W.D. Henshaw **Math 6840: Problem Set 5** **Due:** Thurs. April 2, 2020.

General note: For computer exercises please provide listings of all computer programs that were used to generate the results. Please present your results in a clear and neat manner.

1. (15 points)

(a) Consider the modified advection equation on an infinite domain,

$$\begin{aligned} &\partial_t u + c\partial_x u = \alpha u, \qquad -\infty < x < \infty, \quad t > 0, \\ &u(x,0) = u_0(x), \qquad\qquad -\infty < x < \infty, \end{aligned}$$

where c and α are real constants. Find the solution to this problem for arbitrary $u_0(x)$ by using the method of characteristics.

(b) Derive a second-order accurate Lax-Wendroff scheme for the problem in (a) by using a Taylor series in time and using the PDE to replace time derivatives by space derivatives.

(c) Find the modified equation for the Lax-Wendroff scheme applied to the advection equation, $u_t + cu_x = 0$. The Lax-Wendroff scheme is

$$D_{+t}U_i^n + cD_0U_i^n - \frac{c^2\Delta t}{2}D_+D_-U_i^n = 0.$$

Expand the modified equation as the advection equation plus one correction term. Comment on what the modified equations says about well resolved solutions to the discrete solution of the Lax-Wendroff scheme.

2. (30 points) Upwind schemes for the advection equation. Consider again the problem from the previous homework for the initial value problem for the advection equation,

$$\begin{aligned} &u_t + cu_x = 0, \qquad 0 \le x < 2\pi, \quad 0 \le t \le t_{\text{final}}, \\ &u(x,0) = u_0(x), \\ &u(x+2\pi,t) = u(x,t), \end{aligned}$$

with periodic boundary conditions, and where $c > 0$. Discretize on a uniform grid $x_j = j\Delta x$, $j = -2, -1, 0, \ldots, N_x$ with $\Delta x = 2\pi/N_x$. Use TWO ghost points on the left to implement the periodic boundary conditions.

Consider solving this problem using

(i) The first-order upwind scheme UPW1 as discussed in class,

$$D_{+t}U_i^n + cD_-U_i^n = 0.$$

(ii) A second-order accurate upwind scheme that uses the spatial approximation

$$\partial_x u(x_i) \approx \mathcal{D}_-^{(2)}U_i = \frac{3U_i - 4U_{i-1} + U_{i-2}}{2\Delta x} \tag{1}$$

and the time-stepping scheme given by the 3rd-order Runge-Kutta scheme, RK3, which for the ODE $y' = f(y,t)$ is given by

$$\begin{aligned} Y^{n+1} &= Y^n + \frac{\Delta t}{6}(k_1 + k_2 + 4k_3), \\ k_1 &= f(Y^n, t^n), \\ k_2 &= f(Y^n + \Delta t k_1, t^n + \Delta t), \\ k_3 &= f(Y^n + (\Delta t/4)(k_1 + k_2), t^n + \Delta t/2), \end{aligned}$$

where $Y^n \approx y(t^n)$. Remember to apply the boundary conditions after each stage. Write a separate function to apply the boundary conditions to make the code cleaner.

(a) Find the symbol (Fourier transform) of the operator $\mathcal{D}_-^{(2)}$ and confirm that the approximation in equation (1) is second-order accurate by expanding the symbol for small $k\Delta x$.

(b) Check the accuracy of the RK3 scheme by finding the solution to the test equation $y' = \lambda y$, by looking for solutions of the form $Y^n = A^n Y^0$. Find the amplification factor A and show that $A \approx e^{\lambda \Delta t}$ to sufficient accuracy (Hint: the approximation should be locally fourth-order accurate in $\lambda \Delta t$).

(c) The region of absolute stability of the RK3 scheme can be approximated by the circle in the complex plane for $z = x + iy = \lambda_{\text{ts}} \Delta t$ (where λ_{ts} is the time-stepping eigenvalue)

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\alpha^2} \leq 1,$$

where $\alpha \approx 2.2$. Estimate the time-step restriction for the RK3-UPW2 scheme by using von-Neumann analysis to determine the time-stepping eigenvalue λ_{ts}. (Hint: If you cannot find the time-step restriction analytically, it may help to plot the resulting condition as a function of $\xi = k\Delta x$ to estimate the worst case).

(d) Write a Matlab code to solve this problem using the two upwind schemes, UPW1 and RK3-UPW2.

Take $c = 1$, and $t_{\text{final}} = \pi/2$ (note). Use the initial conditions

$$u_0(x) = \sin(k_x x),$$

with $k_x = 6$ (note the domain is $[0, 2\pi]$ and so k_x is an integer). Choose the time-step according to

$$\frac{c\Delta t}{\Delta x} = C_{\text{cfl}},$$

and adjust Δt in the usual way to exactly reach the final time.
For the UPW1 scheme choose (i) $C_{\text{cfl}} = 1$ and (ii) $C_{\text{cfl}} = .9$ (confirm that scheme is "exact" for $C_{\text{cfl}} = 1$.)

For the RK3-UPW2 scheme choose (iii) $C_{\text{cfl}} = 0.5$.

For each case (there are 3 cases), perform a grid refinement study as in previous homework. Solve the problem for $N_x = 10 \times 2^m$, $m = 1, 2, 3, 4, 5, 6$. Use the print statements:

```
fprintf('%s:%s: t=%6.3f: Nx=%3d Nt=%4d dt=%8.2e errMax=%8.2e errL1=%8.2e',...
         scheme,solution,t,Nx,Nt,dt,errMax(m),errL1(m));
if( m==1 ) fprintf('\n'); else fprintf(' ratio=[%3.2f,%3.2f] (max,L1)\n',...
                                        errMax(m-1)/errMax(m),errL1(m-1)/errL1(m)); end;
```

where `scheme='UPW1'` or `scheme='RK3UPW2'` and `solution='sine'`. Compute both the max-norm error and the discrete L1-norm error (and corresponding ratios). For cases (ii) and (iii), plot the computed solution and exact solution (on the same graph) at the final time for $m = 3, 4, 5, 6$ (four plots for each case).

Confirm that both schemes converge at the expected rate in both norms.

(e) Repeat part (d) but using the 2π-periodic *top-hat* initial condition, which, for $x \in [0, 2\pi)$, is defined by (choose $\alpha = 1$ and $\beta = 3$),

$$u_0(x) = \begin{cases} 1 & \text{if } \alpha < x < \beta, \\ \frac{1}{2} & \text{if } x = \alpha \text{ or } x = \beta, \\ 0 & \text{otherwise.} \end{cases}$$

Hint: if x is outside $[0, 2\pi)$ you can map x to its periodic image in $[0, 2\pi)$ using `xp=mod(x,2*pi)`. It may be easiest to define this function in a separate file `periodicTopHat.m`.

3. (20 pts) Consider the IBVP for the wave equation in *second-order form*,

$$u_{tt} - c^2 u_{xx} = f(x,t), \qquad a \le x \le b, \quad 0 \le t \le t_{\text{final}}, \tag{2}$$

$$u(a,t) = g_a(t), \quad u(b,t) = g_b(t), \qquad 0 \le t \le t_{\text{final}}, \tag{3}$$

$$u(x,0) = u_0(x), \quad u_t(x,0) = u_1(x), \quad a \le x \le b. \tag{4}$$

(a) Discretize this problem using the second-order accurate centered approximations in time and space,

$$u_{tt} \approx D_{+t} D_{-t} U_i^n, \qquad u_{xx} \approx D_+ D_- U_i^n$$

Carefully write down the scheme for (2)-(4), including boundary and initial conditions. Use the starting values for the first step U_i^1 derived in part (b).

(b) Starting value: The scheme is a three level scheme in time and requires special treatment to compute the solution at the first time step, U_i^1. Derive a starting value for the solution at the first step, $t = \Delta t$, by using Taylor series in time,

$$u(x, \Delta t) = u(x,0) + \Delta t u_t(x,0) + \frac{\Delta t^2}{2} u_{tt}(x,0) + \dots$$

and keeping the first three terms for a locally third-order accurate approximation. Use the initial conditions, and spatial derivatives of the initial conditions in this expansion.

(c) Write a Matlab code to solve this problem. You may want to animate the code by plotting the solution after every time-step (or every 2nd or 5th time-step). Use the command `drawnow` after plotting to force Matlab to refresh the plot.

Choose the forcing functions so that the following are exact solutions,

(i) Polynomial manufactured solution:

```
b0=.5; b1=.7; b2=.9;   % time coefficients
c0=1.; c1=.8; c2=.6;   % space coefficients
ue = @(x,t) (b0+b1*t+b2*t^2)*(c0+c1*x+c2*x.^2);
```

(ii) Traveling pulse:

$$u(x,t) = e^{-\beta^2(x-x_0-ct)^2},$$

where β and x_0 are arbitrary constants.

Choose $c = 0.5$ and $\beta = 20$, $a = 0$ and $b = 1$, and $x_0 = .25$, $t_{\text{final}} = 1.0$. Choose the time-step according to

$$\frac{c\Delta t}{\Delta x} = C_{\text{cfl}},$$

where $C_{\text{cfl}} = 0.9$, and adjust Δt in the usual way to exactly reach the final time.

For each case (there are 2 cases), perform a grid refinement study as in previous homework. Solve the problem for $N_x = 10 \times 2^m$, $m = 1, 2, 3, 4, 5$. Use the print statements:

```
fprintf('ms=%s: t=%8.2e: m=%d Nx=%3d Nt=%4d dt=%8.2e maxErr=%8.2e',ms,t,m,Nx,Nt,dt,errMax(m));
if( m==1 ) fprintf('\n'); else fprintf(' ratio=%5.2f rate=%5.2f\n',...
           errMax(m-1)/errMax(m),log2(errMax(m-1)/errMax(m))); end;
```

where `ms='poly'` or `ms='pulse'`. Compute the max-norm error (and corresponding ratios). Confirm that the scheme is "exact" for the polynomial manufactured solution.

For the case of the pulse solution, plot the computed solution and exact solution (on the same graph) and the error (on a separate graph) at the final time for $m = 1, 2, 3, 4$.

Q1

(a) Modified Advection Equation

$$u_t + cu_x = \alpha u \qquad -\infty < x < \infty, \quad t > 0 \qquad c, \alpha \in \mathbb{R}$$
$$u(x,0) = u_0(x) \qquad -\infty < x < \infty$$

Method of Characteristics,

Consider an arbitrary curve $x = I(t)$ and $U(t) = u(I(t), t)$.

Using Chain Rule,

$$\frac{dU(t)}{dt} = \frac{\partial u}{\partial t} + \frac{dI}{dt}\frac{du}{dx}$$

Then $\frac{dU(t)}{dt} = u_t + cu_x = \alpha u$ along the curve $\frac{dI}{dt} = c$

$$\Rightarrow \quad U(t) = U_0 e^{\alpha t} \quad \text{and} \quad X(t) = X_0 + ct$$

Using IC $u(x,0) = u_0(x)$,

$$U(0) = u_0(X_0) = U_0(X(t) - ct) = U_0(x - ct)$$

$$\therefore \quad \boxed{u(x,t) = u_0(x-ct)\, e^{\alpha t}}$$

(b) Using TS in time

(1) $$u(x, t+\Delta t) = u(x,t) + \Delta t\, u_t(x,t) + \frac{\Delta t^2}{2} u_{tt}(x,t) + O(\Delta t^3)$$

and $\partial_t \to \partial_x$ with

(2) $$u_t = \alpha u - cu_x$$

(3) $$u_{tt} = (\alpha u - cu_x)_t = \alpha u_t - c(u_t)_x = \alpha(\alpha u - cu_x) - c(\alpha u - cu_x)_x$$
$$= \alpha^2 u - 2\alpha c u_x + c^2 u_{xx}$$

Plug (2), (3) in (1)

$$\Rightarrow u(x, t+\Delta t) = u(x,t) + \Delta t(\alpha u - cu_x) + \frac{\Delta t^2}{2}(\alpha^2 u - 2\alpha c u_x + c^2 u_{xx}) + O(\Delta t^3)$$

collecting similar terms,

$$\Rightarrow u(x, t+\Delta t) = \left[1 + \alpha\Delta t + \alpha^2\frac{\Delta t^2}{2}\right]u(x,t) + \left[-c\Delta t - \alpha c \Delta t^2\right]u_x(x,t) + \frac{(c\Delta t)^2}{2}u_{xx}(x,t)$$

$$\Rightarrow \text{LW}_{\text{2nd order accurate}} \longrightarrow \boxed{U_i^{n+1} = \left[1 + \alpha\Delta t + \alpha^2\frac{\Delta t^2}{2}\right]U_i^n - c\Delta t\left[1 + \alpha\Delta t\right]D_{0x}U_i^n + \frac{(c\Delta t)^2}{2}D_{+x}D_{-x}U_i^n}$$

(c) Substituting a solution $v(x,t)$ to the modified equation into the LW scheme

$$\underbrace{\frac{v(x,t+\Delta t)-v(x,t)}{\Delta t}}_{①} + c\underbrace{\left[\frac{v(x+\Delta x,t)-v(x-\Delta x,t)}{2\Delta x}\right]}_{②} = \frac{c^2\Delta t}{2}\underbrace{\left[\frac{v(x+\Delta x,t)-2v(x,t)+v(x-\Delta x,t)}{2\Delta x}\right]}_{③}$$

Using TS,

$$① = v_t(x,t) + \frac{\Delta t}{2}v_{tt}(x,t) + \frac{\Delta t^2}{6}v_{ttt}(x,t) + O(\Delta t^3)$$

$$② = c\left[v_x(x,t) + \frac{\Delta x^2}{6}v_{xxx} + O(\Delta x^4)\right]$$

$$③ \quad \frac{c^2\Delta t}{2}\left[v_{xx} + O(\Delta x^2)\right]$$

$$\Rightarrow \quad v_t(x,t) + cv_x(x,t) = -\frac{\Delta t}{2}v_{tt} - \frac{\Delta t^2}{6}v_{ttt} - c\frac{\Delta x^2}{6}v_{xxx} + \frac{c^2\Delta t}{2}v_{xx} + O(\Delta t^3 + \Delta x^4 + \Delta t\Delta x^2)$$

$$\Rightarrow \quad v_t(x,t) + cv_x(x,t) = -\frac{\Delta t}{2}v_{tt} + \frac{c^2\Delta t}{2}v_{xx} + O(\Delta t^2 + \Delta x^2)$$

$$\Rightarrow \quad v_t(x,t) + cv_x(x,t) =$$

Q2 Advection Equation

$$u_t + c u_x = 0, \quad 0 \le x \le 2\pi, \quad 0 \le t \le t_{final}$$
$$u(x,0) = u_0(x)$$
$$u(x+2\pi, t) = u(x,t)$$

(a) $$D_-^{(2)} U_i = \frac{3U_i - 4U_{i-1} + U_{i-2}}{2\Delta x}$$

$$D_-^{(2)} e^{ikx_j} = \frac{3e^{ikx_j} - 4e^{ikx_{j-1}} + e^{ikx_{j-2}}}{2\Delta x}$$

$$= \frac{3e^{ikx_j} - 4e^{ikx_j} e^{-ik\Delta x} + e^{-2ik\Delta x}}{2\Delta x}$$

$$= \underbrace{\left[\frac{3 - 4e^{-ik\Delta x} + e^{-2ik\Delta x}}{2\Delta x}\right]}_{\widehat{D_-^{(2)}} \text{ symbol}} e^{ikx_j}$$

For small $k\Delta x$,

$$\widehat{D_-^{(2)}} \approx \frac{3 - 4\left[1 + (-ik\Delta x) + \frac{(-ik\Delta x)^2}{2!} + O(k\Delta x^3)\right] + \left[1 + (-2ik\Delta x) + \frac{(-2ik\Delta x)^2}{2!} + O((k\Delta x)^3)\right]}{2\Delta x}$$

$$= \frac{(3-4+1) + (4ik\Delta x - 2ik\Delta x) + (-2k^2\Delta x^2 + 2k^2\Delta x^2) + O(\Delta x^3)}{2\Delta x}$$

$$= ik + O(\Delta x^3)$$

(b) Test equation $y' = \lambda y$

RK3 scheme

(0) $Y^{n+1} = Y^n + \frac{\Delta t}{6}(k_1 + k_2 + 4k_3)$

(1) $k_1 = \lambda Y^n$

(2) $k_2 = f(Y^n + \Delta t k_1, t^n + \Delta t)$
$= f(Y^n + \Delta t \lambda Y^n, t^n + \Delta t)$
$= f((1 + \lambda\Delta t)Y^n, t^n + \Delta t)$
$= \lambda(1 + \lambda\Delta t)Y^n$

(3) $k_3 = f(Y^n + \frac{\Delta t}{4}k_1 + \frac{\Delta t}{4}k_2, t^n + \frac{\Delta t}{2})$
$= f(Y^n + \frac{\lambda\Delta t Y^n}{4} + \frac{(\lambda\Delta t)(1+\lambda\Delta t)Y^n}{4}, t^n + \frac{\Delta t}{2})$
$= f((1 + \frac{\lambda\Delta t}{4} + \frac{(\lambda\Delta t)(1+\lambda\Delta t)}{4})Y^n, t^n + \frac{\Delta t}{2})$
$= f((1 + \frac{\lambda\Delta t}{2} + \frac{\lambda^2\Delta t^2}{4})Y^n, t^n + \frac{\Delta t}{2})$
$= \lambda(1 + \frac{\lambda\Delta t}{2} + \frac{\lambda^2\Delta t^2}{4})Y^n$

$\Rightarrow$ Sub (3), (2), (1) in (0)

$Y^{n+1} = Y^n + \frac{\Delta t}{6}\lambda Y^n + \frac{\Delta t}{6}\lambda(1+\lambda\Delta t)Y^n + \frac{4\Delta t}{6}\lambda(1 + \frac{\lambda\Delta t}{2} + \frac{\lambda^2\Delta t^2}{4})Y^n$

$= \underbrace{\left[1 + (\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6=3!}(\lambda\Delta t)^3\right]}_{A} Y^n$

$A = 1 + (\lambda\Delta t) + \frac{1}{2!}(\lambda\Delta t)^2 + \frac{1}{3!}(\lambda\Delta t)^3 + O((\lambda\Delta t)^4)$

and A is a locally fourth-order accurate approximation in $\lambda\Delta t$ to $e^{\lambda\Delta t}$.

(c) Using the Von Neumann analysis, we can write the time-stepping eigenvalue

$$\lambda_{ts} = -c\,\hat{D}_-^{(2)} = -c\left[\frac{3 - 4e^{-ik\Delta x} + e^{-2ik\Delta x}}{2\Delta x}\right]$$

We can find the time-step restriction from

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\alpha^2} \leq 1 \quad \text{where} \quad z = x + iy = \lambda_{ts}\Delta t$$

$$|x + iy| = |\lambda_{ts}\Delta t|$$

$$x^2 + y^2 = |\lambda_{ts}\Delta t|^2 = \left(\frac{-c\Delta t}{2\Delta x}\right)^2 \underbrace{\left|3 - 4e^{-ik\Delta x} + e^{-2ik\Delta x}\right|^2}_{(2)}$$

(1)

(2) $\left|3 - 4e^{-ik\Delta x} + e^{-2ik\Delta x}\right|^2 \equiv C$ and $\xi \equiv k\Delta x$

$$C\bar{C} = \left[3 - 4e^{-i\xi} + e^{-2i\xi}\right]\left[3 - 4e^{i\xi} + e^{2i\xi}\right]$$

$$= 9 - 12e^{i\xi} + 3e^{2i\xi} - 12e^{-i\xi} + 16e^{0} - 4e^{i\xi} + 3e^{-2i\xi} - 4e^{-i\xi} + e^{0}$$

$$= 26 - 16\left[e^{i\xi} + e^{-i\xi}\right] + 3\left[e^{2i\xi} + e^{-2i\xi}\right]$$

using
$e^{i\theta} = \cos\theta + i\sin\theta$
$e^{-i\theta} = \cos\theta - i\sin\theta$

$$\Rightarrow 26 - 32\cos(\xi) + 6\cos(2\xi) = C\bar{C}$$

$$\frac{d(C\bar{C})}{d\xi} = 4(8\sin(\xi) - 3\sin(2\xi)) = 0$$

$\Rightarrow \pi = 0$ at max point

$\Rightarrow$ Max value of $C\bar{C} = 64$

Now, (1)(2) $\left(\frac{-c\Delta t}{2\Delta x}\right)^2 (64) \leq 2.2^2 \Rightarrow \frac{|c|\Delta t}{2\Delta x}(8) \leq 2.2$

$$\Rightarrow \boxed{\frac{|c|\Delta t}{\Delta x} \leq 0.55}$$

MATH-6840
Samiullah Malik
April 2, 2020

Numerical Solutions of PDEs HW - 4

Question 2

Listing 1: Matlab Code

```
1
2  %% Question 2
3
4  % 1. Input Data
5  a = 0.; b = 2*pi; % Domain [a,b]
6  tfinal = pi/2.;
7  c = 1.;       % Wave speed
8  kx = 6.;
9  alpha = 1.; beta = 3.; % top_hat parameters
10
11 ms = 'rk3-upw2';
12 solution = 'top_hat';
13 %ms = 'upw1-case2';
14
15 % 2. Set cfl for different schemes.
16 if( strcmp(ms, 'upw1-case1') )
17     cfl = 1.;
18 elseif ( strcmp(ms, 'upw1-case2') )
19     cfl = .9;
20 elseif ( strcmp(ms, 'rk3-upw2') )
21     cfl = .5;
22 end
23
24 % 3. Define functions
25 if ( strcmp(solution, 'sine') )
26     uexact = @(x,t) sin(kx*(x-c*t));
27 elseif ( strcmp(solution, 'top_hat') )
28     uexact = @(x,t) periodicTopHat(x-c*t, alpha, beta);
29 end
30 u0 = @(x) uexact(x,0); % Given initial condition function
31
32
33
34 % 4(a). Grid Resolutions (1,2,3,4,5,6)
35 numResolutions=6;
36 for m=1:numResolutions
37
38     % (b). Setup Grid
39     Nx=10*2^m;          % no. of grid intervals
40     dx=(b-a)/Nx;        % size of each interval
41     lgp = 2;            % no. of left ghost points
42     ia=lgp+1;           % index of LHS boundary point at x=a
43     ib=ia+Nx;           % index of RHS boundary point at x=b
44     i1=ia;              % first interior index (due to left ghost points)
45     i2=ib-1;            % last interior index
46     I = i1:i2;          % interior points (PDE is applied here)
47     x = linspace(a-(lgp*dx),b,Nx+ia)'; % grid (coordinate values)
48
49     % (c). Allocating space for the solution. (3-levels).
```

```
    un = zeros(Nx+ia,1);
    unp1 = zeros(Nx+ia,1);
    if ( strcmp(ms,'rk3-upw2') )
        k1 = zeros(Nx+ia,1);
        k2 = zeros(Nx+ia,1);
        k3 = zeros(Nx+ia,1);
        temp = zeros(Nx+ia,1);
    end

    % (d). Adjust time step
    dt = cfl*dx/c;
    Nt = round(tfinal/dt);   % no. of time-steps
    dt = tfinal/Nt; %        adjust dt to reach tfinal exactly

    % (e). Define difference operators
    Dp = @(U,I) ( U(I+1) - U(I) )/ dx;
    Dm = @(U,I) ( U(I) - U(I-1) )/ dx;
    Dm2 = @(U,I) ( 3*U(I) - 4*U(I-1) + U(I-2) )/(2*dx); %upwind scheme

    % (f). Apply Initial Conditions
    un = u0(x);            % at t=0

    % (g). Apply time-stepping loop
    for n = 1:Nt
        t_current = dt*(n-1);
        t_new = dt*n;

        if ( strcmp(ms, 'upw1-case1') || strcmp(ms, 'upw1-case2') )
            unp1(I) = un(I) - (c*dt/dx)*( un(I)-un(I-1));
            unp1 = applyBCs(lgp,ia,Nx, unp1);
        elseif ( strcmp(ms, 'rk3-upw2') )
            k1(I) = -c*Dm2(un,I);           % Stage 1
            temp(I) = un(I) + dt*k1(I);
            temp = applyBCs(lgp,ia,Nx,temp);

            k2(I) = -c*Dm2(temp,I);         % Stage 2
            temp(I) = un(I) + (dt/4.)*(k1(I) + k2(I));
            temp = applyBCs(lgp, ia, Nx, temp);
            k3(I) = -c*Dm2(temp,I);


            unp1(I) = un(I) + (dt/6.)*(k1(I) + k2(I) + 4*k3(I));
            unp1 = applyBCs(lgp, ia, Nx, unp1);
        end
        un = unp1;
    end % time-stepping loop

    % (h). Compute errors and print
    u_exact_solution = uexact(x,tfinal);
    error = un - u_exact_solution;
    maxNormError(m) = max(abs(error(ia:ib)));
    l1NormError(m) = norm(error,1)*dx;

    fprintf('ms=%s: solution=%s: t=%6.3f: Nx=%3d Nt=%4d dt=%8.2e errMax=%8.2e errL1=%8.2e',...
           ms, solution, tfinal,Nx,Nt,dt,maxNormError(m),l1NormError(m));
    if( m==1 ) fprintf('\n'); else fprintf(' ratio=[%3.2f,%3.2f] (max,L1)\n',...
              maxNormError(m-1)/maxNormError(m),l1NormError(m-1)/l1NormError(m)); end

    % (i). Plot computed solution vs exact solution
```

```
    if m==3
        subplot(2,2,1)
        plot( x(ia:ib) ,un(ia:ib), 'r-', x(ia:ib), u_exact_solution(ia:ib), 'b-')
        lgd = legend('computed', 'exact')
        lgd.FontSize = 6;
        title('RK3 - TopHat - Computed vs Exact Solution - m=3')
        xlabel('x');
        ylabel('u');

    elseif m == 4
        subplot(2,2,2)
        plot( x(ia:ib) ,un(ia:ib), 'r-', x(ia:ib), u_exact_solution(ia:ib), 'b-')
        lgd = legend('computed', 'exact')
        lgd.FontSize = 6;
        title('RK3 - TopHat - Computed vs Exact Solution - m=4')
        xlabel('x');
        ylabel('u');


    elseif m == 5
        subplot(2,2,3)
        plot( x(ia:ib) ,un(ia:ib), 'r-', x(ia:ib), u_exact_solution(ia:ib), 'b-')
        lgd = legend('computed', 'exact')
        lgd.FontSize = 6;
        title('RK3 - TopHat - Computed vs Exact Solution - m=5')
        xlabel('x');
        ylabel('u');


    elseif m == 6
        subplot(2,2,4)
        plot( x(ia:ib) ,un(ia:ib), 'r-', x(ia:ib), u_exact_solution(ia:ib), 'b-')
        lgd = legend('computed', 'exact')
        lgd.FontSize = 6;
        title('RK3 - TopHat - Computed vs Exact Solution - m=6')
        xlabel('x');
        ylabel('u');


    end

end % loop grid resolutions









% Apply the Boundary Conditions
function [U] = applyBCs(lgp,ia,Nx,U) % solution: U, left bdry index: ia
    U(ia+Nx) = U(ia); % RHS bdry point
    for n=1:lgp       % left ghost points
        U(ia-n) = U(ia-n+Nx);
    end
end
```

Output Results

```
>> hw4_q2
ms=upw1-case1: solution=sine: t= 1.571: Nx= 20 Nt=   5 dt=3.14e-01 errMax=5.88e-15 errL1=6.42e-15
ms=upw1-case1: solution=sine: t= 1.571: Nx= 40 Nt=  10 dt=1.57e-01 errMax=3.33e-15 errL1=3.85e-15 ratio=[1.77,1.67] (max,L1)
ms=upw1-case1: solution=sine: t= 1.571: Nx= 80 Nt=  20 dt=7.85e-02 errMax=5.88e-15 errL1=7.21e-15 ratio=[0.57,0.53] (max,L1)
ms=upw1-case1: solution=sine: t= 1.571: Nx=160 Nt=  40 dt=3.93e-02 errMax=7.08e-15 errL1=8.88e-15 ratio=[0.83,0.81] (max,L1)
ms=upw1-case1: solution=sine: t= 1.571: Nx=320 Nt=  80 dt=1.96e-02 errMax=8.66e-15 errL1=7.02e-15 ratio=[0.82,1.26] (max,L1)
ms=upw1-case1: solution=sine: t= 1.571: Nx=640 Nt= 160 dt=9.82e-03 errMax=7.77e-15 errL1=6.56e-15 ratio=[1.11,1.07] (max,L1)
>> hw4_q2
ms=upw1-case2: solution=sine: t= 1.571: Nx= 20 Nt=   6 dt=2.62e-01 errMax=8.12e-01 errL1=3.71e+00
ms=upw1-case2: solution=sine: t= 1.571: Nx= 40 Nt=  11 dt=1.43e-01 errMax=3.31e-01 errL1=1.43e+00 ratio=[2.45,2.59] (max,L1)
ms=upw1-case2: solution=sine: t= 1.571: Nx= 80 Nt=  22 dt=7.14e-02 errMax=1.83e-01 errL1=7.48e-01 ratio=[1.81,1.91] (max,L1)
ms=upw1-case2: solution=sine: t= 1.571: Nx=160 Nt=  44 dt=3.57e-02 errMax=9.60e-02 errL1=3.86e-01 ratio=[1.90,1.94] (max,L1)
ms=upw1-case2: solution=sine: t= 1.571: Nx=320 Nt=  89 dt=1.76e-02 errMax=5.46e-02 errL1=2.19e-01 ratio=[1.76,1.77] (max,L1)
ms=upw1-case2: solution=sine: t= 1.571: Nx=640 Nt= 178 dt=8.82e-03 errMax=2.77e-02 errL1=1.11e-01 ratio=[1.97,1.97] (max,L1)
>> hw4_q2
ms=rk3-upw2: solution=sine: t= 1.571: Nx= 20 Nt=  10 dt=1.57e-01 errMax=9.51e-01 errL1=4.35e+00
ms=rk3-upw2: solution=sine: t= 1.571: Nx= 40 Nt=  20 dt=7.85e-02 errMax=1.07e+00 errL1=4.64e+00 ratio=[0.89,0.94] (max,L1)
ms=rk3-upw2: solution=sine: t= 1.571: Nx= 80 Nt=  40 dt=3.93e-02 errMax=6.01e-01 errL1=2.47e+00 ratio=[1.77,1.88] (max,L1)
ms=rk3-upw2: solution=sine: t= 1.571: Nx=160 Nt=  80 dt=1.96e-02 errMax=1.71e-01 errL1=7.02e-01 ratio=[3.52,3.52] (max,L1)
ms=rk3-upw2: solution=sine: t= 1.571: Nx=320 Nt= 160 dt=9.82e-03 errMax=4.35e-02 errL1=1.76e-01 ratio=[3.93,3.98] (max,L1)
ms=rk3-upw2: solution=sine: t= 1.571: Nx=640 Nt= 320 dt=4.91e-03 errMax=1.09e-02 errL1=4.39e-02 ratio=[3.99,4.02] (max,L1)
```

Figure 1: Output

```
>> hw4_q2
ms=upw1-case1: solution=top_hat: t= 1.571: Nx= 20 Nt=   5 dt=3.14e-01 errMax=0.00e+00 errL1=0.00e+00
ms=upw1-case1: solution=top_hat: t= 1.571: Nx= 40 Nt=  10 dt=1.57e-01 errMax=0.00e+00 errL1=0.00e+00 ratio=[NaN,NaN] (max,L1)
ms=upw1-case1: solution=top_hat: t= 1.571: Nx= 80 Nt=  20 dt=7.85e-02 errMax=0.00e+00 errL1=0.00e+00 ratio=[NaN,NaN] (max,L1)
ms=upw1-case1: solution=top_hat: t= 1.571: Nx=160 Nt=  40 dt=3.93e-02 errMax=0.00e+00 errL1=0.00e+00 ratio=[NaN,NaN] (max,L1)
ms=upw1-case1: solution=top_hat: t= 1.571: Nx=320 Nt=  80 dt=1.96e-02 errMax=0.00e+00 errL1=0.00e+00 ratio=[NaN,NaN] (max,L1)
ms=upw1-case1: solution=top_hat: t= 1.571: Nx=640 Nt= 160 dt=9.82e-03 errMax=0.00e+00 errL1=0.00e+00 ratio=[NaN,NaN] (max,L1)
>> hw4_q2
ms=upw1-case2: solution=top_hat: t= 1.571: Nx= 20 Nt=   6 dt=2.62e-01 errMax=3.35e-01 errL1=4.21e-01
ms=upw1-case2: solution=top_hat: t= 1.571: Nx= 40 Nt=  11 dt=1.43e-01 errMax=3.50e-01 errL1=2.20e-01 ratio=[0.96,1.91] (max,L1)
ms=upw1-case2: solution=top_hat: t= 1.571: Nx= 80 Nt=  22 dt=7.14e-02 errMax=3.93e-01 errL1=1.62e-01 ratio=[0.89,1.36] (max,L1)
ms=upw1-case2: solution=top_hat: t= 1.571: Nx=160 Nt=  44 dt=3.57e-02 errMax=4.24e-01 errL1=1.17e-01 ratio=[0.93,1.39] (max,L1)
ms=upw1-case2: solution=top_hat: t= 1.571: Nx=320 Nt=  89 dt=1.76e-02 errMax=4.49e-01 errL1=8.83e-02 ratio=[0.95,1.33] (max,L1)
ms=upw1-case2: solution=top_hat: t= 1.571: Nx=640 Nt= 178 dt=8.82e-03 errMax=4.64e-01 errL1=6.27e-02 ratio=[0.97,1.41] (max,L1)
>> hw4_q2
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx= 20 Nt=  10 dt=1.57e-01 errMax=4.75e-01 errL1=8.42e-01
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx= 40 Nt=  20 dt=7.85e-02 errMax=5.25e-01 errL1=6.24e-01 ratio=[0.90,1.35] (max,L1)
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx= 80 Nt=  40 dt=3.93e-02 errMax=5.53e-01 errL1=4.06e-01 ratio=[0.95,1.53] (max,L1)
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx=160 Nt=  80 dt=1.96e-02 errMax=5.79e-01 errL1=2.71e-01 ratio=[0.96,1.50] (max,L1)
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx=320 Nt= 160 dt=9.82e-03 errMax=5.98e-01 errL1=1.80e-01 ratio=[0.97,1.50] (max,L1)
ms=rk3-upw2: solution=top_hat: t= 1.571: Nx=640 Nt= 320 dt=4.91e-03 errMax=6.13e-01 errL1=1.19e-01 ratio=[0.98,1.51] (max,L1)
```

Figure 2: Output

Graphs

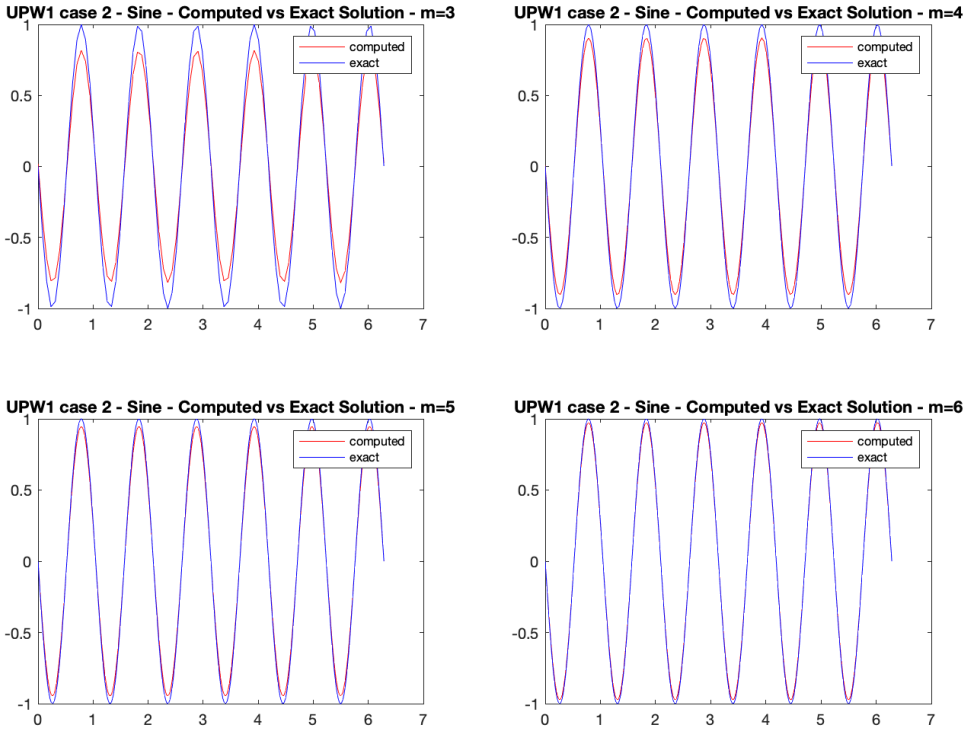

Figure 3: Output

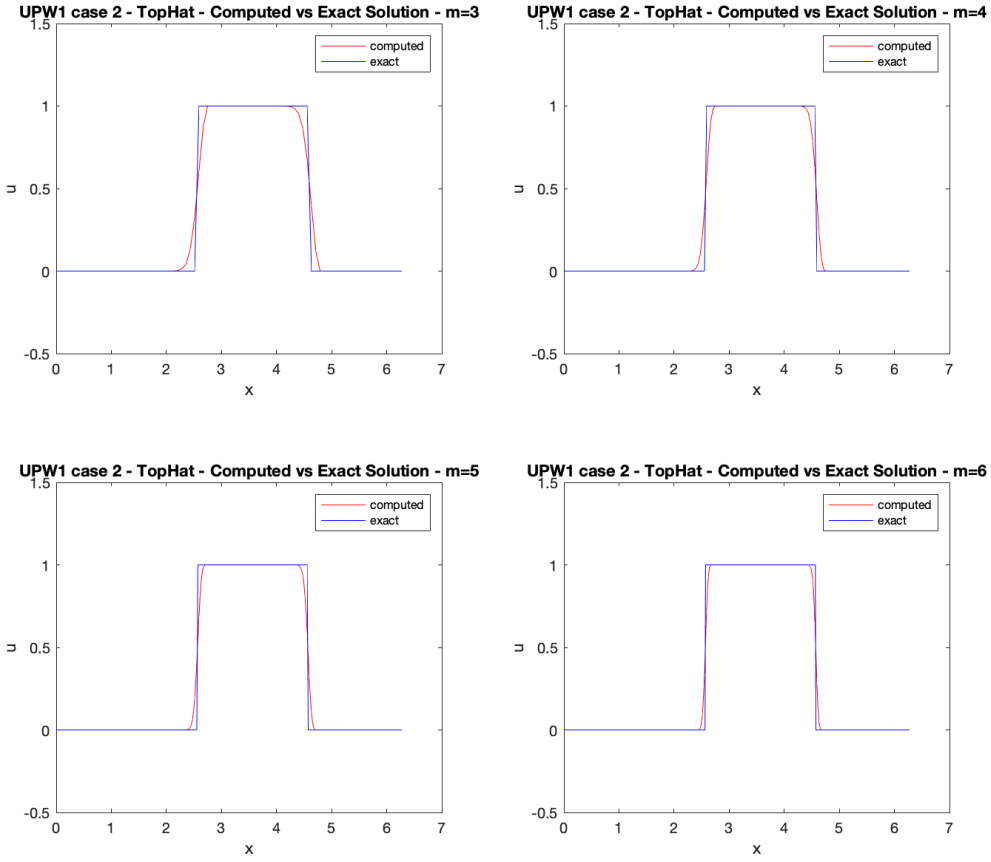

Figure 4: Output

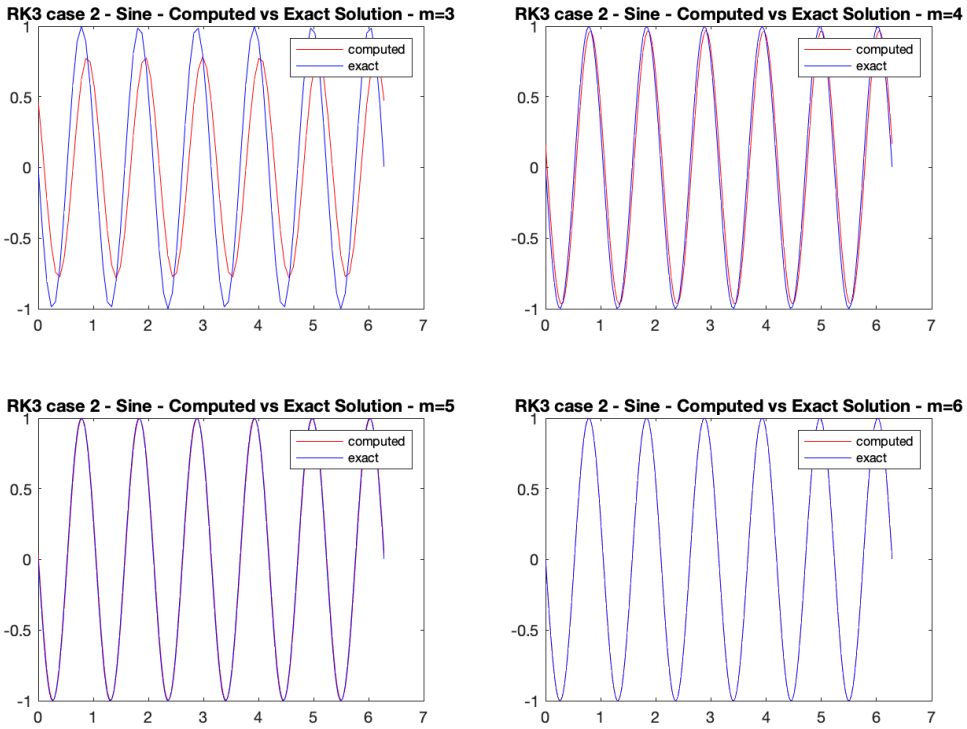

Figure 5: Output

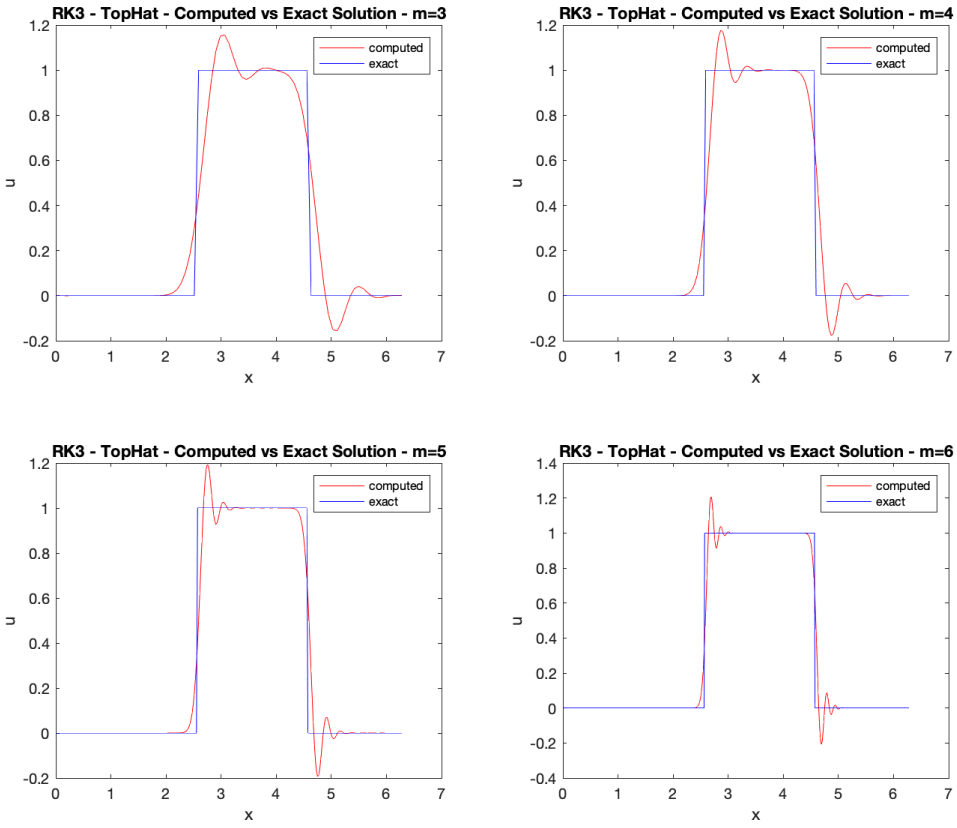

Figure 6: Output

Q3

(a) $D_{+t}D_{-t}U_i^n - c^2 D_{+x}D_{-x}U_i^n = f(x_i, t^n)$ $\quad i = 1, 2, \ldots, N_x - 1$

BCs $\quad U_0^n = g_a(t)$, $U_{N_x}^n = g_b(t)$ $\quad n = 1, 2, \ldots$

ICs $\quad U_i^0 = u_0(x)$ $\quad i = 0, 1, 2, \ldots, N_x$

$U_i^1 =$ given below $\quad i = 0, 1, 2, \ldots, N_x$

(b) Given

$$u(x, \Delta t) = u(x,0) + \Delta t\, u_t(x,0) + \frac{\Delta t^2}{2} u_{tt}(x,0) + \ldots$$

we know that $u_t(x,0) = u_1(x)$, so

$$u(x, \Delta t) = u_0(x_i, 0) + \Delta t\, u_1(x_i, 0) + \frac{\Delta t^2}{2} u_{tt}(x, 0)$$

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

$$u_{tt}(x,0) = c^2 u_{xx} + f(x_i, 0)$$

↓ using spatial derivatives of the initial condition

$$c^2 \frac{\partial^2 u(x_i, 0)}{\partial x^2}$$

$$\Rightarrow \quad U_i^1 = u_0(x_i, 0) + \Delta t\, u_1(x_i, 0) + \frac{\Delta t^2}{2}\left[c^2 \frac{\partial^2 u_0(x_i, 0)}{\partial x^2} + f(x_i, 0) \right]$$

MATH-6840
Samiullah Malik
April 2, 2020

Numerical Solutions of PDEs HW - 4

Question 3

Listing 1: Matlab Code

```
%% Question # 3


% 1. Input Data
a = 0.; b = 1; % Domain [a,b]
tfinal = 1.;
c = 0.5;       % Wave speed
cfl = 0.9;


ms = 'pulse';

% 2. Define Manufactured Polynomial and Traveling pulse.
% For manufactured polynomial solution
if( strcmp(ms, 'poly') )
    b0=.5; b1=.7; b2=.9; % time coefficients
    c0=1.; c1=.8; c2=.6; % space coefficients

    ue = @(x,t) (b0+b1*t+b2*t^2)*(c0+c1*x+c2*x.^2);
    uet = @(x,t) (b1+2.*b2*t)*(c0+c1*x+c2*x.^2);
    uett = @(x,t) (2.*b2)*(c0+c1*x+c2*x.^2);
    uex = @(x,t) (b0+b1*t+b2*t^2)*(c1+2.*c2*x);
    uexx = @(x,t) (b0+b1*t+b2*t^2)*(2.*c2);
    f = @(x,t) uett(x,t) - (c^2)*uexx(x,t);
end

% For traveling pulse solution
if( strcmp(ms, 'pulse') )
    beta = 20; x0 = 0.25;

    ue = @(x,t) exp( -(beta*(x - x0 - c*t) ).^2 );
    uet = @(x,t) (2.*c*beta^2 * (x - x0 - c*t)) .*ue(x,t);
    %uett = @(x,t)
    uex = @(x,t) -2.*(beta^2)*(x-x0-ct).*ue(x,t);
    uexx = @(x,t) -2.*beta^2 * (-2.*ue(x,t) * beta^2 * (x-x0-c*t)^2 ...
                   + ue(x,t));
    uexx = @(x,t) ( -(2.*beta^2) + (-2.*beta^2*(x-x0-c*t)).^2 ).* ue(x,t);
    f = @(x,t) 0; %% used symbolab.com for symbolic computation
end

% 3. Define BCs and ICs
ga = @(t) ue(a,t);    % BC at x = a (LHS)
gb = @(t) ue(b,t);    % BC at x = b (RHS)
u0 = @(x) ue(x,0);    % IC for u0(x)
u1 = @(x) uet(x,0);   % IC for u1(x)


% 4(a). Grid Resolutions (1,2,3,4,5)
```

```
numResolutions=5;
for m=1:numResolutions

    % (b). Setup Grid
    Nx=10*2^m;          % no. of grid intervals
    dx=(b-a)/Nx;        % size of each interval
    ia=1;               % index of LHS boundary point at x=a
    ib=ia+Nx;           % index of RHS boundary point at x=b
    i1=ia+1;            % first interior index
    i2=ib-1;            % last interior index
    I = i1:i2;          % interior points (PDE is applied here)
    x = linspace(a,b,Nx+1)'; % grid points

    % (c). Allocating space for the solution. (3-levels).
    unm1 = zeros(Nx+1,1);
    un = zeros(Nx+1,1);
    unp1 = zeros(Nx+1,1);

    % (d). Adjust time step
    dt = cfl*dx/c;
    Nt = round(tfinal/dt);    % no. of time-steps
    dt = tfinal/Nt; %         adjust dt to reach tfinal exactly

    % (e). Apply Initial Conditions
    unm1 = u0(x);             % at t=0
    un = unm1 + dt*u1(x) + ((dt^2)/2.) * ((c^2) * uexx(x,0) + f(x,0)); % at t=dt
    t=dt;

    % (f). Apply time-stepping loop
    for n = 2:Nt
        t_current = dt*(n-1);
        t_new = dt*n;

        unp1(I) = 2.*un(I) - unm1(I) + (c*dt/dx)^2*( un(I+1)-2*un(I)+un(I-1) ) + (dt^2)*f(x(I),
            t_current);
        unp1(ia) = ga(t_new); % apply BCs on LHS
        unp1(ib) = gb(t_new); % apply BCs on RHS

        unm1 = un; % set unm1 un and un for next time step
        un = unp1;

    end

    % (g). Compute errors and print
    u_exact_solution = ue(x,tfinal);
    error = un - u_exact_solution;
    errMax(m) = max(abs(error(ia:ib)));
    fprintf('ms=%s: t=%8.2e: m=%d Nx=%3d Nt=%4d dt=%8.2e maxErr=%8.2e',ms,t,m,Nx,Nt,dt,errMax(m));
    if( m==1 ) fprintf('\n'); else fprintf(' ratio=%5.2f rate=%5.2f\n', ...
        errMax(m-1)/errMax(m),log2(errMax(m-1)/errMax(m))); end

    % (h). Plot for pulse
    if( strcmp(ms,'pulse') )
        subplot(2,1,1)
        plot( x(ia:ib) ,un, 'r-', x, u_exact_solution, 'b-')
        legend('computed', 'exact')
        title('Wave Equation Pulse - Computed vs Exact Solution - m=1')

        subplot(2,1,2)
```

```
        plot( x(ia:ib) ,error, 'r-')
        legend('error')
        title('Wave Equation Pulse - Error - m=1')
    end

end
```

Output Results

```
>> hw4_q3
ms=poly: t=9.09e-02: m=1 Nx= 20 Nt=  11 dt=9.09e-02 maxErr=2.66e-15
ms=poly: t=4.55e-02: m=2 Nx= 40 Nt=  22 dt=4.55e-02 maxErr=4.44e-15 ratio= 0.60 rate=-0.74
ms=poly: t=2.27e-02: m=3 Nx= 80 Nt=  44 dt=2.27e-02 maxErr=6.22e-15 ratio= 0.71 rate=-0.49
ms=poly: t=1.12e-02: m=4 Nx=160 Nt=  89 dt=1.12e-02 maxErr=1.87e-14 ratio= 0.33 rate=-1.58
ms=poly: t=5.62e-03: m=5 Nx=320 Nt= 178 dt=5.62e-03 maxErr=2.49e-14 ratio= 0.75 rate=-0.42
>> hw4_q3
ms=pulse: t=9.09e-02: m=1 Nx= 20 Nt=  11 dt=9.09e-02 maxErr=3.18e-01
ms=pulse: t=4.55e-02: m=2 Nx= 40 Nt=  22 dt=4.55e-02 maxErr=9.90e-02 ratio= 3.21 rate= 1.68
ms=pulse: t=2.27e-02: m=3 Nx= 80 Nt=  44 dt=2.27e-02 maxErr=2.17e-02 ratio= 4.57 rate= 2.19
ms=pulse: t=1.12e-02: m=4 Nx=160 Nt=  89 dt=1.12e-02 maxErr=5.77e-03 ratio= 3.75 rate= 1.91
ms=pulse: t=5.62e-03: m=5 Nx=320 Nt= 178 dt=5.62e-03 maxErr=1.46e-03 ratio= 3.97 rate= 1.99
```

Figure 1: Output

Graphs

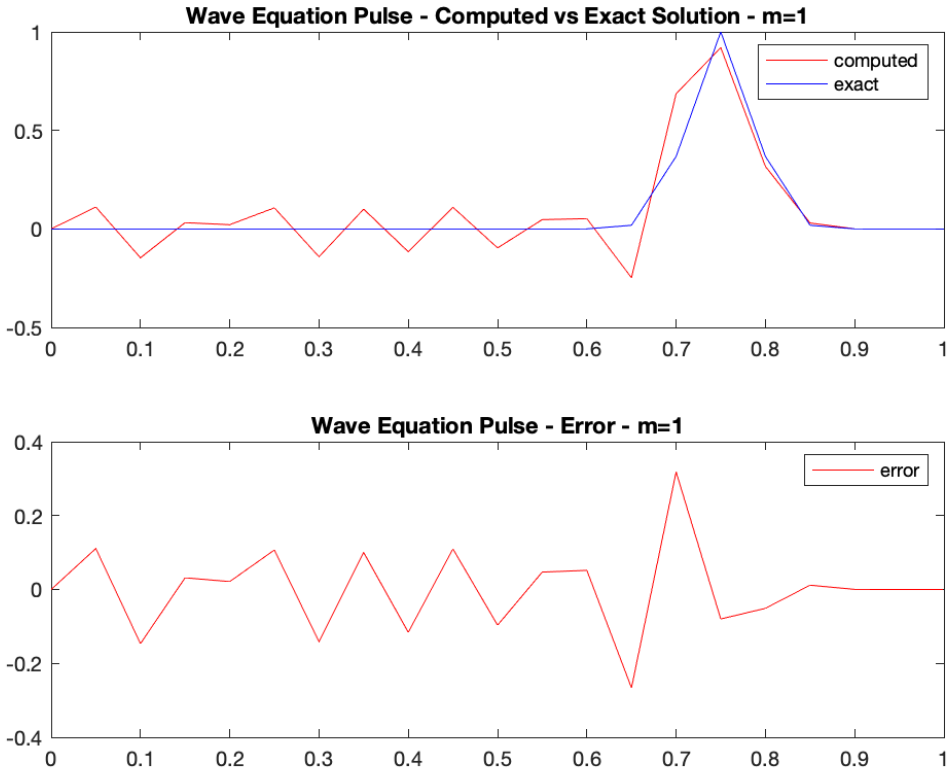

Figure 2: Output

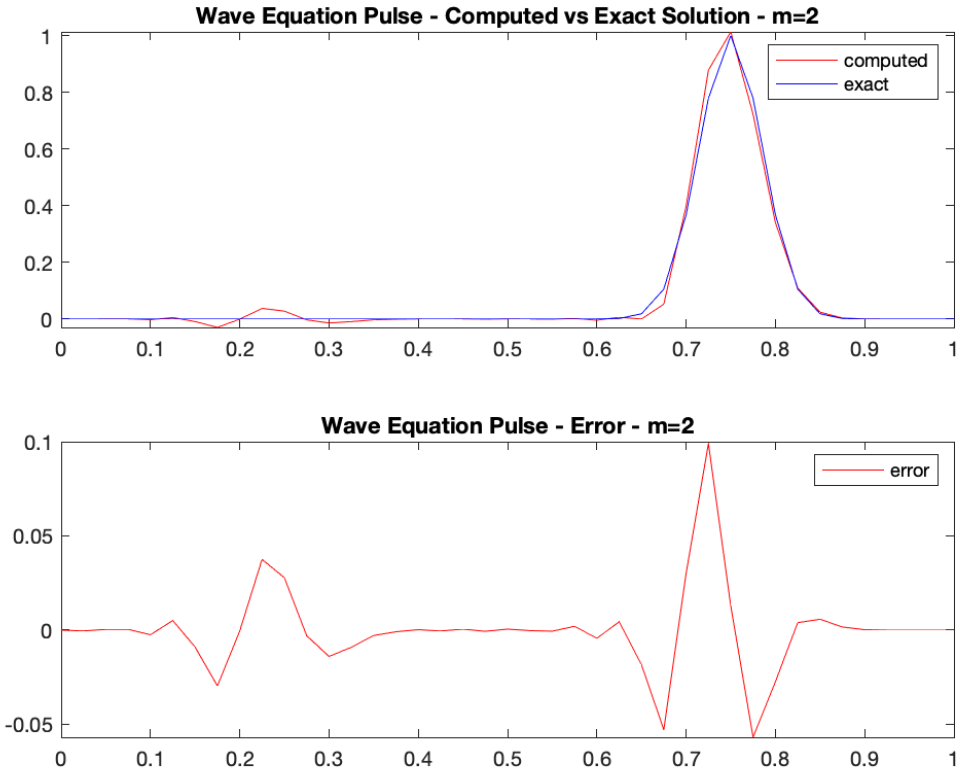

Figure 3: Output

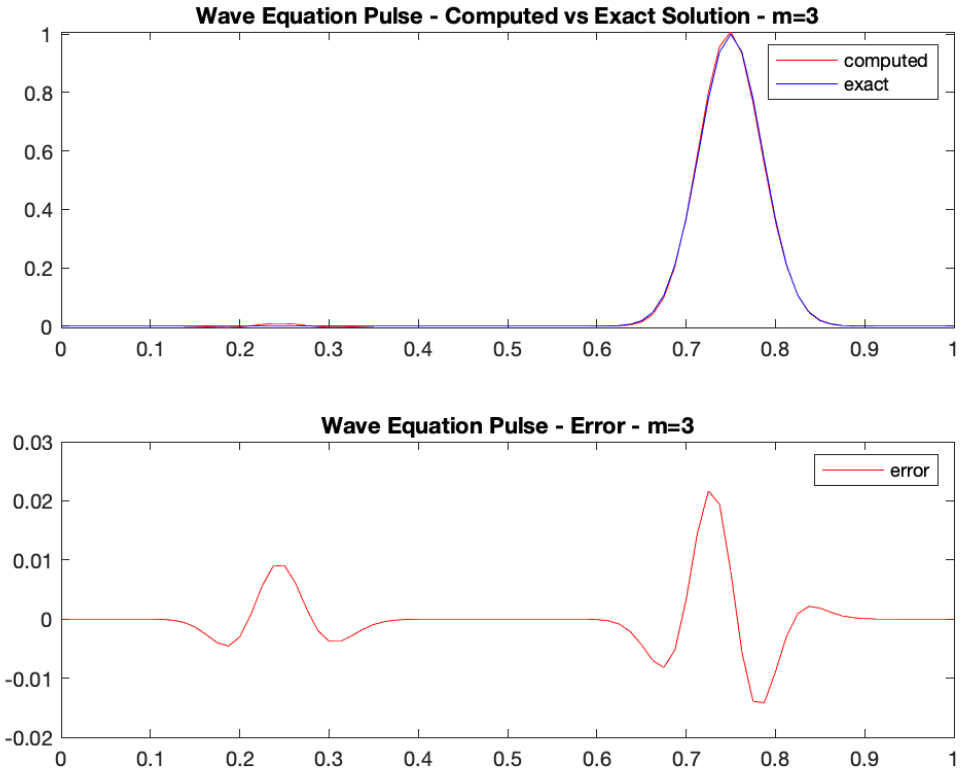

Figure 4: Output

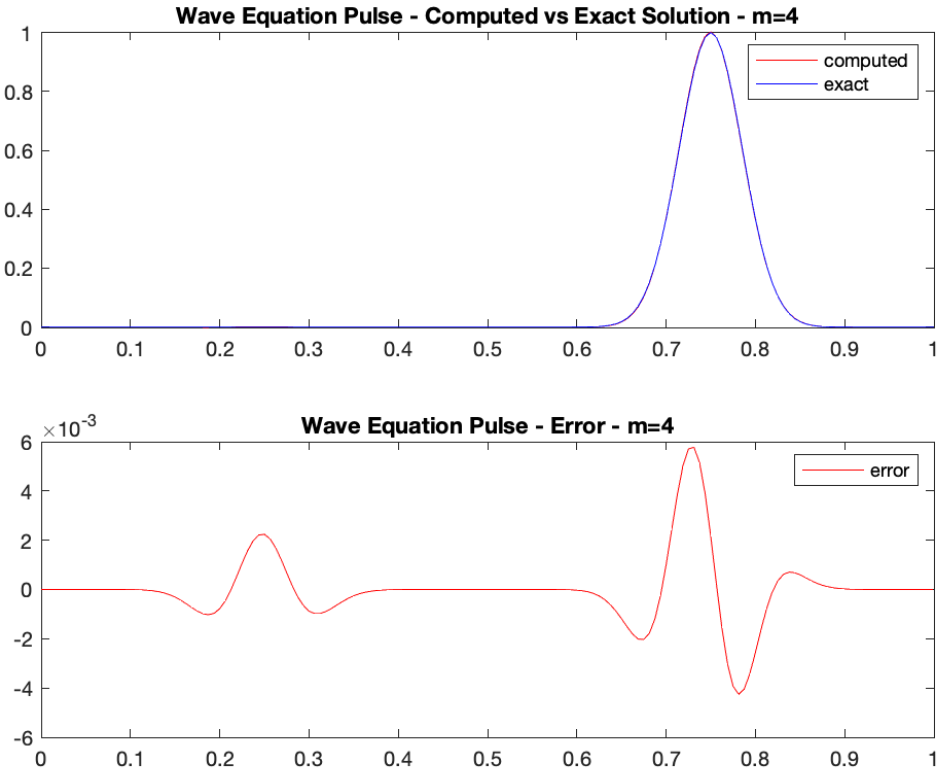

Figure 5: Output