:jbs,jar:jbr));
%     drawnow
%     pause(0.01)
end

str = '$t_f=';
str = strcat(str,num2str(tf));
str = strcat(str,'s$');

figure
surf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),unp1(jas:jbs,jar:jbr));
```

```
colorbar
shading interp
xlabel('$x$','Interpreter','latex');
ylabel('$y$','Interpreter','latex');
zlabel('$u(r,\theta,t)$','Interpreter','latex');
title('$Numerical Solution, \ u(r,\theta,t)$',str,'Interpreter','latex');

uex = zeros(NsTot,NrTot);
for k=jas:jbs
    for j=jar:jbr
        rad = mesh.grid{k,j}(1);
        the = mesh.grid{k,j}(2);
        uex(k,j) = getEx(rad,the,t(end),c,fOption);
    end
end
uex = setBC(uex,t(end),c,mesh,fOption);
err = - uex + unp1;
% figure
% surf(X(jas:jbs,jar:jbr),Y(jas:jbs,jar:jbr),err(jas:jbs,jar:jbr));

u    = unp1(jas:jbs,jar:jbr);
uexd = uex(jas:jbs,jar:jbr);
max_err = max(max(abs(err)));

end

%% functions to set Boundary conditions
function uout = setBC(uin,t,c,mesh,fOption)
uout = uin;

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;


% set Left BC and Right BC
for k=jas:jbs
    sjk = mesh.grid{k,jar}(2);

    % Left BC
    uout(k,jar-ng) = 2*getA1(sjk,t,c,fOption)-uin(k,jar+1);

    % Right BC
    uout(k,NrTot)  = 2*dr*getA2(sjk,t,c,fOption)+uin(k,jbr-1);
end
% set Bottom BC and Top BC
for j=jar:jbr
    rjk = mesh.grid{jas,j}(1);

    % Bottom BC
    uout(ng,j) = 2*getA3(rjk,t,c,fOption)-uin(jas+1,j);

    % Top BC
```

```
    uout(NsTot,j) = 2*ds*getA4(rjk,t,c,fOption)+uin(jas-1,j);
end

% set corners
uout(ng,ng)        = getEx(mesh.grid{ng,ng}(1),mesh.grid{ng,ng}(2),t,c,
    fOption);
uout(ng,NrTot)     = getEx(mesh.grid{ng,NrTot}(1),mesh.grid{ng,NrTot}(2),t
    ,c,fOption);
uout(NsTot,ng)     = getEx(mesh.grid{NsTot,ng}(1),mesh.grid{NsTot,ng}(2),t
    ,c,fOption);
uout(NsTot,NrTot) = getEx(mesh.grid{NsTot,NrTot}(1),mesh.grid{NsTot,NrTot
    }(2),t,c,fOption);

end

%% functions
function u = getIC1(r,s,c,fOption)
if fOption==1
    u = sin(5*r)+cos(2*s);
elseif fOption==2
    u = exp(-100*((r-0.75)^2+s^2));
elseif fOption==3
    u = 0;
else
    u = 0*c;
end
end

function ut = getIC2(r,s,c,fOption)
if fOption==1
    ut = (-5*c)*cos(5*r) + (2*c)*sin(2*s);
elseif fOption ==2
    ut = 0;
elseif fOption==3
    ut = sin(pi*r)*cos(s);
else
    ut = 0;
end
end

function f = getF(r,s,t,c,fOption)
if fOption==1
    utt = getUtt(r,s,t,c,fOption);
    Vrr = getVrr(r,s,t,c,fOption);
    Vss = getVss(r,s,t,c,fOption);
    f   = utt-c^2*(Vrr+Vss);
elseif fOption==3
    utt = getUtt(r,s,t,c,fOption);
    Vrr = getVrr(r,s,t,c,fOption);
    Vss = getVss(r,s,t,c,fOption);
    f   = utt-c^2*(Vrr+Vss);
else
    f = 0;
end
end

function a1 = getA1(s,t,c,fOption)
if fOption==1
    a1 = sin((5/2)-5*c*t)+cos(2*s-2*c*t);
```

```
elseif fOption==3
    a1 = cos(s)*sin(t);
else
    a1 = 0;
end
end


function a2 = getA2(s,t,c,fOption)
if fOption==1
    a2 = 5*cos(5-5*c*t);
elseif fOption==3
    a2 = pi*cos(pi)*cos(s)*sin(t);
else
    a2 = 0*s;
end
end

function a3 = getA3(r,t,c,fOption)
if fOption==1
    a3 = sin(5*r-5*c*t)+cos(-pi-2*c*t);
elseif fOption==3
    a3 = sin(pi*r)*cos(-pi/2)*sin(t);
else
    a3 = 0*r;
end
end


function a4 = getA4(r,t,c,fOption)
if fOption==1
    a4 = -2*sin(pi-2*c*t);
elseif fOption==3
    a4 = -sin(pi*r)*sin(pi/2)*sin(t);
else
    a4 = 0*r;
end
end

function uex = getEx(r,s,t,c,fOption)
if fOption==1
    uex = sin(5*r-5*c*t)+cos(2*s-2*c*t);
elseif fOption==3
    uex = sin(pi*r)*cos(s)*sin(t);
else
    uex = 0;
end
end

%% functions - 2
function utt = getUtt(r,s,t,c,fOption)
if fOption==1
    utt = (-5*c)^2*(-sin(5*r-5*c*t))+(-2*c)^2*(-cos(2*s-2*c*t));
elseif fOption==3
    utt = -sin(pi*r)*cos(s)*sin(t);
else
    utt = 0;
end
end
```

```
function Vrr = getVrr(r,s,t,c,fOption)
if fOption==1
    Vrr = (5/r)*cos(5*r-5*c*t)-25*sin(5*r-5*c*t);
elseif fOption==3
    Vrr = pi*cos(s)*sin(t)*((1/r)*cos(pi*r)-pi*sin(pi*r));
else
    Vrr = 0*s;
end
end

function Vss = getVss(r,s,t,c,fOption)
if fOption==1
    Vss = (-4/r^2)*cos(2*s-2*c*t);
elseif fOption==3
    Vss = (-1/r^2)*sin(pi*r)*cos(s)*sin(t);
else
    Vss = 0*r;
end
end
```

Listing 3: Wave Equation under 2D Mapping

The error plot looked smooth to me but I could not get second order convergence for some reason. It can be found in Fig 2

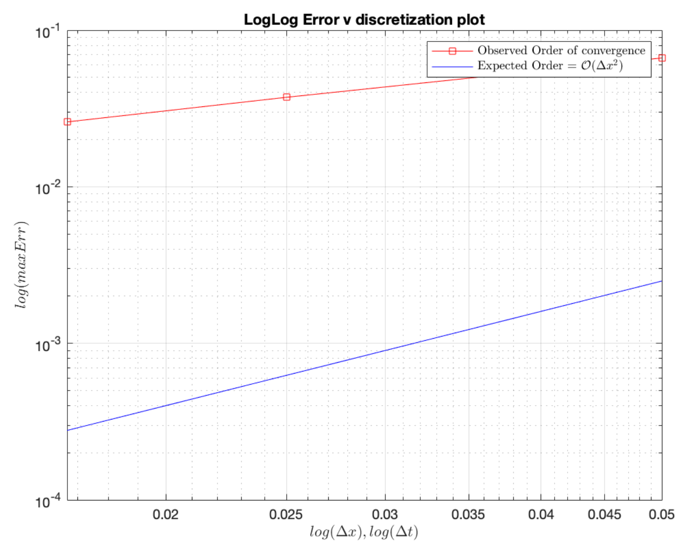

Figure 2: Error Convergence plot 2D Wave Equation

(c) Using $c = 1$, $Nr = 160$ and $N_\theta = 480$, compute numerical solutions to this problem using $f(r, \theta) = \exp(-100((r - 0.75)^2 + (\theta)^2))$, $g(r, \theta) = 0$ at $t = 0, .5, 1.5, 2.5$. Create surface plots of the solution for each time. In addition, create a single line plot with four curves showing the solution along the outer radius ($r = 1$), as a function of θ for all four times.
The solutions are found in Fig 3 and Fig 4.

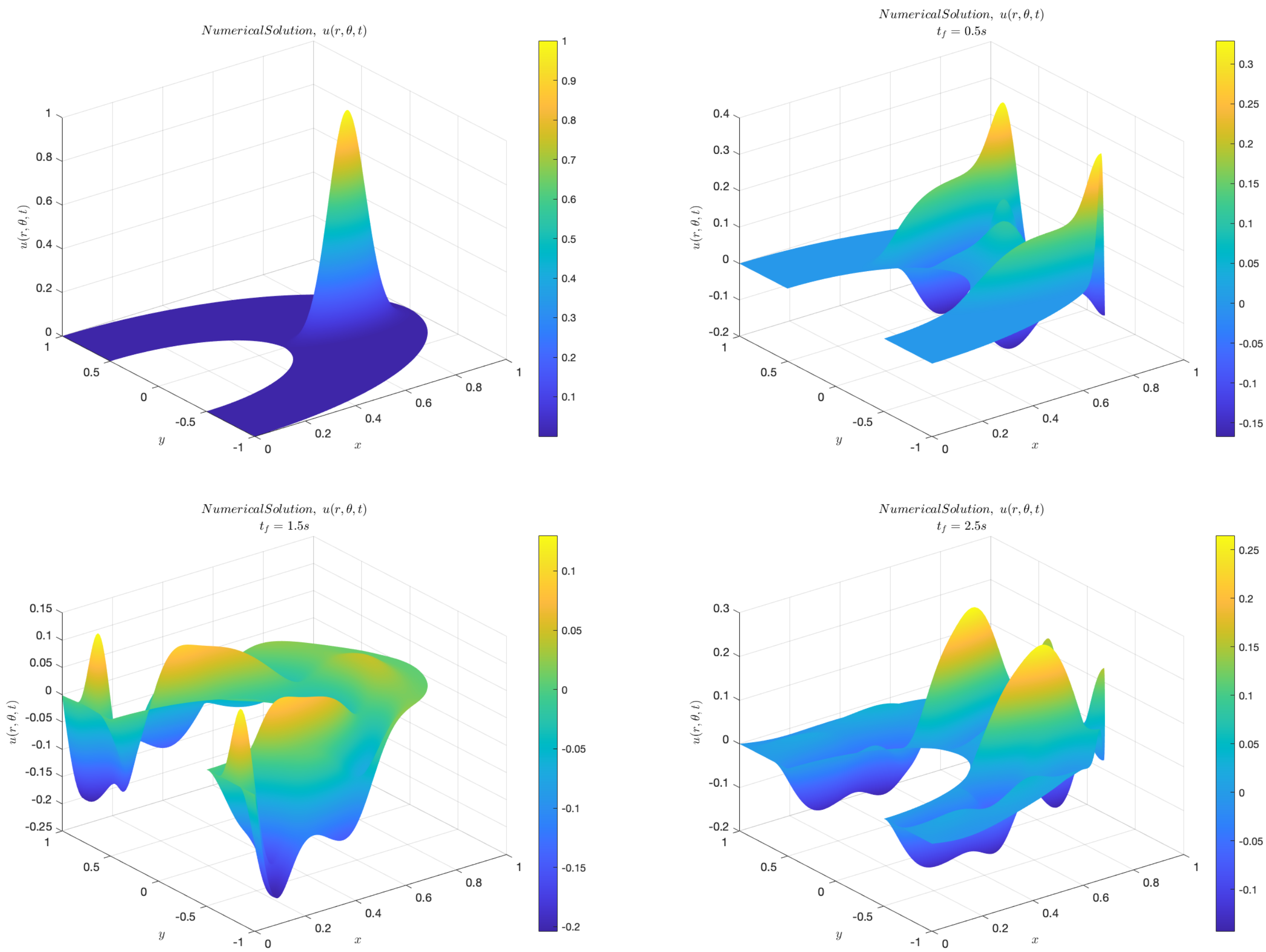

Figure 3: Solution at different final times

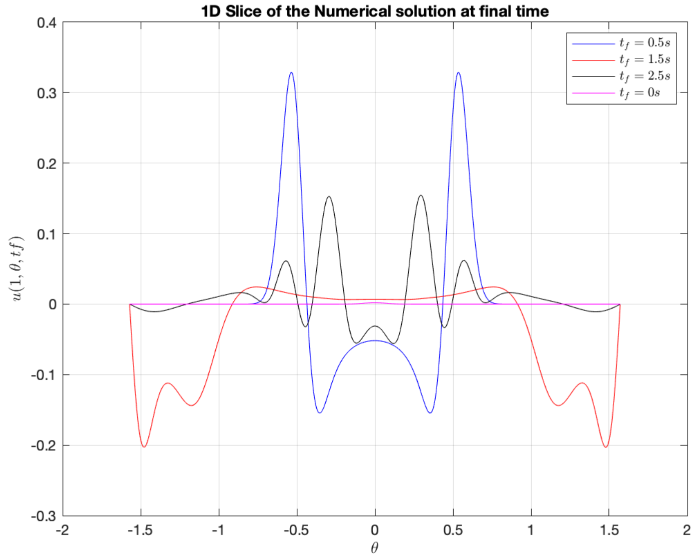

Figure 4: 1D slice of the solution at $r = 1$

The following discrete approximations may be useful for problem (1)

$$
\begin{aligned}
u_x(x_j) &= \frac{u_{j+3} - 9u_{j+2} + 45u_{j+1} - 45u_{j-1} + 9u_{j-2} - u_{j-3}}{60\Delta x} + O(\Delta x^6) \\
u_{xx}(x_j) &= \frac{2u_{j+3} - 27u_{j+2} + 270u_{j+1} - 490u_j + 270u_{j-1} - 27u_{j-2} + 2u_{j-3}}{180\Delta x^2} + O(\Delta x^6) \\
u_{xxx}(x_j) &= \frac{-u_{j+3} + 8u_{j+2} - 13u_{j+1} + 13u_{j-1} - 8u_{j-2} + u_{j-3}}{8\Delta x^3} + O(\Delta x^4) \\
u_{xxxx}(x_j) &= \frac{-u_{j+3} + 12u_{j+2} - 39u_{j+1} + 56u_j - 39u_{j-1} + 12u_{j-2} - u_{j-3}}{6\Delta x^4} + O(\Delta x^4) \\
u_{xxxxx}(x_j) &= \frac{u_{j+3} - 4u_{j+2} + 5u_{j+1} - 5u_{j-1} + 4u_{j-2} - u_{j-3}}{2\Delta x^5} + O(\Delta x^2) \\
u_{xxxxxx}(x_j) &= \frac{u_{j+3} - 6u_{j+2} + 15u_{j+1} - 20u_j + 15u_{j-1} - 6u_{j-2} + u_{j-3}}{\Delta x^6} + O(\Delta x^2)
\end{aligned}
$$