AE 5327 - Homework 3

James Grisham

11/4/2013

Problem Statement

Given the following model equation:

$$\frac{\partial \phi}{\partial t} + 0.1\,\frac{\partial \phi}{\partial x} = 0.0$$

Perform the following tasks:

a). Derive the amplification factor when the forward Euler scheme is used with the following spatial differencing:

- first-order upwind differencing
- second-order upwind differencing

b). Determine the conditions for which the schemes will be stable. Plot the amplification factors for various CFL numbers.

c). Write a program to solve the model equation approximately using the two schemes with the following conditions:

$$\phi(0,t) = 1$$
$$\phi(x,0) = \frac{1}{1+e^{x-10}}$$
$$0 \le x \le 100$$

d). Determine the critical time step size for your program. How does it compare with the results from part a)?

e). On the same graph, plot the exact and approximate solutions for ϕ vs x at time $t = 20$, 200, 800. Use at least 100 cells and a time step close to the stability limit. Compare the solutions for first-order and second-order upwind. For the same grid size, how does your solution change if you reduce the time step size by a factor of ten?

Solution

Part A

The difference equation for forward Euler, first-order upwind differencing is given by

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + c\,\frac{\phi_i^n - \phi_{i-1}^n}{\Delta x} = 0 \tag{1}$$

and the forward Euler, second-order upwind differencing yields

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + c\,\frac{3\phi_i^n - 4\phi_{i-1}^n + \phi_{i-2}^n}{2\Delta x} = 0 \tag{2}$$

where $c = 0.1$.

The numerical solution to the PDE can be written as

$$N = D + \epsilon$$

where D is the exact solution and ϵ is the error in the numerical solution due to round-off errors. This numerical solution must satisfy the difference equations given above. Also, the exact solution must satisfy the difference equation. Therefore, the error must also satisfy the difference equation.

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} + c\,\frac{\epsilon_i^n - \epsilon_{i-1}^n}{\Delta x} = 0 \tag{3}$$

Now, it is assumed that the error can be written as a series given by

$$\epsilon(x,t) = \sum_m b_m(t) e^{ik_m x} \tag{4}$$

Also, an assumption about the solution is made.

$$b_m(t) = z^n = e^{an\Delta t} = e^{at} \tag{5}$$

Therefore,

$$\epsilon_m(x,t) = e^{at} e^{ik_m x} \tag{6}$$

Inserting (6) into (3) and simplifying,

$$e^{a(t+\Delta t)} e^{ik_m x} - e^{at} e^{ik_m x} = -\frac{c\Delta t}{\Delta x}\left(e^{at} e^{ik_m x} - e^{at} e^{ik_m (x-\Delta x)}\right) \tag{7}$$

Dividing both sides by $e^{at} e^{ik_m x}$,

$$e^{a\Delta t} = 1 - r\left(1 - e^{-ik_m \Delta x}\right) \tag{8}$$

Remembering $G \equiv e^{a\Delta t}$, using Euler's identity, and letting $\beta = k_m x$, (8) becomes

$$\boxed{G(\beta) = 1 - r\,(1 - \cos\beta + i \sin\beta)} \tag{9}$$

Now, the same is done for the second-order upwind scheme. Substituting the error into the difference equation,

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} + c\,\frac{3\epsilon_i^n - 4\epsilon_{i-1}^n + \epsilon_{i-2}^n}{2\Delta x} = 0 \tag{10}$$

Inserting (6) into (10) and rearranging,

$$e^{a(t+\Delta t)}e^{ik_m x} - e^{at}e^{ik_m x} = -\frac{c\Delta t}{2\Delta x}\left(3e^{at}e^{ik_m x} - 4e^{at}e^{ik_m(x-\Delta x)} + e^{at}e^{ik_m(x-2\Delta x)}\right) \tag{11}$$

Dividing both sides by $e^{at}e^{ik_m x}$ and simplifying yields

$$e^{a\Delta t} = 1 - \frac{r}{2}\left(3 - 4e^{-ik_m\Delta x} + e^{-2ik_m\Delta x}\right) \tag{12}$$

Using Euler's identity, (12) becomes

$$\boxed{G(\beta) = 1 - \frac{r}{2}\left(3 - 4\left(\cos\beta - i\,\sin\beta\right) + \cos(2\beta) - i\,\sin(2\beta)\right)} \tag{13}$$

Because of the instability of the low-frequency modes in the second-order upwind method, a different MacCormack method variant was implemented. The difference equation using this scheme is given by

$$\phi_i^{n+1} = \phi_i^n - r\left(\phi_i^n - \phi_{i-1}^n\right) + \frac{1}{2}r(r-1)\left(\phi_i^n - 2\phi_{i-1}^n + \phi_{i-2}^n\right) \tag{14}$$

where r is the CFL number. The amplification factor is derived using the same procedure as before.

$$\boxed{G(\beta) = \frac{r}{2}(r-1)(1 - 2\cos\beta + \cos(2\beta) + 2i\sin\beta - i\sin(2\beta))} \tag{15}$$

Part B

The condition for stability is $|G| \leq 1$. Therefore, the first-order upwind scheme is stable if the following condition is met

$$|1 - r\left(1 - \cos\beta + i\sin\beta\right)| \leq 1 \tag{16}$$

Solving this inequality yields

$$\boxed{0 \leq r \leq 1} \tag{17}$$

Similarly, the condition for stability for the second-order upwind scheme is

$$\left|1 - \frac{r}{2}\left(3 - 4\left(\cos\beta - i\,\sin\beta\right) + \cos(2\beta) - i\,\sin(2\beta)\right)\right| \leq 1 \tag{18}$$

Solving the inequality yields

$$\boxed{0 \leq r \leq \frac{2(\cos\beta - 1)}{3\cos\beta - 5}} \tag{19}$$

So, the CFL number required for the scheme to be stable is dependent upon β. A plot of (19) is shown in Figure 1. This plot shows that in order to meet the stability condition of $|G| \leq 1$, the CFL number must approach zero to get rid of the low frequency instabilities. Because a CFL number of zero is impossible, this method is not considered further.

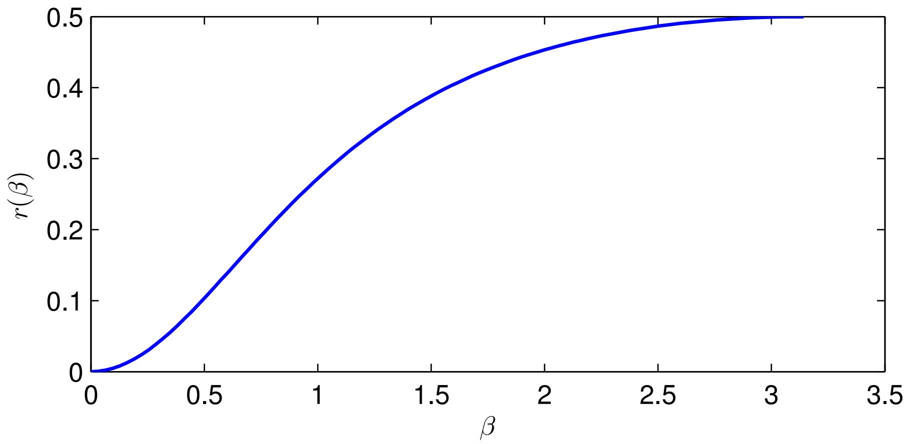

Figure 1. CFL number vs β for the second-order upwind method.

Now, the condition for stability of the MacCormack method variant is

$$\left|1 - \frac{r}{2}\left(3 - 4\left(\cos\beta - i\,\sin\beta\right) + \cos(2\beta) - i\,\sin(2\beta)\right)\right| \leq 1 \tag{20}$$

Solving this inequality yields

$$\boxed{0 \leq r \leq 2} \tag{21}$$

The amplification factors are plotted in Figure 2.

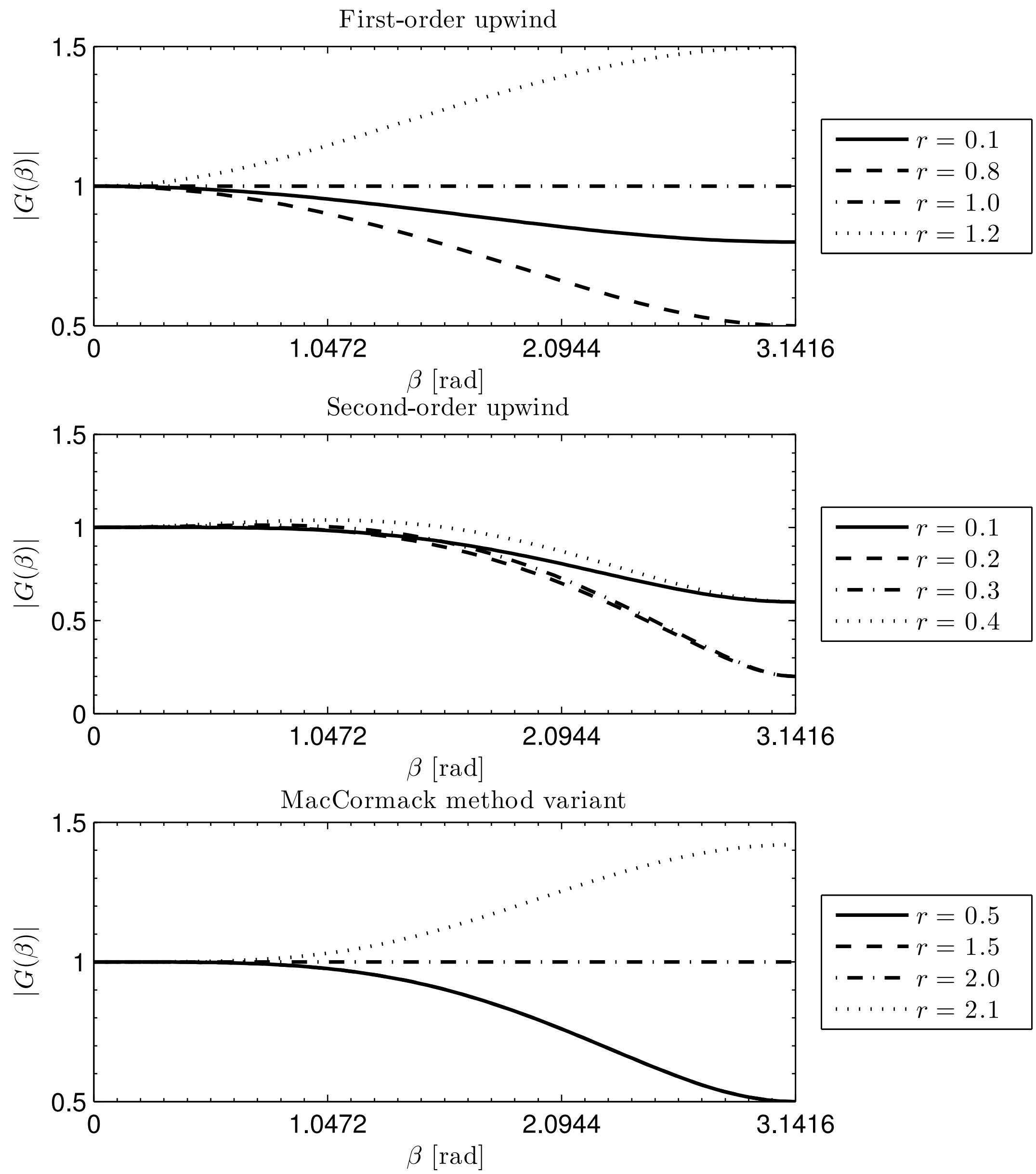

Figure 2. Amplification factors.

Part C

A C++ program was written to numerically approximate the solution to the model equation using all three discretization schemes. The code can be seen in the appendix.

Part D

Assuming $\Delta x = 1$, $\Delta t \leq 10$ for the first-order upwind method. The results in Figure 3 confirm that the critical time step is approximately 10.

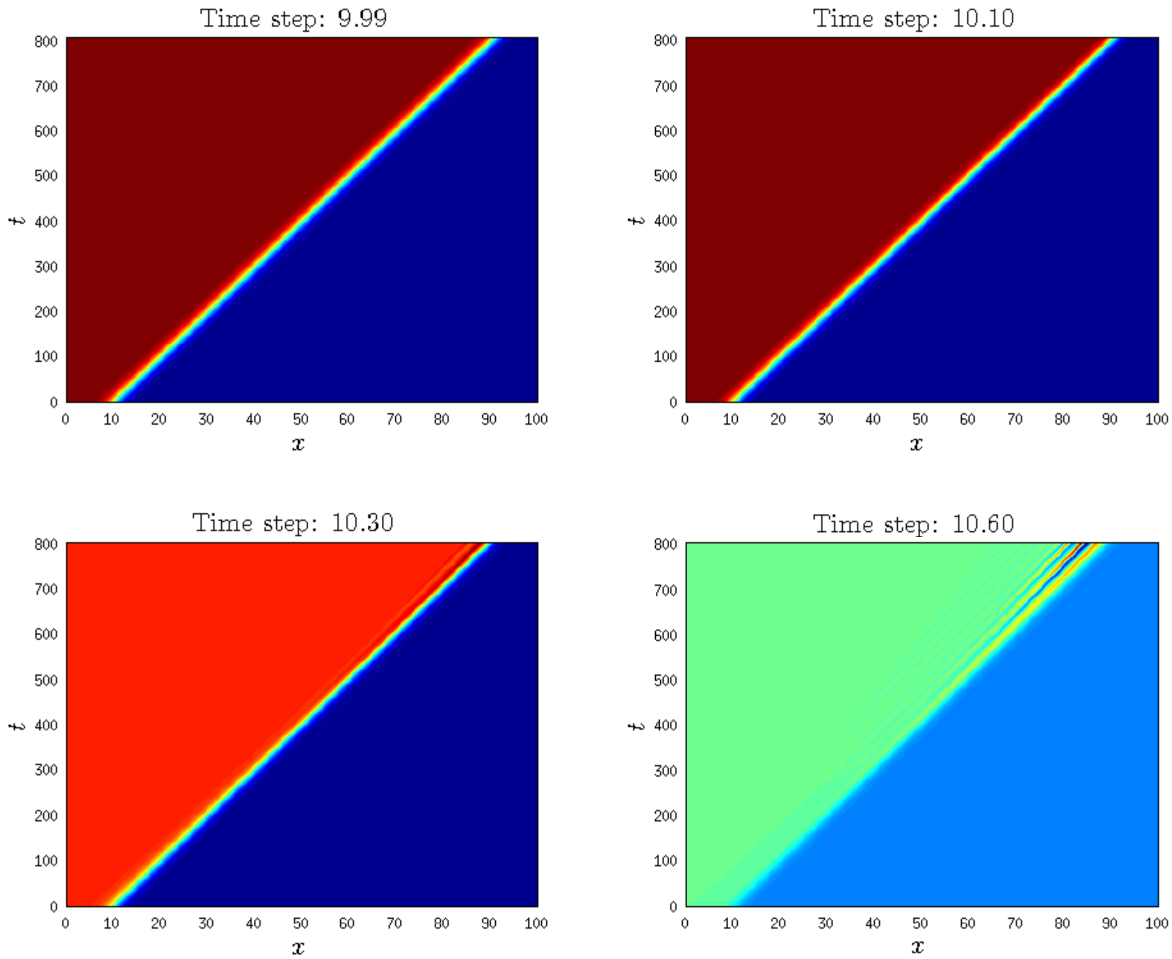

Figure 3. Contour plot of solutions with different time steps.

For the MacCormack method variant, assuming $\Delta x = 1$, $\Delta t \leq 20$. The contour plots in Figure 4 show that the critical time step is approximately 20. Beyond a time step of 20.5, the solution degrades quickly.

The results from the code agree very well with the predictions from the Von Neumann analysis.

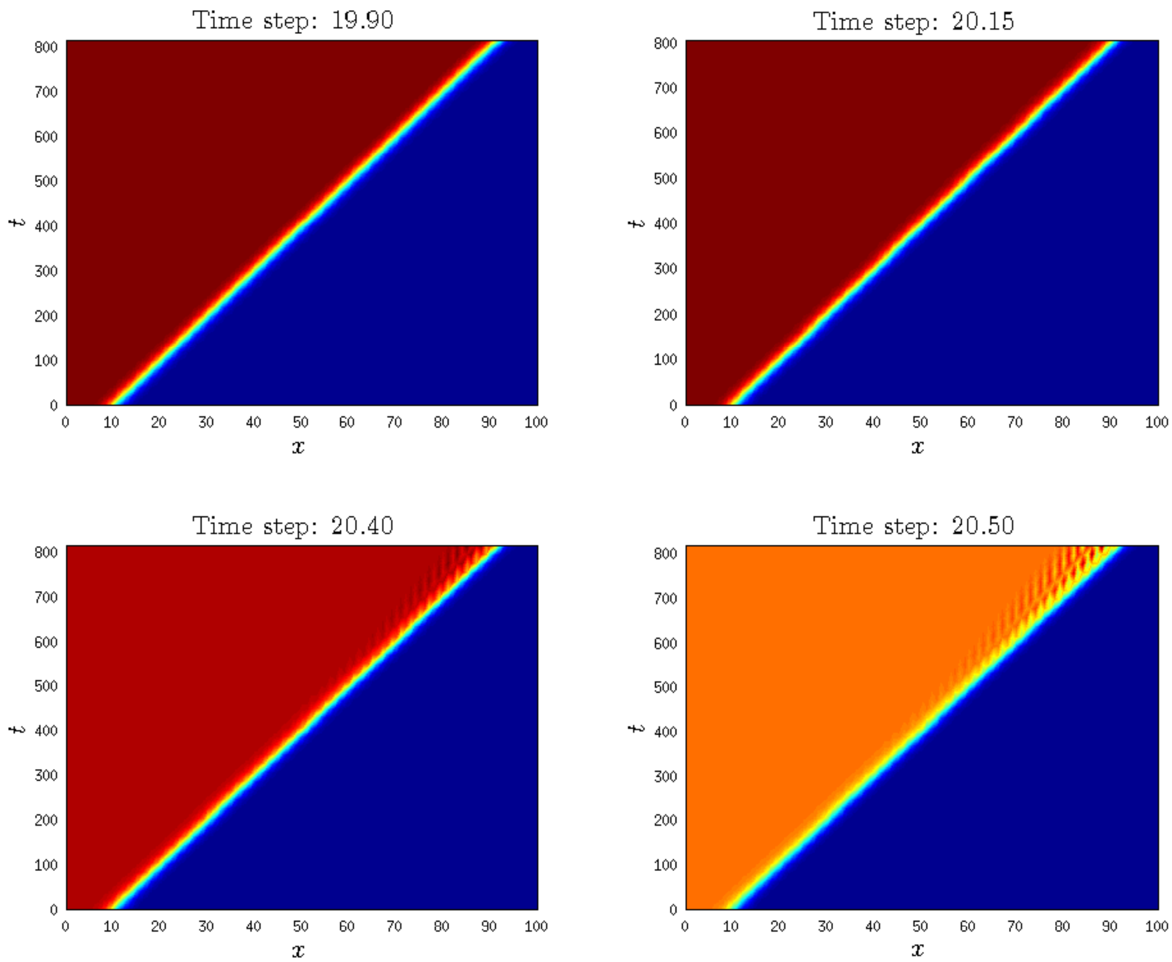

Figure 4. Contour plot of solutions with different time steps.

Part E

A comparison between the solutions for the first-order upwind and MacCormack method variant is shown in Figure 5. The methods agree almost perfectly when the time step is very close to the critical time step.

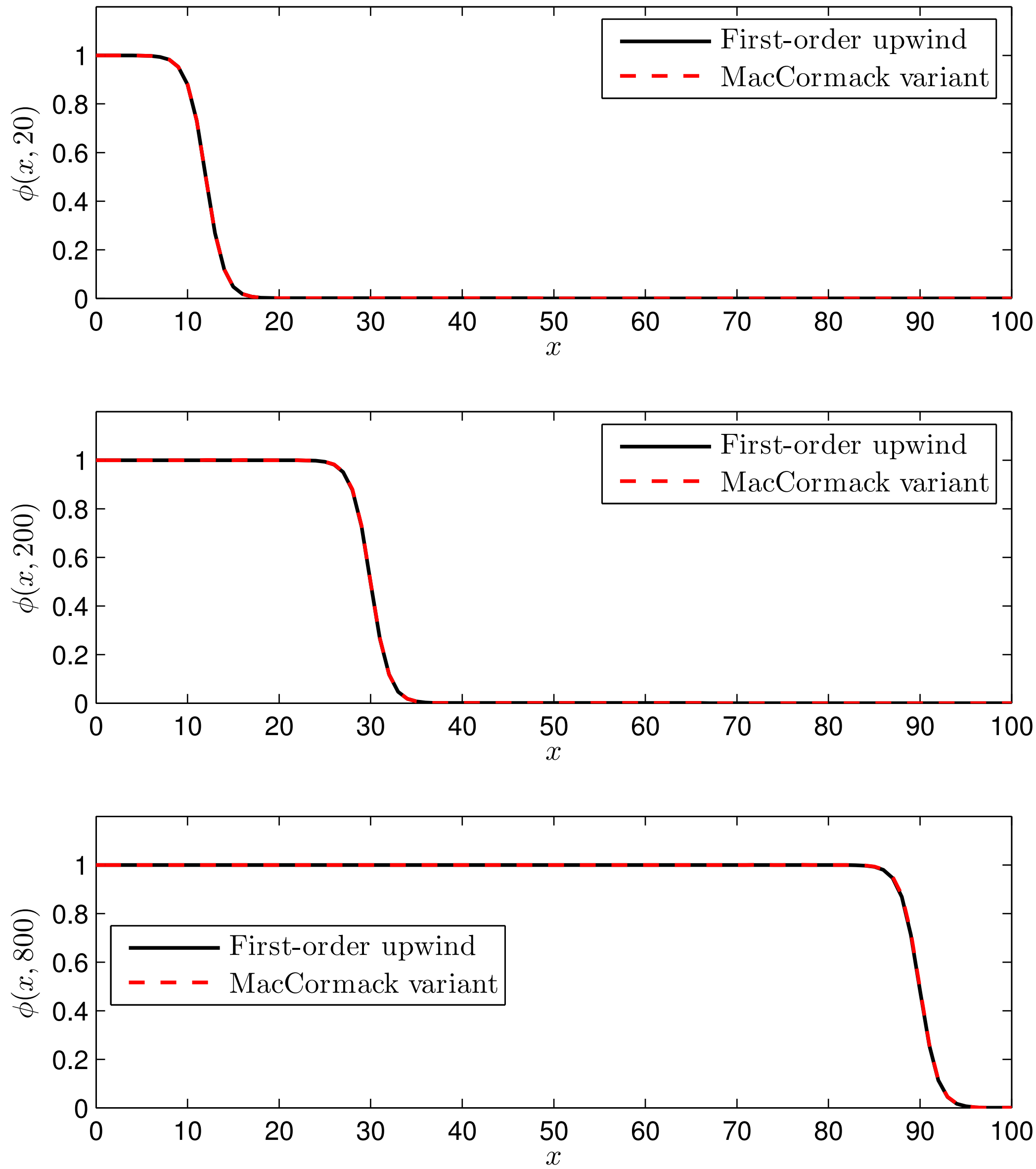

Figure 5. Comparison between solution at selected times.

Next, the time step used to create Figure 5 is reduced by a factor of ten and the results are compared.

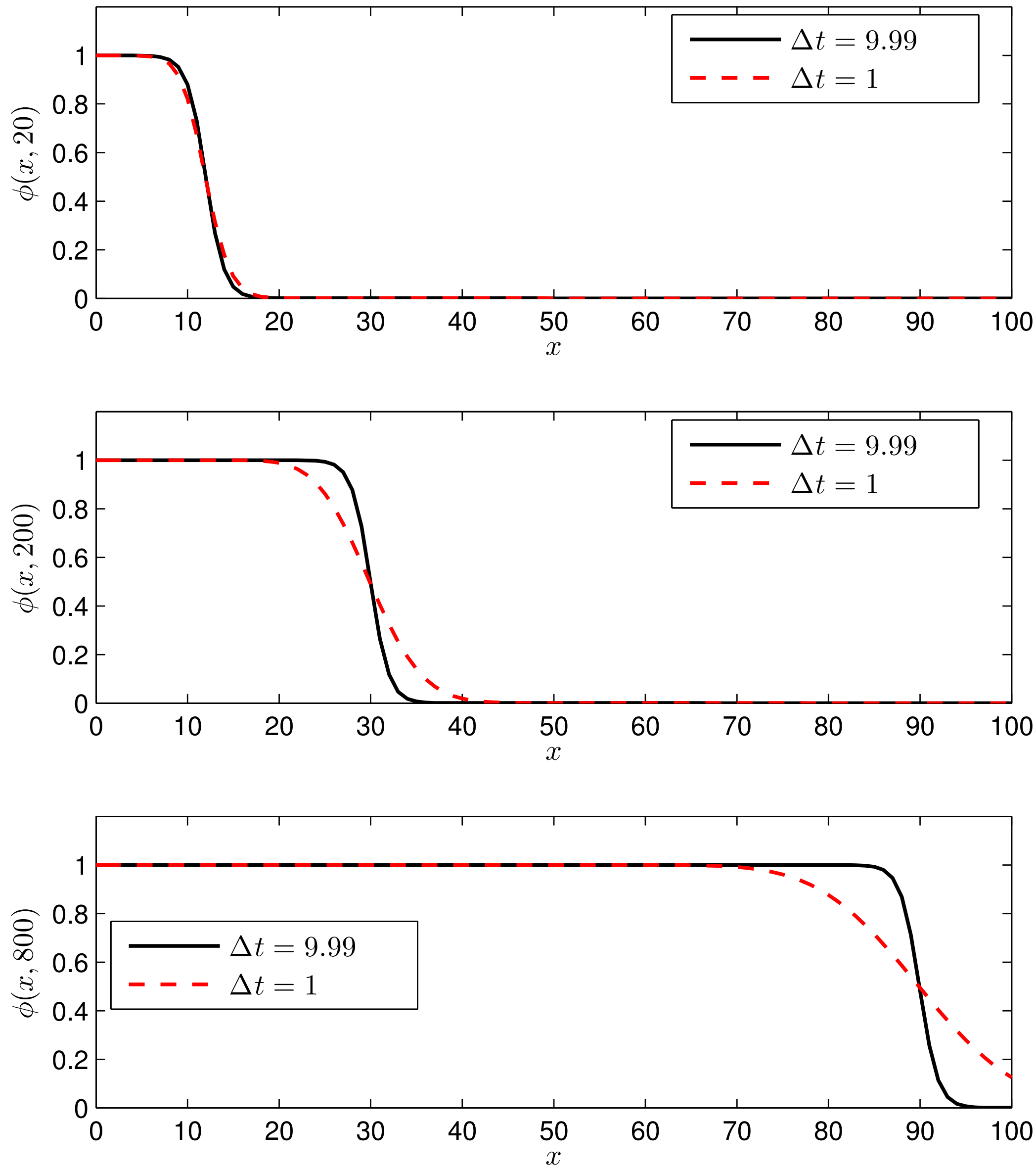

Figure 6. First-order upwind comparison between solution with critical time step and a time step a factor of 10 less.

Examining Figure 6, it is evident that the first-order upwind method is a dissipative scheme.

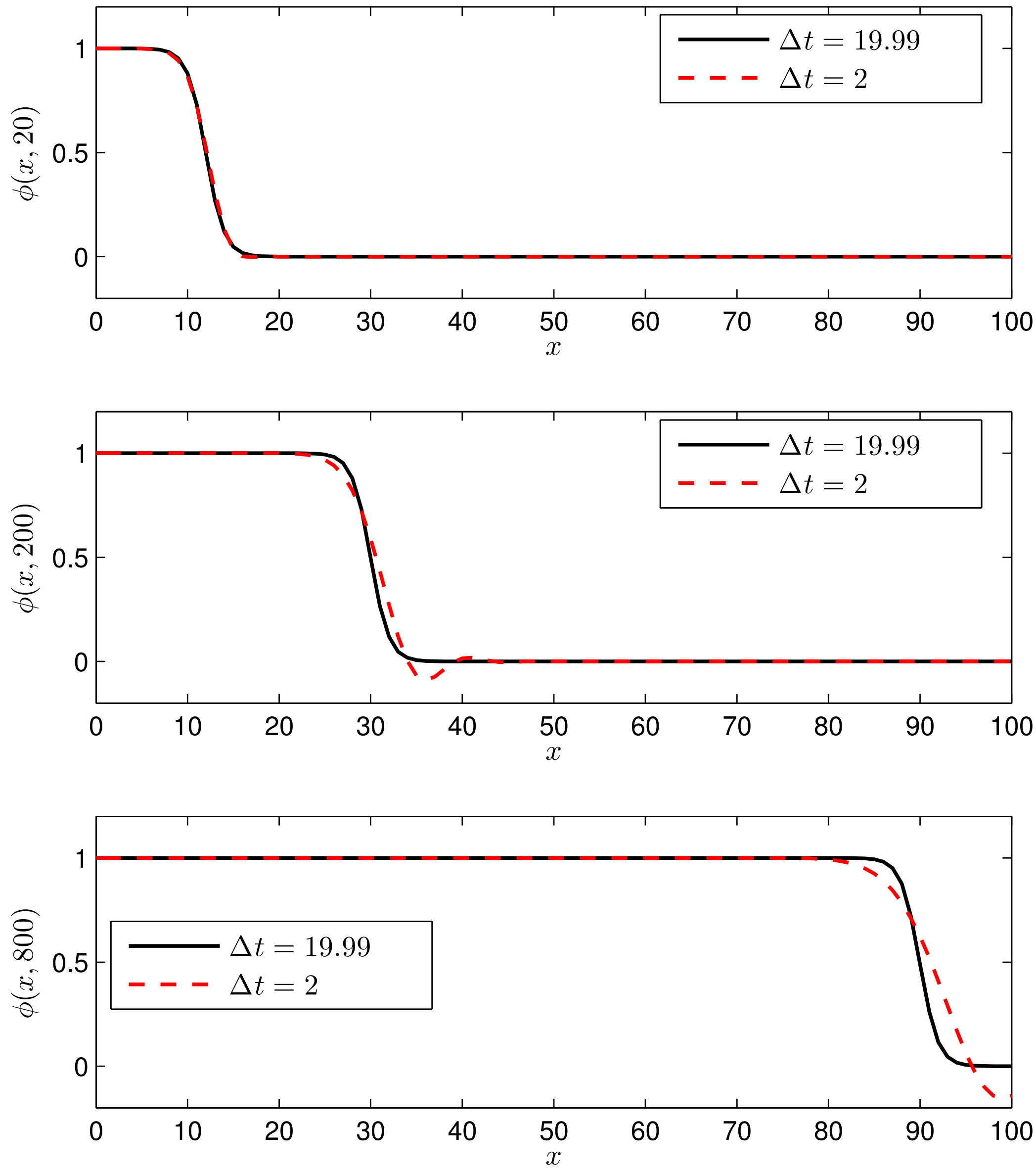

Figure 7. MacCormack method variant comparison between solution with critical time step and a time step a factor of 10 less.

Examing Figure 7, it is evident that the MacCormack method variant is a dispersive scheme.

Appendices

MATLAB code

Plotting Amplification Factors

```matlab
%==========================================================================
%% Clearing workspace
%==========================================================================

clc,clear,close all

%==========================================================================
%% Inputs
%==========================================================================

% Path for images
ImgPath = '../Images/';

% Font size
fSize = 10;

% Setting up beta vector and CFL number
beta = linspace(0,pi,100);
r_a = [0.1 0.75 1 1.25];
r_b = [0.1 0.2 0.3 0.4];
r_c = [0.5 1.5 2.0 2.1];

% Preallocating
G_a = zeros(numel(r_a),numel(beta));
G_b = zeros(numel(r_b),numel(beta));
G_c = zeros(numel(r_c),numel(beta));

% Filling in values
for n = 1:numel(r_a)
    G_a(n,:) = abs(1 - r_a(n)*(1 - cos(beta) + 1i*sin(beta)));
end

for n = 1:numel(r_b)
    G_b(n,:) = abs(1 - r_b(n)/2*(3 - 4*exp(-1i*beta) + exp(-2*1i*beta)));
end

for n = 1:numel(r_c)
    G_c(n,:) = abs(1 - r_c(n)*(1 - cos(beta) + 1i*sin(beta)) + ...
        1/2*r_c(n)*(r_c(n) - 1)*(1 - 2*(cos(beta) - 1i*sin(beta)) + ...
        cos(2*beta) - 1i*sin(2*beta)));
end

%==========================================================================
%% Plotting the amplification factor
%==========================================================================

% Setting up line style
lStyle = {'-k','--k','-.k',':k','ok','sk'};
```

```

% Setting up figure
figure('Position',[100 40 600 630])

% Preallocating
ph1 = zeros(size(r_a));
ph2 = zeros(size(r_b));
ph3 = zeros(size(r_c));
legString_a = numel(ph1)*[];
legString_b = numel(ph2)*[];
legString_c = numel(ph3)*[];

% Iteratively plotting different CFL numbers
subplot(3,1,1)
hold on
for n = 1:numel(r_a)

    ph1(n) = plot(beta,G_a(n,:),lStyle{n},'LineWidth',1.4);
    legString_a{n} = ['$r$ = ',sprintf('%1.1f',r_a(n))];

end

subplot(3,1,2)
hold on
for n = 1:numel(r_b)

    ph2(n) = plot(beta,G_b(n,:),lStyle{n},'LineWidth',1.4);
    legString_b{n} = ['$r$ = ',sprintf('%1.1f',r_b(n))];

end

subplot(3,1,3)
hold on
for n = 1:numel(r_c)

    ph3(n) = plot(beta,G_c(n,:),lStyle{n},'LineWidth',1.4);
    legString_c{n} = ['$r$ = ',sprintf('%1.1f',r_c(n))];

end

% Adding legend and labels/fixing some aesthetic issues for first plot
subplot(3,1,1)
set(gca,'Box','on','XLim',[0 pi],'XTick',linspace(0,pi,4), ...
    'XMinorTick','on','YMinorTick','on')
xlabel('$\beta$ [rad]','Interpreter','LaTeX','FontSize',fSize)
ylabel('$|G(\beta)|$','Interpreter','LaTeX','FontSize',fSize)
title('First-order upwind','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend(ph1,legString_a);
set(lh1,'Location','EastOutside')
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)

% Adding legend and labels/fixing some aesthetic issues for second plot
subplot(3,1,2)
set(gca,'Box','on','XLim',[0 pi],'XTick',linspace(0,pi,4), ...
    'XMinorTick','on','YMinorTick','on')
xlabel('$\beta$ [rad]','Interpreter','LaTeX','FontSize',fSize)
```

```
ylabel('$|G(\beta)|$','Interpreter','LaTeX','FontSize',fSize)
title('Second-order upwind','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend(ph2,legString_b);
set(lh1,'Location','EastOutside')
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)

% Adding legend and labels/fixing some aesthetic issues for third plot
subplot(3,1,3)
set(gca,'Box','on','XLim',[0 pi],'XTick',linspace(0,pi,4), ... Hw3_SOUcfl
    'XMinorTick','on','YMinorTick','on')
xlabel('$\beta$ [rad]','Interpreter','LaTeX','FontSize',fSize)
ylabel('$|G(\beta)|$','Interpreter','LaTeX','FontSize',fSize)
title('MacCormack method variant','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend(ph3,legString_c);
set(lh1,'Location','EastOutside')
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'Hw3_ampFact.eps'])

%===========================================================================
%% Plotting the CFL number vs beta for the second-order upwind method
%===========================================================================

r = 2*(cos(beta) - 1)./(3*cos(beta) - 5);

figure('Position',[750 40 480 215])
plot(beta,r,'-b','LineWidth',1.3)

% Adding labels
xlabel('$\beta$','Interpreter','LaTeX','FontSize',fSize)
ylabel('$r(\beta)$','Interpreter','LaTeX','FontSize',fSize)

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'Hw3_SOUcfl.eps'])
```

Solution script

```
%===========================================================================
%% Clearing workspace
%===========================================================================

clc,clear,close all

%===========================================================================
%% Inputs
%===========================================================================
```

```
% Font size
fSize = 11;

% Image path
ImgPath = '../Images/';

% Time step and spatial discretization
dt = [9.99 10.1 10.3 10.6 19.9 20.15 20.4 20.5 9.99 19.99 1 2];
dx = 1;

% Characteristic speed
c = 0.1;

% Start and stop points
tStart = 0;
tStop = 800;
xStart = 0;
xStop = 100;

% Structure for solution
soln = struct('x',[],'dx',[],'t',[],'dt',[],'phi_f',[],'phi_m',[]);

for I = 1:numel(dt)

    soln(I).dx = dx;
    soln(I).dt = dt(I);
    soln(I).x = xStart:dx:xStop;
    soln(I).t = tStart:dt(I):tStop;

    if soln(I).t(end) < tStop
        soln(I).t(end+1) = soln(I).t(end) + soln(I).dt;
    end

end

%========================================================================
%% Solution
%========================================================================


for K = 1:numel(dt)

    % Preallocating
    soln(K).phi_f = zeros(numel(soln(K).t),numel(soln(K).x));
    soln(K).phi_m = zeros(numel(soln(K).t),numel(soln(K).x));

    % Boundary condition
    for n = 1:numel(soln(K).t);
        soln(K).phi_f(n,1) = 1;
        soln(K).phi_m(n,1) = 1;
    end

    % Initial condition
    for I = 1:numel(soln(K).x)
        soln(K).phi_f(1,I) = 1/(1 + exp(soln(K).x(I) - 10));
        soln(K).phi_m(1,I) = 1/(1 + exp(soln(K).x(I) - 10));
    end

```

```
    % First order upwind
    for n = 1:(numel(soln(K).t) - 1)
        for I = 2:numel(soln(K).x)

            soln(K).phi_f(n+1,I) = soln(K).phi_f(n,I) - ...
                c*soln(K).dt/dx*(soln(K).phi_f(n,I) - ...
                soln(K).phi_f(n,I-1));

        end
    end

    % Second order upwind
    v = c*soln(K).dt/dx;
    for n = 1:(numel(soln(K).t)-1)
        for I = 2:numel(soln(K).x)

            if I == 2

                soln(K).phi_m(n+1,I) = soln(K).phi_m(n,I) - v*( ...
                    soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1)) + 1/2*v ...
                    *(v-1)*(soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1));

            else

                soln(K).phi_m(n+1,I) = soln(K).phi_m(n,I) - v*( ...
                    soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1)) + 1/2*v* ...
                    (v-1)*(soln(K).phi_m(n,I) - 2*soln(K).phi_m(n,I-1) ...
                    + soln(K).phi_m(n,I-2));

            end

        end
    end

end

%===========================================================================
%% Determining critical time step
%===========================================================================

%---------------------------------------------------------------------------
% First-order upwind
%---------------------------------------------------------------------------

% Data records to compare
dSets = [1:4];

% Setting up figure
figure('Position',[40 100 660 500],'renderer','zbuffer')

% Iteratively plotting solution contours
for J = 1:numel(dSets)

    % Plotting
    subplot(2,2,J)
    [X,T] = meshgrid(soln(dSets(J)).x,soln(dSets(J)).t);
    contour(X,T,soln(dSets(J)).phi_f)
    pcolor(X,T,soln(dSets(J)).phi_f)
    shading interp
```

```

    % Labels
    str = sprintf('%1.2f',soln(dSets(J)).dt);
    title(['Time step: ',str],'Interpreter','LaTeX','FontSize',fSize)
    xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
    ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)

end

% Saving plot
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'fou_tcrit.eps'])

%---------------------------------------------------------------------------
% MacCormack method variant
%---------------------------------------------------------------------------

% Data records to compare
dSets = [5:8];

% Setting up figure
figure('Position',[40 100 660 500],'renderer','zbuffer')

% Iteratively plotting solution contours
for J = 1:numel(dSets)

    % Plotting
    subplot(2,2,J)
    [X,T] = meshgrid(soln(dSets(J)).x,soln(dSets(J)).t);
    contour(X,T,soln(dSets(J)).phi_m)
    pcolor(X,T,soln(dSets(J)).phi_m)
    shading interp

    % Labels
    str = sprintf('%1.2f',soln(dSets(J)).dt);
    title(['Time step: ',str],'Interpreter','LaTeX','FontSize',fSize)
    xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
    ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)

end

% Saving plot
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'mmv_tcrit.eps'])

%===========================================================================
%% Plotting solution at specific times
%===========================================================================

% Find the indices at which t = 20, 200, 800 for first-order upwind
fouSoln = 9;
mmvSoln = 10;
tol = 0.5;
x1 = soln(fouSoln).x;
x2 = soln(mmvSoln).x;
n1_a = find(soln(fouSoln).t<=20+tol);
n1_b = find(soln(fouSoln).t>=20-tol);
n1_20 = intersect(n1_a,n1_b);
n1_a = find(soln(fouSoln).t<=200+tol);
```

```
n1_b = find(soln(fouSoln).t>=200-tol);
n1_200 = intersect(n1_a,n1_b);
n1_a = find(soln(fouSoln).t<=800+tol);
n1_b = find(soln(fouSoln).t>=800-tol);
n1_800 = intersect(n1_a,n1_b);
n1_800 = numel(soln(fouSoln).t)-1;

% Find the indices at which t = 20, 200, 800 for MacCormack variant
n2_a = find(soln(mmvSoln).t<=20+tol);
n2_b = find(soln(mmvSoln).t>=20-tol);
n2_20 = intersect(n2_a,n2_b);
n2_a = find(soln(mmvSoln).t<=200+tol);
n2_b = find(soln(mmvSoln).t>=200-tol);
n2_200 = intersect(n2_a,n2_b);
n2_a = find(soln(mmvSoln).t<=800+tol);
n2_b = find(soln(mmvSoln).t>=800-tol);
n2_800 = intersect(n2_a,n2_b);
n2_800 = numel(soln(mmvSoln).t)-1;

% Extracting solutions at desired time steps
phi_f_20 = soln(fouSoln).phi_f(n1_20(1),:);
phi_f_200 = soln(fouSoln).phi_f(n1_200(1),:);
phi_f_800 = soln(fouSoln).phi_f(n1_800(1),:);

phi_m_20 = soln(mmvSoln).phi_m(n2_20(1),:);
phi_m_200 = soln(mmvSoln).phi_m(n2_200(1),:);
phi_m_800 = soln(mmvSoln).phi_m(n2_800(1),:);


% Setting up figure
figure('Position',[70 10 560 640])

%--------------------------------------------------------------------------
% t = 20
%--------------------------------------------------------------------------

% Plotting
subplot(3,1,1)
hold on
plot(x1,phi_f_20,'-k','LineWidth',1.4)
plot(x2,phi_m_20,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
ylabel('$\phi(x,20)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('First-order upwind','MacCormack variant');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

%--------------------------------------------------------------------------
% t = 200
%--------------------------------------------------------------------------

% Plotting
```

```
subplot(3,1,2)
hold on
plot(x1,phi_f_200,'-k','LineWidth',1.4)
plot(x2,phi_m_200,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,200)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('First-order upwind','MacCormack variant');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

%-------------------------------------------------------------------------
% t = 800
%-------------------------------------------------------------------------

% Plotting
subplot(3,1,3)
hold on
plot(x1,phi_f_800,'-k','LineWidth',1.4)
plot(x2,phi_m_800,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,800)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('First-order upwind','MacCormack variant');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
set(lh1,'Location','West')
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'Hw3PartE.eps'])

%-------------------------------------------------------------------------
% Comparing with different time steps
%-------------------------------------------------------------------------

% Records for solns with time step decreased by 10
a = 11;
b = 12;
x1p = soln(a).x;
x2p = soln(b).x;

% Finding indices
n1_20 = find(soln(a).t==20);
n1_200 = find(soln(a).t==200);
n1_800 = find(soln(a).t==800);
n2_20 = find(soln(b).t==20);
```

```
n2_200 = find(soln(b).t==200);
n2_800 = find(soln(b).t==800);

% Separating solutions
phi_fp_20 = soln(a).phi_f(n1_20,:);
phi_fp_200 = soln(a).phi_f(n1_200,:);
phi_fp_800 = soln(a).phi_f(n1_800,:);
phi_mp_20 = soln(b).phi_m(n2_20,:);
phi_mp_200 = soln(b).phi_m(n2_200,:);
phi_mp_800 = soln(b).phi_m(n2_800,:);

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% First-order upwind comparison
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% Setting up figure
figure('Position',[70 10 560 640])

% Plotting
subplot(3,1,1)
hold on
plot(x1,phi_f_20,'-k','LineWidth',1.4)
plot(x1p,phi_fp_20,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,20)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 9.99$','$\Delta t = 1$');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

% Plotting
subplot(3,1,2)
hold on
plot(x1,phi_f_200,'-k','LineWidth',1.4)
plot(x1p,phi_fp_200,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,200)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 9.99$','$\Delta t = 1$');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

% Plotting
subplot(3,1,3)
hold on
```

```
plot(x1,phi_f_800,'-k','LineWidth',1.4)
plot(x1p,phi_fp_800,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[0 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,800)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 9.99$','$\Delta t = 1$');
set(lh1,'Location','West')
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'Hw3PartE1.eps'])

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
% MacCormack method variant comparison
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% Setting up figure
figure('Position',[70 10 560 640])

% Plotting
subplot(3,1,1)
hold on
plot(x2,phi_m_20,'-k','LineWidth',1.4)
plot(x2p,phi_mp_20,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[-0.2 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,20)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 19.99$','$\Delta t = 2$');
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

% Plotting
subplot(3,1,2)
hold on
plot(x2,phi_m_200,'-k','LineWidth',1.4)
plot(x2p,phi_mp_200,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[-0.2 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,200)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 19.99$','$\Delta t = 2$');
```

```matlab
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])

% Plotting
subplot(3,1,3)
hold on
plot(x2,phi_m_800,'-k','LineWidth',1.4)
plot(x2p,phi_mp_800,'--r','LineWidth',1.4)

% Adding labels and improving aesthetics
set(gca,'YLim',[-0.2 1.2],'Box','on')
xlabel('$x$','Interpreter','LaTeX','FontSize',fSize);
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
ylabel('$\phi(x,800)$','Interpreter','LaTeX','FontSize',fSize)
[lh1,lh2] = legend('$\Delta t = 19.99$','$\Delta t = 2$');
set(lh1,'Location','West')
txtobj = findall(lh1,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
txtobj = findall(lh2,'type','text');
set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
lPos = get(lh1,'Position');
set(lh1,'Position',[lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'Hw3PartE2.eps'])
```

Post-processing

```matlab
%==========================================================================
% Clearing workspace
%==========================================================================

clc,clear,close all

%==========================================================================
% Inputs
%==========================================================================

% Path for images
ImgPath = '../Images/';

% Font size
fSize = 11;

% Path to data file
MainPath = '../C++Files/DataFiles/';

%==========================================================================
% Determining critical time step for first-order upwind method
%==========================================================================
```

```
% Paths to first-order upwind data
fouData{1} = 'Hw3_tStep4.9_xStep0.5.dat';
fouData{2} = 'Hw3_tStep5.1_xStep0.5.dat';
fouData{3} = 'Hw3_tStep5.2_xStep0.5.dat';
fouData{4} = 'Hw3_tStep5.35_xStep0.5.dat';

% Vector of time steps
tStep = [4.9 5.1 5.2 5.35];

% Setting up figure
figure('Position',[50 40 680 450])

for I = 1:numel(tStep)

    % Importing data
    DataPath = [MainPath,fouData{I}];
    data = csvread(DataPath);

    % Separating data
    x = data(:,1);
    n = data(:,2);
    phi_f = data(:,3);

    % Gridding the data
    [X,N,Phi] = griddata(x,n,phi_f,unique(x),unique(n)');

    % Creating contour plot
    subplot(2,2,I)
    [c,h] = contour(X,N,Phi);

    % Adding labels
    str = sprintf('%1.2f',tStep(I));
    title(['Time step: ',str]);
    xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
    ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)

end

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'fou_tcrit.eps'])

%==========================================================================
% Determining critical time step for MacCormack method variant
%==========================================================================

% Paths to first-order upwind data
mmvData{1} = 'Hw3_tStep4.9_xStep0.5.dat';
mmvData{2} = 'Hw3_tStep5.1_xStep0.5.dat';
mmvData{3} = 'Hw3_tStep5.2_xStep0.5.dat';
mmvData{4} = 'Hw3_tStep10_xStep0.5.dat';

% Vector of time steps
tStep = [4.9 5.1 5.2 8];

% Setting up figure
figure('Position',[50 40 680 450])

```

```
for I = 1:numel(tStep)

    % Importing data
    DataPath = [MainPath,mmvData{I}];
    data = csvread(DataPath);

    % Separating data
    x = data(:,1);
    n = data(:,2);
    phi_m = data(:,4);

    % Gridding the data
    [X,N,Phi] = griddata(x,n,phi_m,unique(x),unique(n)');

    % Creating contour plot
    subplot(2,2,I)
    [c,h] = contour(X,N,Phi);

    % Adding labels
    str = sprintf('%1.2f',tStep(I));
    title(['Time step: ',str]);
    xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
    ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)

end

% Saving figure
set(gcf,'PaperPositionMode','Auto')
print(gcf,'-depsc',[ImgPath,'mmv_tcrit.eps'])
```

C++ code

```
#include "cptstd.hpp"
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <cmath>
#include "matrix.hpp"
#include <iomanip>

using namespace cpt;

/*-----------------------------------------------------------------------------
   This program must be compiled using g++ -o Hw3 -std=c++0x Hw3.cpp because
   of a num to string conversion that requires C++ 11 capabilities.
----------------------------------------------------------------------------*/

int main() {

  //=============================================================================
  // Declaring variables and generating grid
  //=============================================================================

```

```
cout << "\n=====================================================" << endl;
cout << "|               AE 5327 -- Homework 3                |" << endl;
cout << "=====================================================\n" << endl;

// Asking the user for time step
double dt,dx;
cout << "Enter dt: ";
cin >> dt;
cout << "Enter dx: ";
cin >> dx;

// Setting up the time vector
double tStart = 0.0;
double tStop = 800.0;
double tPoints;
tPoints = ceil((tStop - tStart)/dt);
vector<double> t;
t.reserve((int) tPoints);

// Setting up x vector
double xStart = 0.0;
double xStop = 100.0;
double xPoints;
xPoints = ceil((xStop - xStart)/dx);
vector<double> x;
x.reserve((int) xPoints);

// Generating the grid
for (int i=0; i<=((int) tPoints); i++){

  if (i==0){
  t[i] = tStart;
  }
  else {
  t[i] = t[i-1] + dt;
  }

}
for (int j=0; j<=((int) xPoints); j++){

  if (j==0) {
  x[j] = xStart;
  }
  else {
  x[j] = x[j-1] + dx;
  }

}

double c = 0.1;
double v;
int nRow = (int) tPoints;
int nCol = (int) xPoints;

Matrix<double,2> phi_f(nRow,nCol);
Matrix<double,2> phi_s(nRow,nCol);
Matrix<double,2> phi_m(nRow,nCol);
phi_f = Matrix<double,2>(nRow,nCol);
phi_s = Matrix<double,2>(nRow,nCol);
```

```
  phi_m = Matrix<double,2>(nRow,nCol);
  cout << "\n" << nRow << " by " << nCol << " grid with " << phi_f.size()
     << " cells." << endl;

  //=========================================================================
  // First-order upwind solution
  //=========================================================================

  // Enforcing boundary condition
  for (int n=0; n<nRow; n++) {
    phi_f[n][0] = 1.0;
  }

  // Enforcing initial condition
  for (int i=0; i<nCol; i++) {
    phi_f[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
  }

  for (int n=0; n<(nRow-1); n++) {

    for (int i=1; i<nCol; i++) {

    phi_f[n+1][i] = phi_f[n][i] - c*dt/dx*(phi_f[n][i] - phi_f[n][i-1]);

    }

  }

  //=========================================================================
  // Second-order upwind solution
  //=========================================================================

  // Enforcing boundary condition
  for (int n=0; n<nRow; n++) {
    phi_s[n][0] = 1.0;
  }

  // Enforcing initial condition
  for (int i=0; i<nCol; i++) {
    phi_s[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
  }

  for (int n=0; n<(nRow-1); n++) {

    for (int i=1; i<nCol; i++) {

    if (i==1) {

      phi_s[n+1][i] = phi_s[n][i] - c*dt/(2.0*dx)*(3.0*phi_s[n][i] -
         3.0*phi_s[n][i-1]);

    }
    else {

      phi_s[n+1][i] = phi_s[n][i] - c*dt/(2.0*dx)*(3.0*phi_s[n][i] -
         4.0*phi_s[n][i-1] + phi_s[n][i-2]);

    }

```

```
    }

  }

  //=========================================================================
  // MacCormack method variant
  //=========================================================================

  // Enforcing boundary condition
  for (int n=0; n<nRow; n++) {
    phi_m[n][0] = 1.0;
  }

  // Enforcing initial condition
  for (int i=0; i<nCol; i++) {
    phi_m[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
  }

  // CFL number
  v = c*dt/dx;

  for (int n=0; n<(nRow-1); n++) {

    for (int i=1; i<nCol; i++) {

    if (i==1) {

      phi_m[n+1][i] = phi_m[n][i] - v*(phi_m[n][i] - phi_m[n][i-1]) +
        1/2*v*(v-1)*(phi_m[n][i] - phi_m[n][i-1]);

    }
    else {

      phi_m[n+1][i] = phi_m[n][i] - v*(phi_m[n][i] - phi_m[n][i-1]) +
        1/2*v*(v-1)*(phi_m[n][i] - 2.0*phi_m[n][i-1] + phi_m[n][i-2]);

    }

    }

  }

  //=========================================================================
  // Writing data to file
  //=========================================================================

  // Constructing file name
  stringstream str1;
  stringstream str2;
  str1 << setprecision(3) << dt;
  str2 << setprecision(3) << dx;
  string tStepStr = str1.str();
  string xStepStr = str2.str();
  string fName = "Hw3_tStep" + tStepStr + "_xStep" + xStepStr + ".dat";
  string fullfName = "./DataFiles/" + fName;

  // Outputting some info
  cout << "\nWriting data file...\n\n";
  cout << "Solution in file " << fName << endl;
```

```

  // Opening stream and writing data file
  ofstream data_file;
  data_file.open(fullfName);
  double prog = 0;
  for (int i=0; i<nCol; i++) {
    for (int j=0; j<nRow; j++) {

    data_file << setprecision(10) << x[i] << ", " << t[j] << ", " << phi_f[j][i]
            << ", " << phi_m[j][i] << endl;

    }
    data_file << "\n";

  }

  // Closing data file
  data_file.close();

  return 0;
}
```