MATH-6840
Vignesh Ramakrishnan
Due: Monday April 18, 2022

Problem Set 9

1. (20 pts.) Consider the linear hyperbolic system

$$\mathbf{u}_t + A\mathbf{u}_x = 0$$

where A is a constant coefficient matrix. Now introduce the upwind method

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^n - A_+ \frac{\Delta t}{\Delta x}\left(\mathbf{v}_j^n - \mathbf{v}_{j-1}^n\right) - A_- \frac{\Delta t}{\Delta x}\left(\mathbf{v}_{j+1}^n - \mathbf{v}_j^n\right)$$

where $A_\pm$ are the matrices constructed using the positive/negative eigenvalues and $A = A_+ + A_-$ as discussed in class. For each A given below do the following

(a) Find the matrices A_+ and A_- in the upwind method above.

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 3 & -3 \\ 1 & 2 & -1 \\ 1 & 3 & -2 \end{bmatrix}$$

$$\begin{aligned} Av =& \ \lambda v \\ (A - \lambda I)v =& \ 0 \end{aligned}$$

$$\begin{bmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{bmatrix} v = 0 \qquad \begin{bmatrix} 2-\lambda & 3 & -3 \\ 1 & 2-\lambda & -1 \\ 1 & 3 & -2-\lambda \end{bmatrix} v = 0$$

The characteristic equations are

$$(\lambda - 3)^2 - 1 = 0 \qquad -(\lambda-2)\left(\lambda^2 - 1\right) + 3(\lambda+1) - 3\left(\lambda+1\right) = 0$$

$$\lambda = 2, 4 \qquad \lambda = 2, 1, -1$$

Using these λ's we can find out the eigen-vectors as follows,

$$(\lambda = 2), \ \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} v = 0 \qquad (\lambda = 4), \ \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} v = 0$$

Therefore, $R = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ which contains the eigen-vectors as its column entries.

Similar procedure is followed for the second case and $R = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix}$.

Now, the eigenvalue decomposition of square matrix $A = R\Lambda R^{-1}$, where, Λ is a diagonal matrix containing the eigen-values.

$$\Lambda = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

This can be split up and written as Λ^+ and Λ^- such that $\Lambda = \Lambda^+ + \Lambda^-$.

$$\Lambda^+ = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \qquad \Lambda^+ = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Lambda^- = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \Lambda^- = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Therefore,

$$\begin{aligned} A &= R\Lambda R^{-1} \\ A &= R\left(\Lambda^+ + \Lambda^-\right) R^{-1} \\ A &= R\Lambda^+ R^{-1} + R\Lambda^- R^{-1} \\ A &= A^+ + A^- \end{aligned}$$

$$A^+ = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \qquad A^+ = \begin{bmatrix} 2 & 2 & -2 \\ 1 & 2 & -1 \\ 1 & 2 & -1 \end{bmatrix}$$

$$A^- = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad A^- = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

(b) If the PDE is defined for $x \in (-1, 1)$ how many boundary conditions are needed at $x = -1$, and how many at $x = 1$?

For the first case, we need two boundary conditions at $x = -1$ since both the eigenvalues are positive, which means the waves propagate from the left to right. No boundary condition is needed at $x = 1$.

For the second case, we need two boundary conditions at $x = -1$ and one at $x = 1$, since we have 2 positive eigenvalues and one negative eigenvalue.

(c) Given BCs of the type just derived, determine an exact solution for the second of the two problems. Note that you may find it convenient to use either homogeneous or non-homogeneous BCs, it is your choice.

$$\begin{aligned} \mathbf{u}_t + A\mathbf{u}_x &= \mathbf{0} \\ R^{-1}\mathbf{u}_t + R^{-1}R\Lambda R^{-1}\mathbf{u}_x &= \mathbf{0} \end{aligned}$$

Let $R^{-1}\mathbf{u} = \mathbf{w}$, then

$$\mathbf{w}_t + \Lambda\mathbf{w}_x = \mathbf{0}$$

These are three linear advection problems of the form

$$w_t + cw_x = 0$$

$w = e^{-a(x-ct+b)^2}$ is an exact solution to the PDE described above.

$$\begin{aligned} w_t &= 2ac\,(x - ct + b)\,e^{-a(x-ct+b)^2} \\ w_x &= -\,2a\,(x - ct + b)\,e^{-a(x-ct+b)^2} \end{aligned}$$

Hence,

$$w_t + cw_x = 0$$

Hence, the chosen exact solutions are

$$\mathbf{w} = \begin{bmatrix} e^{-10(x-\lambda_1 t+0.1)^2} \\ e^{-10(x-\lambda_2 t)^2} \\ e^{-10(x-\lambda_3 t-0.1)^2} \end{bmatrix}$$

where, λ_1, λ_2 and λ_3 are the eigenvalues of the matrix A. Two Dirchlet Boundary conditions at $x = -1$ for $\mathbf{w}[0]$ and $\mathbf{w}[1]$ are set up and one Dirchlet Boundary condition at $x = 1$ for $\mathbf{w}[2]$.

(d) Implement the upwind method with BCs. Use your exact solution from (c) to verify first-order convergence.

The domain is discretized with 2 ghost points on either ends of the domain. $\Delta x = 2/N$ and $x_j = -1 + j\Delta x$ where $j = 0, 1, 2, \ldots\ldots N$.

The first order scheme is modified a little bit and written as,

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^n - \Lambda^+ \frac{\Delta t}{\Delta x}\left(\mathbf{v}_j^n - \mathbf{v}_{j-1}^n\right) - \Lambda^- \frac{\Delta t}{\Delta x}\left(\mathbf{v}_{j+1}^n - \mathbf{v}_j^n\right)$$

The Boundary conditions are written as,

$$\begin{aligned} \mathbf{v}_{-1}^n &= 2\alpha_{\mathbf{1}}(\mathbf{t^n}, -\mathbf{1}) - \mathbf{v_1^n} \\ \mathbf{v}_{N+1}^n &= 2\alpha_{\mathbf{2}}(\mathbf{t^n}, \mathbf{1}) - \mathbf{v_{N-1}^n} \end{aligned}$$

Here, $\alpha_1(t)$ is the vector of Left Boundary conditions and $\alpha_2(t)$ is the vector of Right Boundary conditions. Then $\mathbf{u}$ is recovered as $R\mathbf{w}$. The error convergence plot is shown in Fig 1.

(e) (extra credit) Using a method-of-lines approach and RK-4, implement the standard centered second-order discretization for the original linear hyperbolic system. Use your exact solution from (c) to verify second-order convergence.

The second order scheme is written as,

$$\frac{d\mathbf{v}}{dt} = -\,\Lambda^+ \frac{\mathbf{v}_{j+1} - \mathbf{v}_{j-1}}{2\Delta x} - \Lambda^- \frac{\mathbf{v}_{j+1} - \mathbf{v}_{j-1}}{2\Delta x}$$

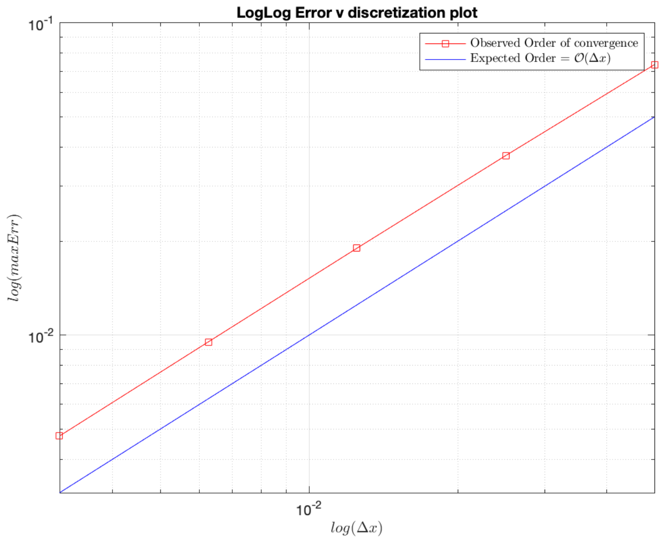

Figure 1: 1st order convergence for Upwind-scheme

The time integration scheme for this set of ODEs is RK-4 and it is formulated as follows,

$$
\begin{aligned}
\mathbf{k^1} &= -\Lambda^{+}\frac{\mathbf{v}^n_{j+1}-\mathbf{v}^n_{j-1}}{2\Delta x}-\Lambda^{-}\frac{\mathbf{v}^n_{j+1}-\mathbf{v}^n_{j-1}}{2\Delta x}\\
\mathbf{k^2} &= -\Lambda^{+}\frac{\left(\mathbf{v}^n_{j+1}+\frac{\Delta t}{2}\mathbf{k^1_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\frac{\Delta t}{2}\mathbf{k^1_{j-1}}\right)}{2\Delta x}-\Lambda^{-}\frac{\left(\mathbf{v}^n_{j+1}+\frac{\Delta t}{2}\mathbf{k^1_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\frac{\Delta t}{2}\mathbf{k^1_{j-1}}\right)}{2\Delta x}\\
\mathbf{k^3} &= -\Lambda^{+}\frac{\left(\mathbf{v}^n_{j+1}+\frac{\Delta t}{2}\mathbf{k^2_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\frac{\Delta t}{2}\mathbf{k^2_{j-1}}\right)}{2\Delta x}-\Lambda^{-}\frac{\left(\mathbf{v}^n_{j+1}+\frac{\Delta t}{2}\mathbf{k^2_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\frac{\Delta t}{2}\mathbf{k^2_{j-1}}\right)}{2\Delta x}\\
\mathbf{k^4} &= -\Lambda^{+}\frac{\left(\mathbf{v}^n_{j+1}+\Delta t\mathbf{k^3_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\Delta t\mathbf{k^3_{j-1}}\right)}{2\Delta x}-\Lambda^{-}\frac{\left(\mathbf{v}^n_{j+1}+\Delta t\mathbf{k^3_{j+1}}\right)-\left(\mathbf{v}^n_{j-1}+\Delta t\mathbf{k^3_{j-1}}\right)}{2\Delta x}\\
\mathbf{v}^{n+1}_j &= \mathbf{v}^n_j+\frac{\Delta t}{6}\left(\mathbf{k^1}+\mathbf{2k^2}+\mathbf{2k^3}+\mathbf{k^4}\right)
\end{aligned}
$$

The Boundary conditions are written as,

$$
\begin{aligned}
\mathbf{v}^n_{-1} &= 2\alpha_{\mathbf{1}}(\mathbf{t^n},\mathbf{-1})-\mathbf{v^n_1}\\
\mathbf{v}^n_{N+1} &= 2\alpha_{\mathbf{2}}(\mathbf{t^n},\mathbf{1})-\mathbf{v^n_{N-1}}
\end{aligned}
$$

Here, $\alpha_1(t)$ is the vector of Left Boundary conditions and $\alpha_2(t)$ is the vector of Right Boundary conditions. Then $\mathbf{u}$ is recovered as $R\mathbf{w}$. The code for both parts (d) and (e) is listed in Listing 1. The error convergence plot is shown in Fig 2.

```
function [max_err,e,u] = VectorAdvectionEqn(N,nStep,tf,A,fOption,iOption)
% function has 2 options for exact solution
% fOption==1, A should be of size(3).

% domain setup
xlim1 = -1;
xlim2 = 1;
```

```
tlim1 = 0;
tlim2 = tf;

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

% eigen value decomposition
[m,~] = size(A);

if (fOption==1)
    assert(m==3,'Exact Solution choice has a mismatch in dimensions');
elseif (fOption==2)
    assert(m==2,'Exact Solution choice has a mismatch in dimensions');
end
[R,L] = eig(A);

Lp = zeros(m);
Lm = zeros(m);

for i=1:m
    if L(i,i)>=0
        Lp(i,i) = L(i,i);
    else
        Lm(i,i) = L(i,i);
    end
end

% domain discretization
ng   = 1;
NTot = N+1+2*ng;
ja   = ng+1;
jb   = NTot-ng;

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

% setting solution variables
w = zeros(m,NTot);
u = zeros(m,NTot);

% setting Integration constants
k1 = zeros(m,NTot);
k2 = zeros(m,NTot);
k3 = zeros(m,NTot);
k4 = zeros(m,NTot);

for j=1:length(x)
    w(:,j) = getEx(x(j),tlim1,L,fOption);
end
% set Boundary conditions
w(:,ng)   = 2*getEx(xlim1,tlim1,L,fOption)-w(:,ja+1);
w(:,NTot) = 2*getEx(xlim2,tlim1,L,fOption)-w(:,jb-1);

u = R*w;
% figure(1)
% for i=1:m
%     subplot(m,1,i)
```

```
%      plot(x(ja:jb),w(i,ja:jb),'ks-');
% end

% figure(2)
for i=2:length(t)
    wold = w;
    if iOption==1
        for j=ja:jb
            w(:,j) = wold(:,j)-Lp*(dt/dx)*(wold(:,j)-wold(:,j-1))-...
                    Lm*(dt/dx)*(wold(:,j+1)-wold(:,j));
        end
    elseif iOption==2
        for j=ja:jb
            k1(:,j) = -Lp*(0.5/dx)*(wold(:,j+1)-wold(:,j-1))-...
                    Lm*(0.5/dx)*(wold(:,j+1)-wold(:,j-1));
        end
        for j=ja:jb
            k2(:,j) = -Lp*(0.5/dx)*( (wold(:,j+1)+(dt/2)*k1(:,j+1))-(wold(:,j-1)+(dt/2)*k1(:,j-1)) )-...
                Lm*(0.5/dx)*( (wold(:,j+1)+(dt/2)*k1(:,j+1))-(wold(:,j-1)+(dt/2)*k1(:,j-1)) );
        end
        for j=ja:jb
            k3(:,j) = -Lp*(0.5/dx)*( (wold(:,j+1)+(dt/2)*k2(:,j+1))-(wold(:,j-1)+(dt/2)*k2(:,j-1)) )-...
                Lm*(0.5/dx)*( (wold(:,j+1)+(dt/2)*k2(:,j+1))-(wold(:,j-1)+(dt/2)*k2(:,j-1)) );
        end
        for j=ja:jb
            k4(:,j) = -Lp*(0.5/dx)*( (wold(:,j+1)+dt*k3(:,j+1))-(wold(:,j-1)+dt*k3(:,j-1)) )-...
                Lm*(0.5/dx)*( (wold(:,j+1)+dt*k3(:,j+1))-(wold(:,j-1)+dt*k3(:,j-1)) );
        end
        for j=ja:jb
            w(:,j) = wold(:,j)+(dt/6)*(k1(:,j)+2*k2(:,j)+2*k3(:,j)+k4(:,j));
        end
    end
    % set BC
    w(:,ng)   = 2*getEx(xlim1,t(i),L,fOption)-w(:,ja+1);
    w(:,NTot) = 2*getEx(xlim2,t(i),L,fOption)-w(:,jb-1);
    u = R*w;
%     cla
%     plot(x(ja:jb),w(1,ja:jb),'bs-');
%     hold on
%     plot(x(ja:jb),w(2,ja:jb),'rs-');
%     plot(x(ja:jb),w(3,ja:jb),'ks-');
%     pause(0.01)
end

for j=1:length(x)
    wex(:,j) = getEx(x(j),tlim2,L,fOption);
end
% set Boundary conditions
wex(:,ng)   = 2*getEx(xlim1,tlim2,L,fOption)-wex(:,ja+1);
wex(:,NTot) = 2*getEx(xlim2,tlim2,L,fOption)-wex(:,jb-1);
uex = R*wex;
```

```
e = abs(u-uex);
max_err = max(max(e));

% figure
% for i=1:m
%     subplot(m,1,i)
%     plot(x(ja:jb),e(i,ja:jb),'ks-');
% end

end

%% Functions
function wex = getEx(x,t,L,fOption)
% max dimension of the options is 3

if fOption==1
    % vector has dimension 3
    wex(1,1) = exp(-10*(x-L(1,1)*t+0.1)^2);
    wex(2,1) = exp(-10*(x-L(2,2)*t)^2);
    wex(3,1) = exp(-10*(x-L(3,3)*t-0.1)^2);
else
    % vector has dimension 2
    wex = zeros(2,1);
end

end
```

Listing 1: Vector Advection Equation

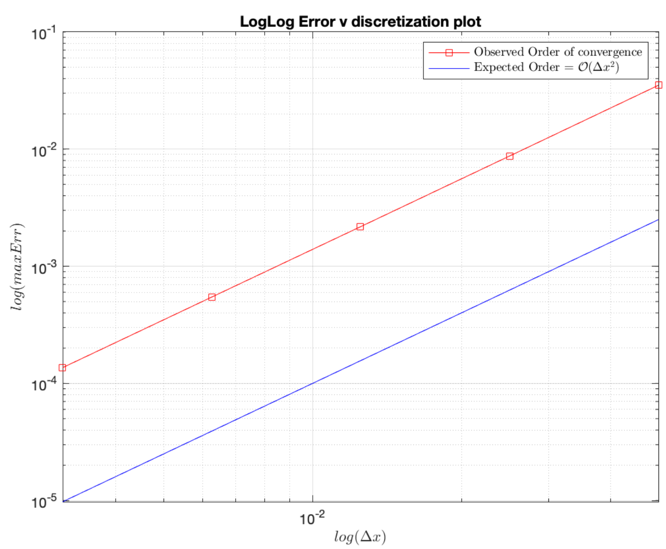

Figure 2: 2nd order convergence for center difference scheme with RK4 time integration

2. (10 pts.) Consider the scalar conservation equation

$$u_t + [f(u)]_x = 0,$$

for $|x| < \infty$, $t > 0$, and $u(x, 0) = u_0(x)$. For each of the following flux functions $f(u)$

$$f(u) = 2u^4, \qquad f(u) = e^{2u}$$

(a) Determine a formula for the propagation speed S of a discontinuity between two states u_L and u_R (L and R for left and right respectively).
Using chain rule, the PDE can be written as,

$$u_t + \frac{df}{du} u_x = 0$$

Now, the characteristics are given by $\frac{dx}{dt} = \frac{df}{du}$. In case of discontinuities, if the PDE is integrated over an interval $[a, b]$ containing the discontinuity, then

$$\begin{aligned}
\frac{d}{dt}\int_a^b u\ dx &= \frac{d}{dt}\int_a^{s(t)} u\ dx + \frac{d}{dt}\int_{s(t)}^b u\ dx \\
&= \int_a^{s(t)} u_t\ dx + \frac{ds}{dt}u(s^-,t) + \int_{s(t)}^b u_t\ dx - \frac{ds}{dt}u(s^+,t) \\
&= \int_a^{s(t)} -(f(u))_x\ dx + \frac{ds}{dt}u(s^-,t) + \int_{s(t)}^b -(f(u))_x\ dx - \frac{ds}{dt}u(s^+,t) \\
&= -(f(u))\,|_a^b + [f(u)] - [u]\frac{ds}{dt} \\
\cancel{\frac{d}{dt}\int_a^b u\ dx} &= \cancel{-(f(u))\,|_a^b} + [f(u)] - [u]\frac{ds}{dt}
\end{aligned}$$

Therefore, speed of discontinuity is given by $\frac{ds}{dt} = \frac{[f]}{[u]}$ across discontinuity.

(b) Determine the speed S when $u_L = 2$, and $u_R = 1$.
Case 1: $f(u) = 2u^4$

$$\begin{aligned}
\frac{ds}{dt} &= \frac{f(u_R) - f(u_L)}{u_R - u_L} \\
&= \frac{2u_R^4 - 2u_L^4}{u_R - u_L} = \frac{2 - 2^5}{-1} = 30
\end{aligned}$$

Case 2: $f(u) = e^{2u}$

$$\begin{aligned}
\frac{ds}{dt} &= \frac{f(u_R) - f(u_L)}{u_R - u_L} \\
&= \frac{e^{2u_R} - e^{2u_L}}{u_R - u_L} = \frac{e^2 - e^4}{-1} = e^4 - e^2
\end{aligned}$$

3. (20 pts.) Consider the conservation equation from #2 above with $f(u) = e^{2u}$.

(a) Find the characteristic form of the equation. What is the characteristic speed?
Characteristic form of the equation is,

$$\begin{aligned}
\frac{dx}{dt} &= \frac{df}{du} \\
x(t) &= \frac{df}{du}t + x_0 \\
x_j(t) &= 2e^{2u_j}t + x_j
\end{aligned}$$

where, the characteristic speed is $\frac{df}{du} = 2e^{2u}$.

(b) Assuming $u_0(x) = \frac{1}{2}(u_L + u_R) + \frac{1}{2}(u_R - u_L)\tanh(10x)$, use the characteristics, and the fact that u is constant along characteristics, to sketch qualitative solutions at various times for the following 2 cases;
The characteristic solutions at several final times are plotted in Fig 3. The initial conditions (u_L and u_R) are mentioned in the figure's title.

   i. $u_L = -1$, $u_R = 1$.
   ii. $u_L = 1$, $u_R = -1$.

(c) Write a conservative upwind code and verify your predictions from (b) above.
The code for a conservative upwind scheme is attached below in Listing 2. The solutions are plotted in Fig 4.

```
function [ChSol, CoSol] = NonlinearConservation(N,CFL,tf,uL,uR,iOption,
    fOption)
% Trial runs
% NonlinearConservation(200,0.9,2,2,1,1,2)     - Burgers Equtation, step
    IC
% NonlinearConservation(500,0.01,0.5,1,-1,2,1)- Exponential Flux,
% interesting Solution
% NonlinearConservation(500,0.01,0.5,-1,1,2,1)- Exponential Flux,
% uninteresting solution

% domain setup
xlim1 = -5;
xlim2 = 5;
tlim1 = 0;
tlim2 = tf;

% calculated parameters
dx = (xlim2-xlim1)/N;

% domain discretization
ng   = 1;
NTot = N+1+2*ng;
ja   = ng+1;
jb   = NTot-ng;

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

u = zeros(NTot,1);
if iOption==1
    for j=1:NTot
        if x(j)<0
            u(j) = uL;
        else
            u(j) = uR;
        end
    end
elseif iOption==2
    for j=1:NTot
        u(j) = 0.5*(uL+uR)+0.5*(uR-uL)*tanh(x(j));
    end
end

u0 = u; % set IC for characteristic solution

```

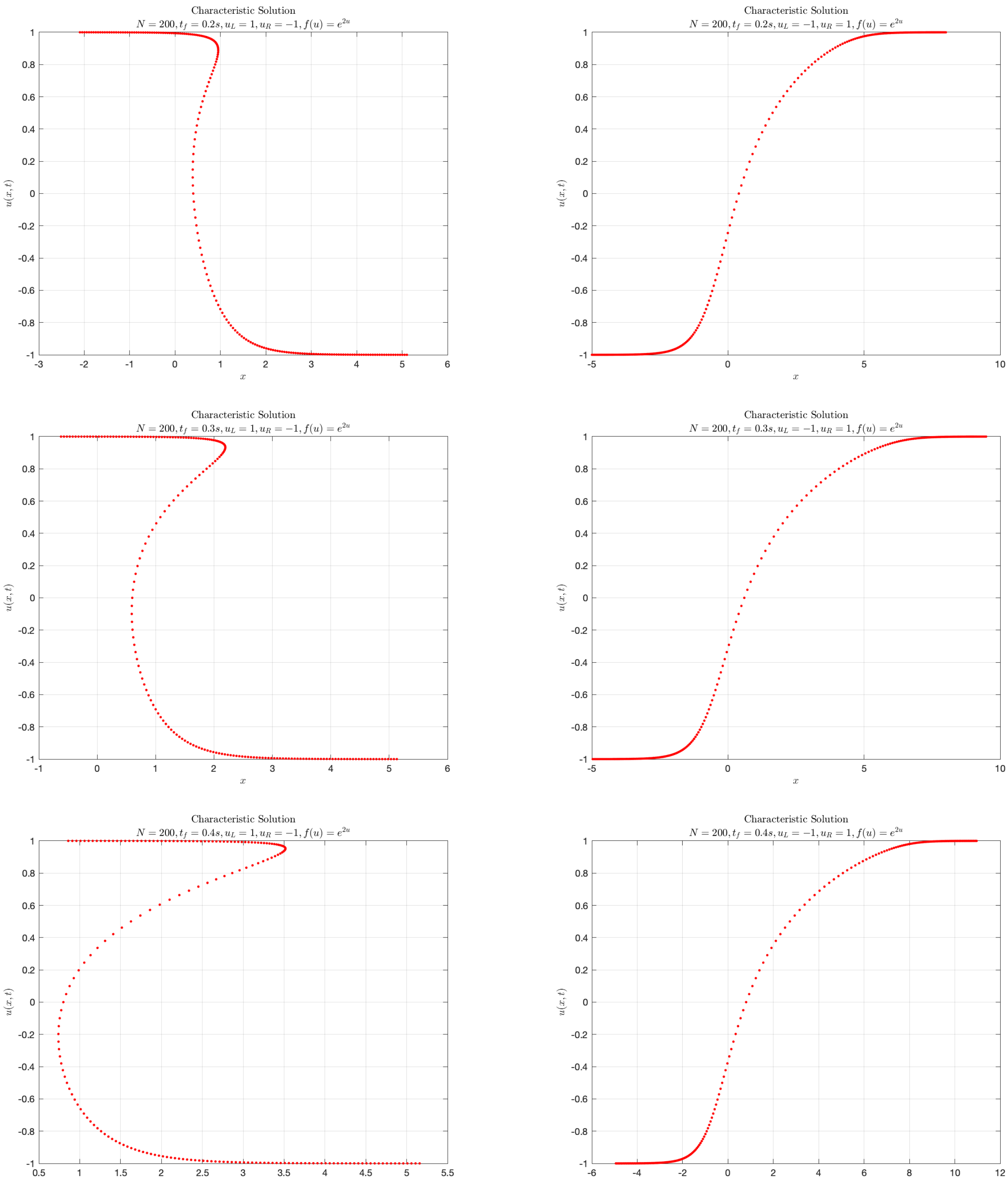

Figure 3: Characteristic Solutions at different final times

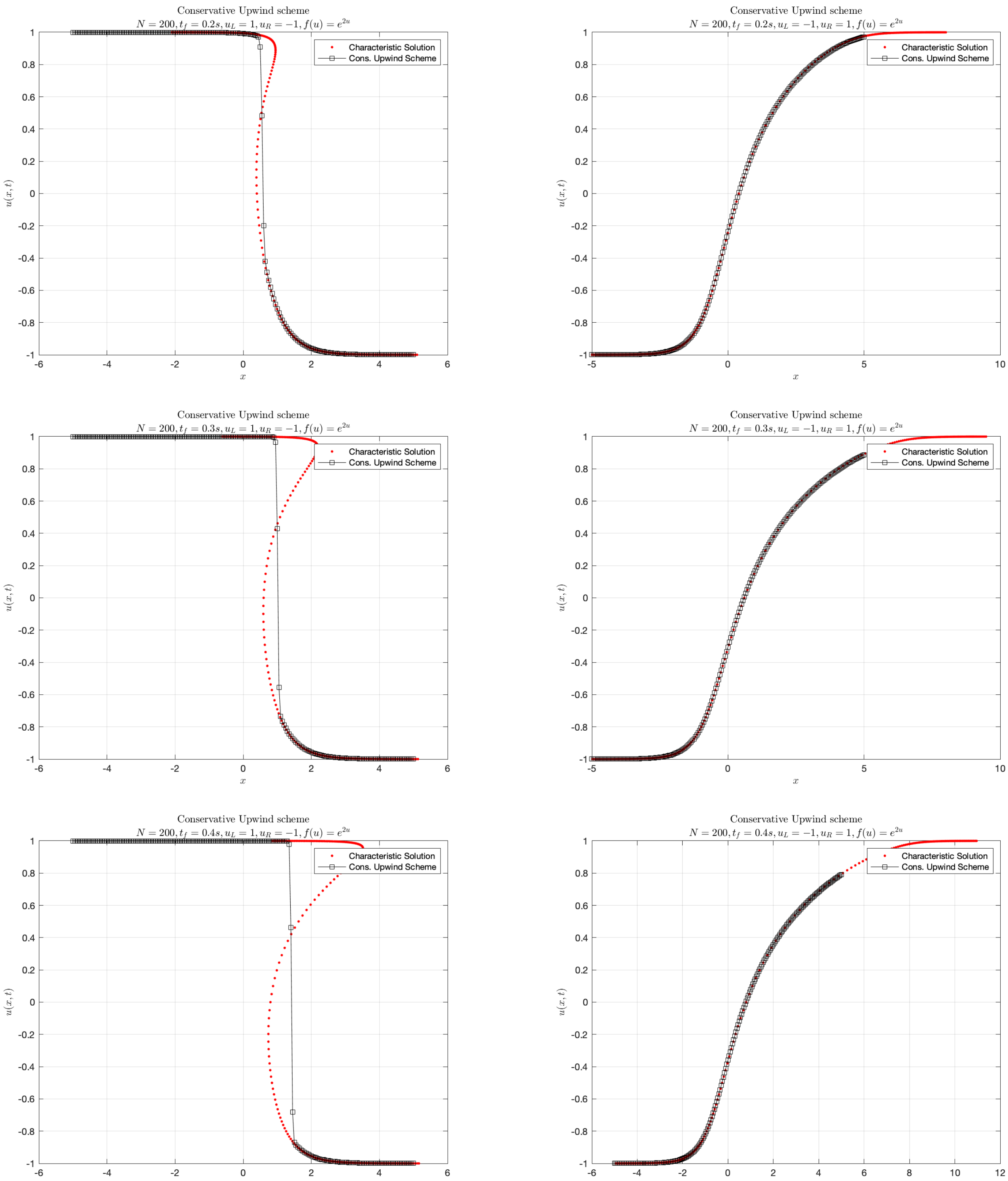

Figure 4: Upwind Scheme Solutions at different final times

```
dt = CFL*dx/max(abs(Cspeed(u,fOption)));
t  = tlim1;

% Nchar = 75;
% figure
% for j = 1:floor(N/Nchar):N
% %   m = 1/max(1e-14,Cspeed(u0(j),fOption));
%     if abs(Cspeed(u0(j),fOption))>1e-14
%         m = 1/Cspeed(u0(j),fOption);
%     else
%         m = 1/(1e-14);
%     end
%
%   y = m*(x-x(j));
%   hold on
%   plot(x,y,'Color',[1*j/N,.2,1-j/N]);
%   hold off
%   ylim([0,tf]);
%   xlabel('x');
%   ylabel('t');
% end
% pause

xc = zeros(NTot,1);
yc = zeros(NTot,1);

% figure
while t<tlim2
    uold = u;
    % conservative scheme
    for j=ja:jb
        fj   = flux(uold(j),fOption);
        fjm1 = flux(uold(j-1),fOption);
        u(j) = uold(j)-(dt/dx)*(fj-fjm1);
    end

    % set Boundary conditions
    u(ng)   = u(ja);
    u(NTot) = u(jb);

    % characteristic solution
    for j = 1:NTot
        m = Cspeed(u0(j),fOption);
        xc(j) = t*m+x(j);
        yc(j) = u0(j);
    end

%     plot(x(ja:jb),u(ja:jb),'ks-');
%     hold on
%     plot(xc,yc,'r.');
%     hold off
%     xlabel('x')
%     ylabel('u(x)')
%     drawnow
%     pause(0.01)
    t  = t + dt;

    dt = CFL*dx/max(abs(Cspeed(u,fOption)));
end
```

```

% characteristic solution at final time
ChSol.xc = xc;
ChSol.yc = yc;

% conservative numerical scheme solution at final time
CoSol.x  = x(ja:jb);
CoSol.u  = u(ja:jb);

end

%%
function f = flux(u,fOption)
if fOption==1 % exponential
    f = exp(2*u);
elseif fOption==2 % burgers
    f = 0.5*u^2;
elseif fOption==3 % 4th order flux
    f = 2*u^4;
elseif fOption==4 % upwind flux
    f = -2*u;
end
end

function dfdu = Cspeed(u,fOption)
if fOption==1
    dfdu = 2*exp(2*u);
elseif fOption==2
    dfdu = u;
elseif fOption==3
    dfdu = 8*u.^3;
elseif fOption==4
    dfdu = -2;
end
end
```

Listing 2: Conservative Upwind scheme for Nonlinear PDE

The script used to generate the plots is attached in Listing 3.

```
%% Convergence analysis - Upwind scheme - Forward Euler
clc
clear all

N       = [20 40 80 160 320];
nStep   = 10*N;
tf      = 0.1;
fOption = 1;
iOption = 1;
A       = [2 3 -3;1 2 -1;1 3 -2];

for i=1:length(N)
    [max_err(i),~,~] = VectorAdvectionEqn(N(i),nStep(i),tf,A,fOption,iOption);
end
dx = 1./N;

logE  = log(max_err);
logdx = log(dx);

```

```
P      = polyfit(logdx,logE,1);
slope = P(1);
exp_order = 1;
logEfit = exp_order*logdx;
Efit    = exp(logEfit);

figure
loglog(dx,max_err,'rs-');
hold on;
grid on;
loglog(dx,Efit,'b');
legend('Observed Order of convergence',...
    'Expected Order = $\mathcal{O}(\Delta x)$','Interpreter','latex');
xlabel('$log(\Delta x)$','Interpreter','latex');
ylabel('$log(maxErr)$','Interpreter','latex');
title('LogLog Error v discretization plot');
print('Q1_ErrPlot','-dpng');

%% Convergence analysis - Central Difference - RK4
clc
clear all

N       = [20 40 80 160 320];
nStep   = 10*N;
tf      = 0.1;
fOption = 1;
iOption = 2;
A       = [2 3 -3;1 2 -1;1 3 -2];

for i=1:length(N)
    [max_err(i),~,~] = VectorAdvectionEqn(N(i),nStep(i),tf,A,fOption,iOption);
end
dx = 1./N;

logE  = log(max_err);
logdx = log(dx);

P      = polyfit(logdx,logE,1);
slope = P(1);
exp_order = 2;
logEfit = exp_order*logdx;
Efit    = exp(logEfit);

figure
loglog(dx,max_err,'rs-');
hold on;
grid on;
loglog(dx,Efit,'b');
legend('Observed Order of convergence',...
    'Expected Order = $\mathcal{O}(\Delta x^2)$','Interpreter','latex');
xlabel('$log(\Delta x)$','Interpreter','latex');
ylabel('$log(maxErr)$','Interpreter','latex');
title('LogLog Error v discretization plot');
print('Q1_ErrPlot_RK4','-dpng');
%%
clc
clear all

N   = 200;
```

```
CFL = 0.1;
tf  = [0.2 0.3 0.4];
uL  = [1 -1];
uR  = [-1 1];

iOption = 2; % tanh function
fOption = 1; % exponential flux

k = 1;
for i=1:length(tf)
    for j=1:length(uL)
        [c,n] = NonlinearConservation(N,CFL,tf(i),uL(j),uR(j),iOption,fOption);

        name1 = 'Q2charac_';
        name1 = strcat(name1,num2str(k));

        name2 = 'Q2_';
        name2 = strcat(name2,num2str(k));

        str = '$N=';
        str = strcat(str,num2str(N));
        str = strcat(str,', t_f =');
        str = strcat(str,num2str(tf(i)));
        str = strcat(str,'s, u_L =');
        str = strcat(str,num2str(uL(j)));
        str = strcat(str,', u_R =');
        str = strcat(str,num2str(uR(j)));
        str = strcat(str,', f(u)=e^{2u}');
        str = strcat(str,'$');

        figure
        plot(c.xc,c.yc,'r.');
        grid on;
        xlabel('$x$','Interpreter','latex');
        ylabel('$u(x,t)$','Interpreter','latex');
        title('Characteristic Solution',str,'Interpreter','latex');
        print(name1,'-dpng');

        figure
        plot(c.xc,c.yc,'r.');
        hold on;
        grid on;
        plot(n.x,n.u,'ks-');
        xlabel('$x$','Interpreter','latex');
        ylabel('$u(x,t)$','Interpreter','latex');
        legend('Characteristic Solution','Cons. Upwind Scheme');
        title('Conservative Upwind scheme',str,'Interpreter','latex');
        print(name2,'-dpng');

        k = k+1;
    end
end
```

Listing 3: Script to generate plots