

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:

$$\begin{aligned}\phi(0, t) &= 1 \\ \phi(x, 0) &= \frac{1}{1 + e^{x-10}} \\ 0 &\leq x \leq 100\end{aligned}$$

- 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

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + c \frac{\phi_i^n - \phi_{i-1}^n}{\Delta x} = 0 \quad (1)$$

and the forward Euler, second-order upwind differencing yields

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

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.

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

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

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

Also, an assumption about the solution is made.

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

Therefore,

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

Inserting (6) into (3) and simplifying,

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

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

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

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

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

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

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

Inserting (6) into (10) and rearranging,

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

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

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

Using Euler's identity, (12) becomes

$$G(\beta) = 1 - \frac{r}{2} (3 - 4(\cos \beta - i \sin \beta) + \cos(2\beta) - i \sin(2\beta)) \quad (13)$$

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

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

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

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

Part B

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

$$|1 - r(1 - \cos \beta + i \sin \beta)| \leq 1 \quad (16)$$

Solving this inequality yields

$$0 \leq r \leq 1 \quad (17)$$

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

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

Solving the inequality yields

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

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.

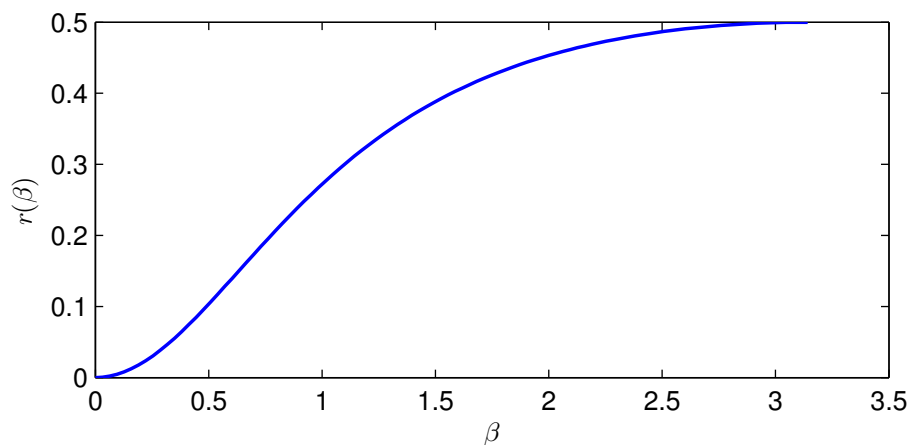


Figure 1. CFL number vs β for the second-order upwind method.

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

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

Solving this inequality yields

$$\boxed{0 \leq r \leq 2} \quad (21)$$

The amplification factors are plotted in Figure 2.

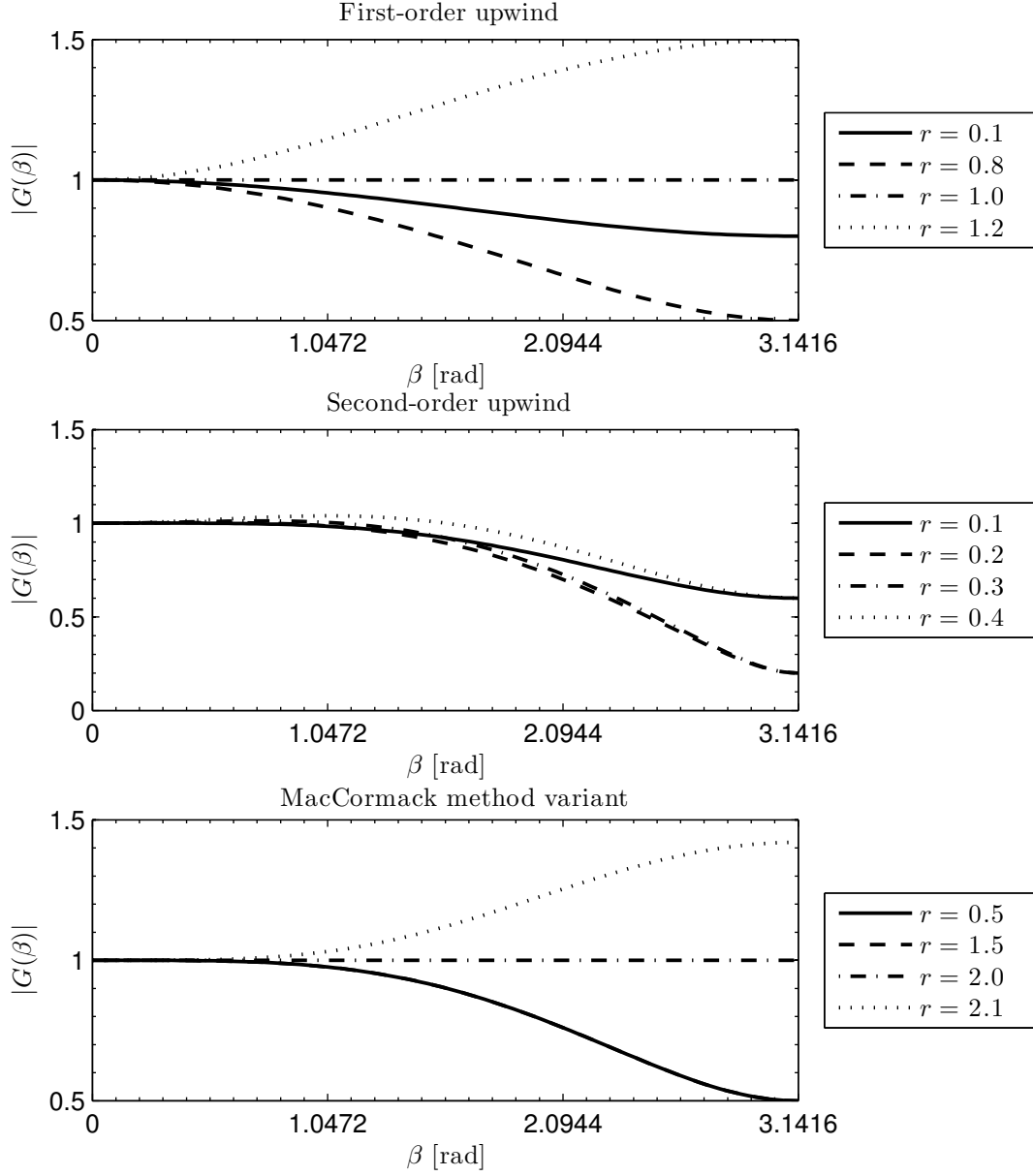


Figure 2. Amplification factors.

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.

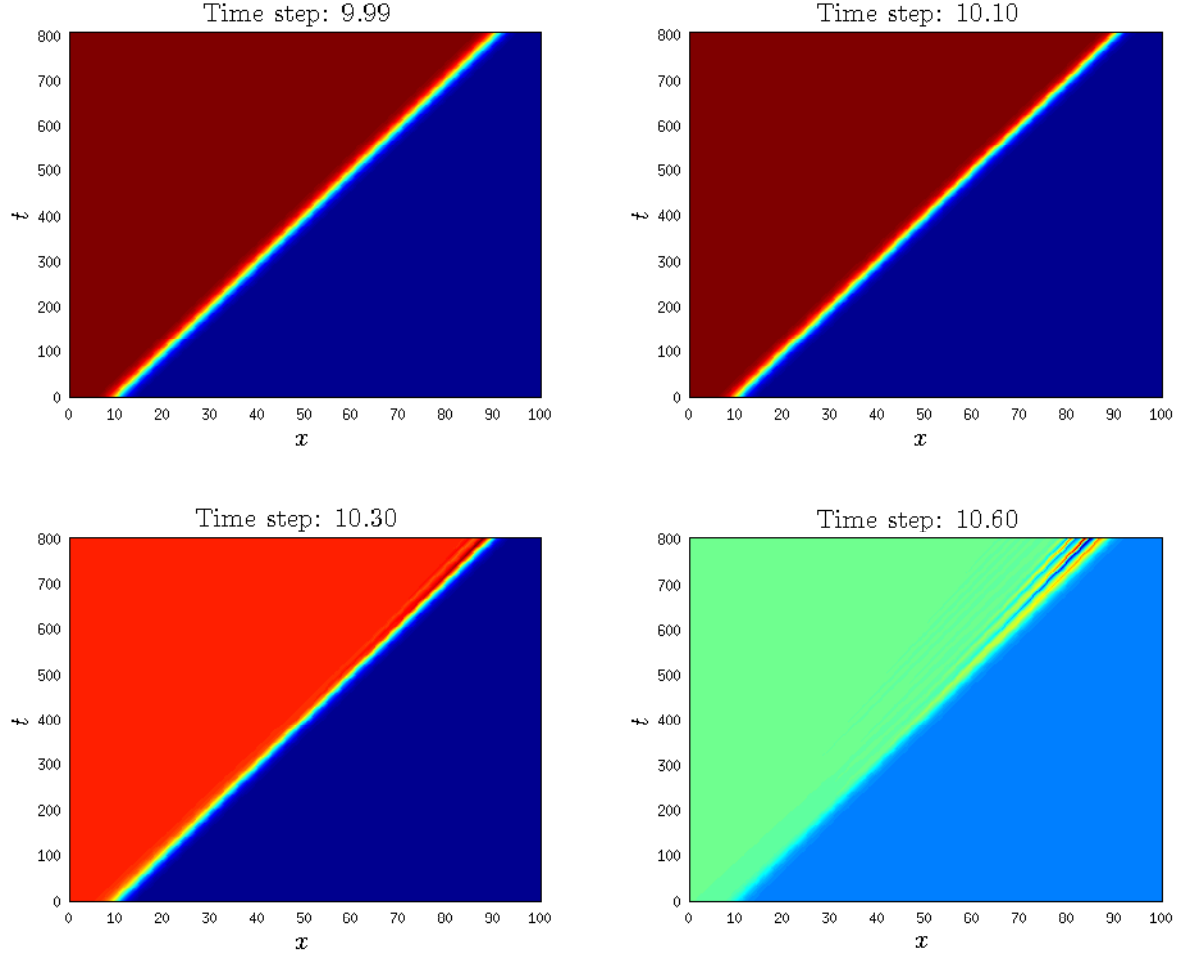


Figure 3. Contour plot of solutions with different time steps.

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.

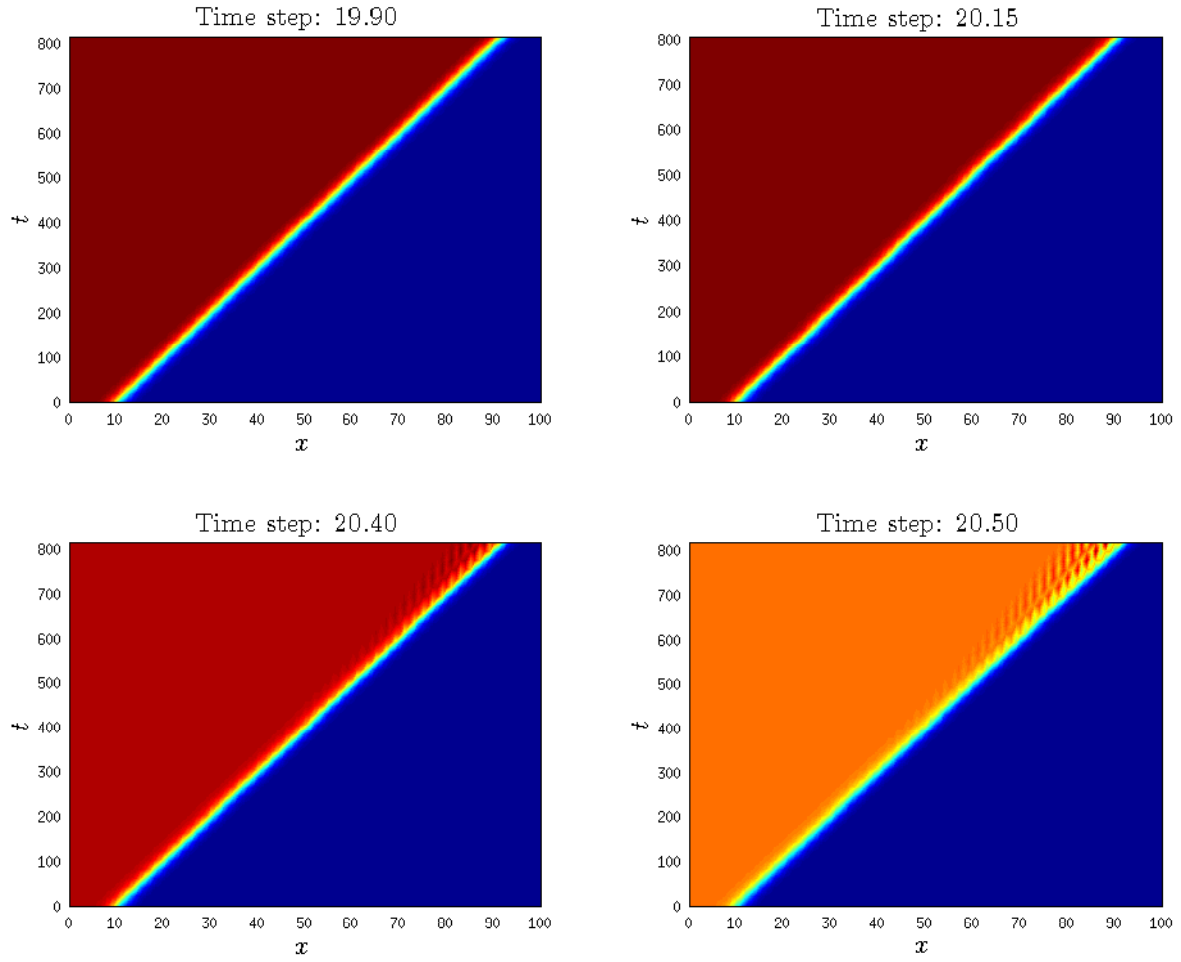


Figure 4. Contour plot of solutions with different time steps.

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.

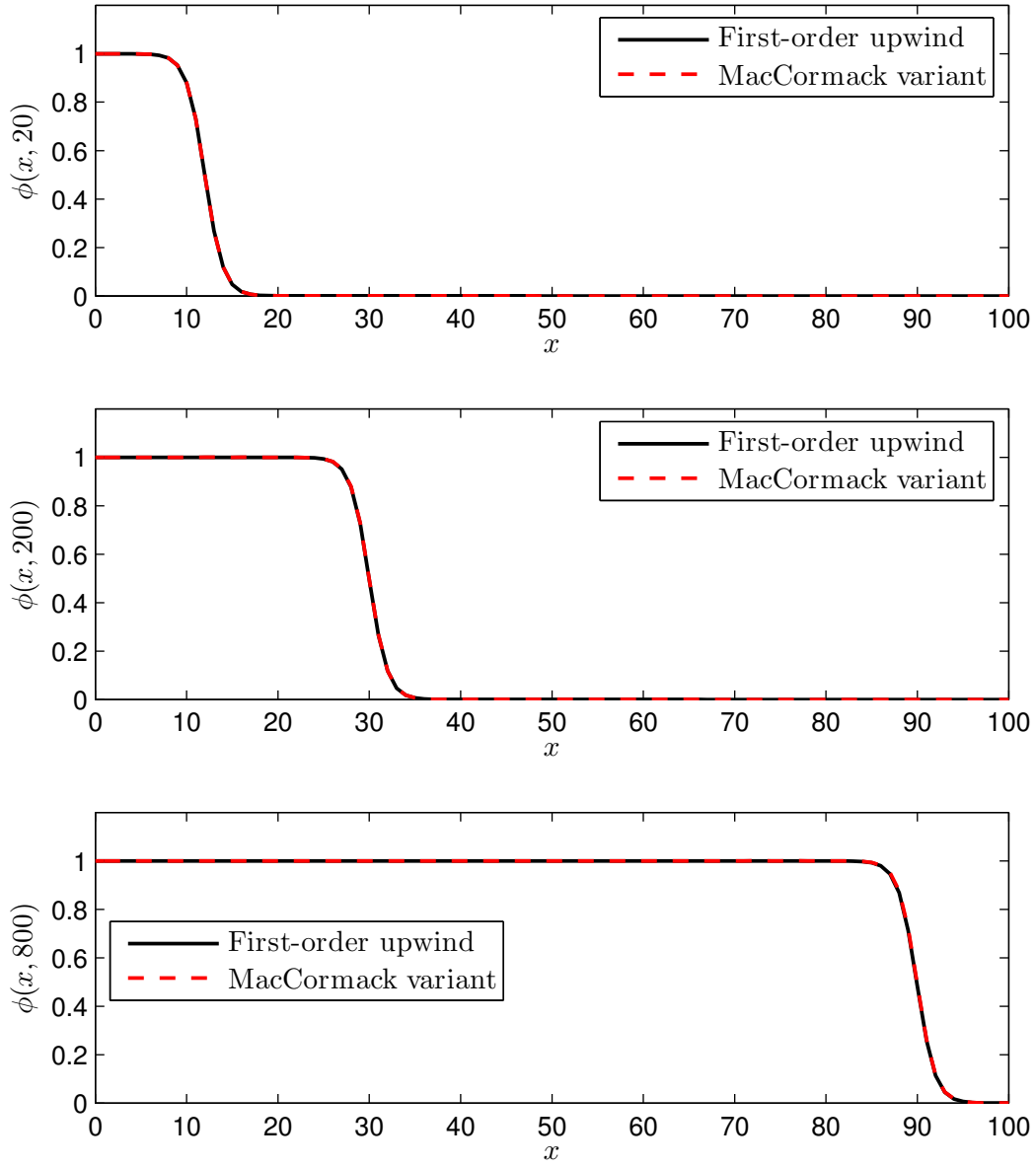


Figure 5. Comparison between solution at selected times.

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

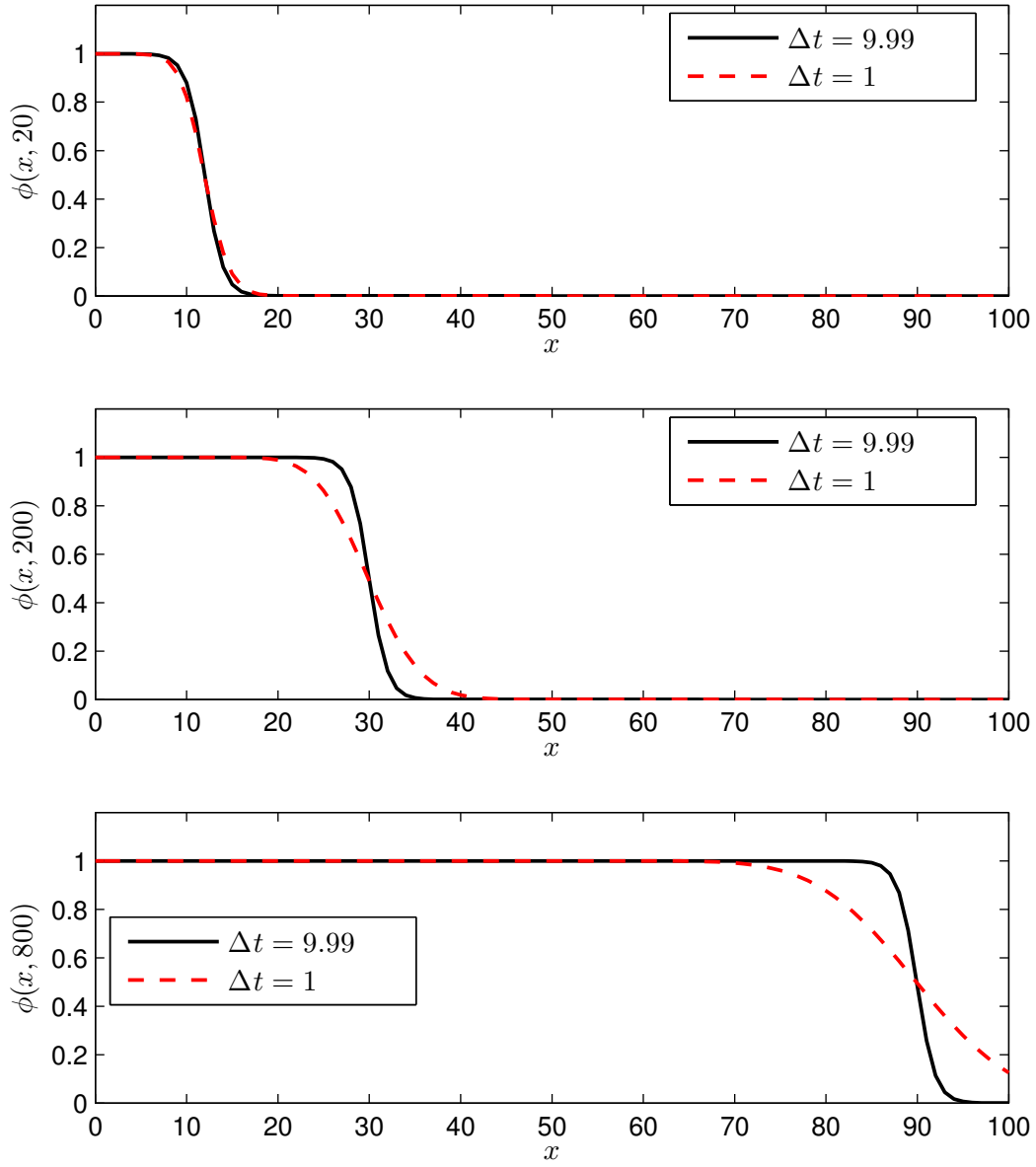


Figure 6. First-order upwind comparison between solution with critical time step and a time step a factor of 10 less.

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

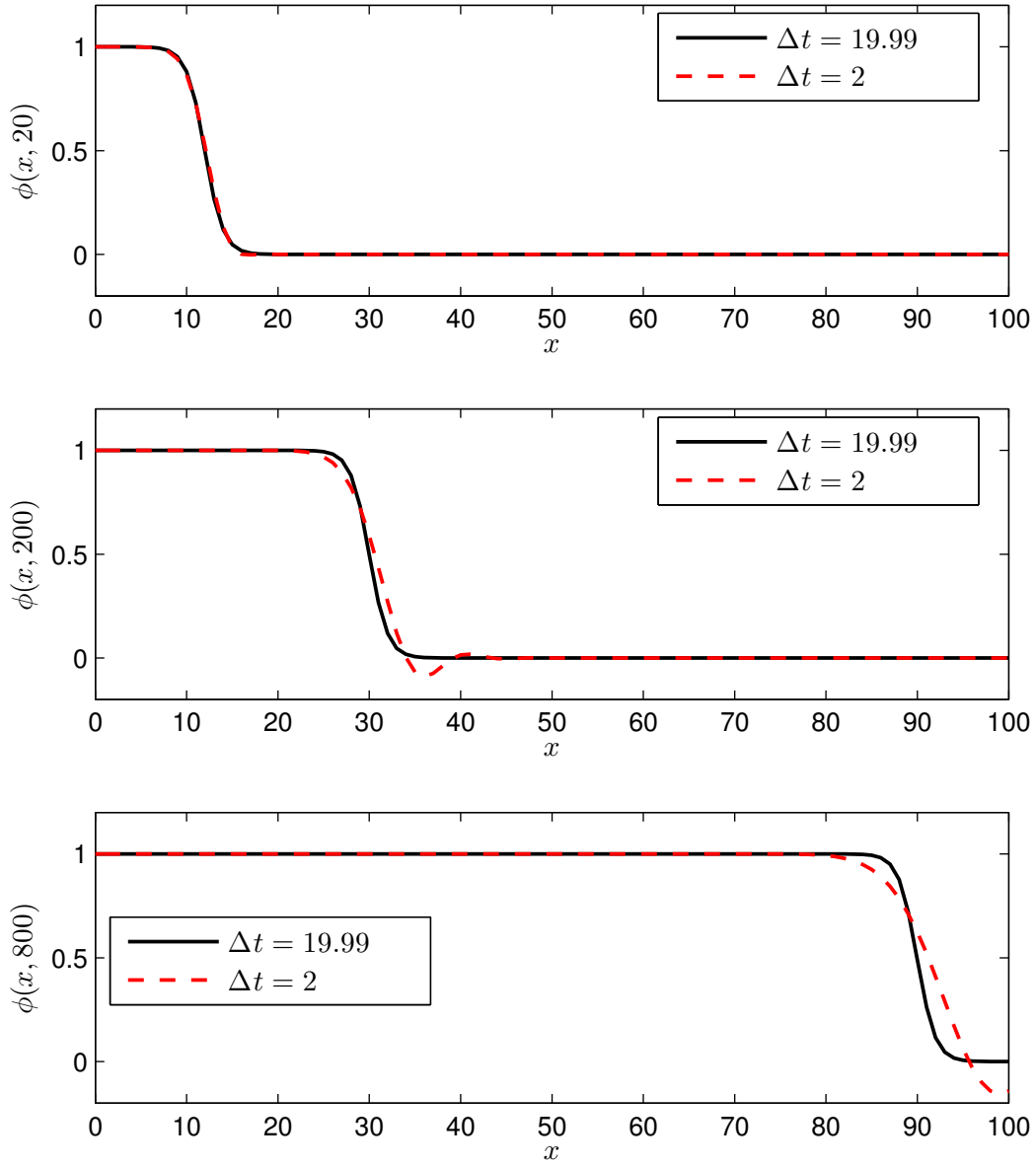


Figure 7. MacCormack method variant comparison between solution with critical time step and a time step a factor of 10 less.

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

Appendices

MATLAB code

Plotting Amplification Factors

```
1  %=====
2  %% Clearing workspace
3  %=====
4
5  clc,clear,close all
6
7  %=====
8  %% Inputs
9  %=====
10
11 % Path for images
12 ImgPath = '../Images/';
13
14 % Font size
15 fSize = 10;
16
17 % Setting up beta vector and CFL number
18 beta = linspace(0,pi,100);
19 r_a = [0.1 0.75 1 1.25];
20 r_b = [0.1 0.2 0.3 0.4];
21 r_c = [0.5 1.5 2.0 2.1];
22
23 % Preallocating
24 G_a = zeros(numel(r_a),numel(beta));
25 G_b = zeros(numel(r_b),numel(beta));
26 G_c = zeros(numel(r_c),numel(beta));
27
28 % Filling in values
29 for n = 1:numel(r_a)
30     G_a(n,:) = abs(1 - r_a(n)*(1 - cos(beta) + 1i*sin(beta)));
31 end
32
33 for n = 1:numel(r_b)
34     G_b(n,:) = abs(1 - r_b(n)/2*(3 - 4*exp(-1i*beta) + exp(-2*1i*beta)));
35 end
36
37 for n = 1:numel(r_c)
38     G_c(n,:) = abs(1 - r_c(n)*(1 - cos(beta) + 1i*sin(beta)) + ...
39         1/2*r_c(n)*(r_c(n) - 1)*(1 - 2*(cos(beta) - 1i*sin(beta)) + ...
40         cos(2*beta) - 1i*sin(2*beta)));
41 end
42
43 %=====
44 %% Plotting the amplification factor
45 %=====
46
47 % Setting up line style
48 lStyle = {'-k','--k','-.k',':k','ok','sk'};
```

```

49
50 % Setting up figure
51 figure('Position',[100 40 600 630])
52
53 % Preallocating
54 ph1 = zeros(size(r_a));
55 ph2 = zeros(size(r_b));
56 ph3 = zeros(size(r_c));
57 legString_a = numel(ph1)*[];
58 legString_b = numel(ph2)*[];
59 legString_c = numel(ph3)*[];
60
61 % Iteratively plotting different CFL numbers
62 subplot(3,1,1)
63 hold on
64 for n = 1:numel(r_a)
65
66     ph1(n) = plot(beta,G_a(n,:),lStyle{n},'LineWidth',1.4);
67     legString_a{n} = ['$r$ = ',sprintf('%1.1f',r_a(n))];
68
69 end
70
71 subplot(3,1,2)
72 hold on
73 for n = 1:numel(r_b)
74
75     ph2(n) = plot(beta,G_b(n,:),lStyle{n},'LineWidth',1.4);
76     legString_b{n} = ['$r$ = ',sprintf('%1.1f',r_b(n))];
77
78 end
79
80 subplot(3,1,3)
81 hold on
82 for n = 1:numel(r_c)
83
84     ph3(n) = plot(beta,G_c(n,:),lStyle{n},'LineWidth',1.4);
85     legString_c{n} = ['$r$ = ',sprintf('%1.1f',r_c(n))];
86
87 end
88
89 % Adding legend and labels/fixing some aesthetic issues for first plot
90 subplot(3,1,1)
91 set(gca,'Box','on','XLim',[0 pi],'XTick',linspace(0,pi,4), ...
92     'XMinorTick','on','YMinorTick','on')
93 xlabel('$\beta$ [rad]','Interpreter','LaTeX','FontSize',fSize)
94 ylabel('$G(\beta)$','Interpreter','LaTeX','FontSize',fSize)
95 title('First-order upwind','Interpreter','LaTeX','FontSize',fSize)
96 [lh1,lh2] = legend(ph1,legString_a);
97 set(lh1,'Location','EastOutside')
98 txtobj = findall(lh1,'type','text');
99 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
100 txtobj = findall(lh2,'type','text');
101 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
102
103 % Adding legend and labels/fixing some aesthetic issues for second plot
104 subplot(3,1,2)
105 set(gca,'Box','on','XLim',[0 pi],'XTick',linspace(0,pi,4), ...
106     'XMinorTick','on','YMinorTick','on')
107 xlabel('$\beta$ [rad]','Interpreter','LaTeX','FontSize',fSize)

```

```

108 ylabel('$|G(\beta)|$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
109 title('Second-order upwind', 'Interpreter', 'LaTeX', 'FontSize', fSize)
110 [lh1, lh2] = legend(ph2, legString_b);
111 set(lh1, 'Location', 'EastOutside')
112 txtobj = findall(lh1, 'type', 'text');
113 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
114 txtobj = findall(lh2, 'type', 'text');
115 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
116
117 % Adding legend and labels/fixing some aesthetic issues for third plot
118 subplot(3,1,3)
119 set(gca, 'Box', 'on', 'XLim', [0 pi], 'XTick', linspace(0, pi, 4), ... Hw3_SOUcfl
120 'XMinorTick', 'on', 'YMinorTick', 'on')
121 xlabel('$\beta$ [rad]', 'Interpreter', 'LaTeX', 'FontSize', fSize)
122 ylabel('$|G(\beta)|$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
123 title('MacCormack method variant', 'Interpreter', 'LaTeX', 'FontSize', fSize)
124 [lh1, lh2] = legend(ph3, legString_c);
125 set(lh1, 'Location', 'EastOutside')
126 txtobj = findall(lh1, 'type', 'text');
127 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
128 txtobj = findall(lh2, 'type', 'text');
129 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
130
131 % Saving figure
132 set(gcf, 'PaperPositionMode', 'Auto')
133 print(gcf, '-depsc', [ImagePath, 'Hw3_ampFact.eps'])
134
135 %=====
136 %% Plotting the CFL number vs beta for the second-order upwind method
137 %=====
138
139 r = 2*(cos(beta) - 1)/(3*cos(beta) - 5);
140
141 figure('Position', [750 40 480 215])
142 plot(beta, r, '-b', 'LineWidth', 1.3)
143
144 % Adding labels
145 xlabel('$\beta$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
146 ylabel('$r(\beta)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
147
148 % Saving figure
149 set(gcf, 'PaperPositionMode', 'Auto')
150 print(gcf, '-depsc', [ImagePath, 'Hw3_SOUcfl.eps'])

```

Solution script

```

1 %=====
2 %% Clearing workspace
3 %=====
4
5 clc, clear, close all
6
7 %=====
8 %% Inputs
9 %=====

```

```

10
11 % Font size
12 fSize = 11;
13
14 % Image path
15 ImgPath = '../Images/';
16
17 % Time step and spatial discretization
18 dt = [9.99 10.1 10.3 10.6 19.9 20.15 20.4 20.5 9.99 19.99 1 2];
19 dx = 1;
20
21 % Characteristic speed
22 c = 0.1;
23
24 % Start and stop points
25 tStart = 0;
26 tStop = 800;
27 xStart = 0;
28 xStop = 100;
29
30 % Structure for solution
31 soln = struct('x',[],'dx',[],'t',[],'dt',[],'phi_f',[],'phi_m',[]);
32
33 for I = 1:numel(dt)
34
35     soln(I).dx = dx;
36     soln(I).dt = dt(I);
37     soln(I).x = xStart:dx:xStop;
38     soln(I).t = tStart:dt(I):tStop;
39
40     if soln(I).t(end) < tStop
41         soln(I).t(end+1) = soln(I).t(end) + soln(I).dt;
42     end
43
44 end
45
46 %=====
47 %% Solution
48 %=====
49
50
51 for K = 1:numel(dt)
52
53     % Preallocating
54     soln(K).phi_f = zeros(numel(soln(K).t),numel(soln(K).x));
55     soln(K).phi_m = zeros(numel(soln(K).t),numel(soln(K).x));
56
57     % Boundary condition
58     for n = 1:numel(soln(K).t);
59         soln(K).phi_f(n,1) = 1;
60         soln(K).phi_m(n,1) = 1;
61     end
62
63     % Initial condition
64     for I = 1:numel(soln(K).x)
65         soln(K).phi_f(1,I) = 1/(1 + exp(soln(K).x(I) - 10));
66         soln(K).phi_m(1,I) = 1/(1 + exp(soln(K).x(I) - 10));
67     end
68

```

```

69 % First order upwind
70 for n = 1:(numel(soln(K).t) - 1)
71     for I = 2:numel(soln(K).x)
72
73         soln(K).phi_f(n+1,I) = soln(K).phi_f(n,I) - ...
74             c*soln(K).dt/dx*(soln(K).phi_f(n,I) - ...
75                 soln(K).phi_f(n,I-1));
76
77     end
78 end
79
80 % Second order upwind
81 v = c*soln(K).dt/dx;
82 for n = 1:(numel(soln(K).t)-1)
83     for I = 2:numel(soln(K).x)
84
85         if I == 2
86
87             soln(K).phi_m(n+1,I) = soln(K).phi_m(n,I) - v*( ...
88                 soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1)) + 1/2*v ...
89                 *(v-1)*(soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1));
90
91         else
92
93             soln(K).phi_m(n+1,I) = soln(K).phi_m(n,I) - v*( ...
94                 soln(K).phi_m(n,I) - soln(K).phi_m(n,I-1)) + 1/2*v* ...
95                 (v-1)*(soln(K).phi_m(n,I) - 2*soln(K).phi_m(n,I-1) ...
96                     + soln(K).phi_m(n,I-2));
97
98         end
99
100     end
101 end
102
103 end
104
105 %=====
106 %% Determining critical time step
107 %=====
108
109 %-----
110 % First-order upwind
111 %-----
112
113 % Data records to compare
114 dSets = [1:4];
115
116 % Setting up figure
117 figure('Position',[40 100 660 500],'renderer','zbuffer')
118
119 % Iteratively plotting solution contours
120 for J = 1:numel(dSets)
121
122     % Plotting
123     subplot(2,2,J)
124     [X,T] = meshgrid(soln(dSets(J)).x,soln(dSets(J)).t);
125     contour(X,T,soln(dSets(J)).phi_f)
126     pcolor(X,T,soln(dSets(J)).phi_f)
127     shading interp

```

```

128
129     % Labels
130     str = sprintf('%1.2f',soln(dSets(J)).dt);
131     title(['Time step: ',str],'Interpreter','LaTeX','FontSize',fSize)
132     xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
133     ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)
134
135 end
136
137 % Saving plot
138 set(gcf,'PaperPositionMode','Auto')
139 print(gcf,'-depsc',[ImgPath,'fou_tcrit.eps'])
140
141 %-----
142 % MacCormack method variant
143 %-----
144
145 % Data records to compare
146 dSets = [5:8];
147
148 % Setting up figure
149 figure('Position',[40 100 660 500],'renderer','zbuffer')
150
151 % Iteratively plotting solution contours
152 for J = 1:numel(dSets)
153
154     % Plotting
155     subplot(2,2,J)
156     [X,T] = meshgrid(soln(dSets(J)).x,soln(dSets(J)).t);
157     contour(X,T,soln(dSets(J)).phi_m)
158     pcolor(X,T,soln(dSets(J)).phi_m)
159     shading interp
160
161     % Labels
162     str = sprintf('%1.2f',soln(dSets(J)).dt);
163     title(['Time step: ',str],'Interpreter','LaTeX','FontSize',fSize)
164     xlabel('$x$','Interpreter','LaTeX','FontSize',fSize)
165     ylabel('$t$','Interpreter','LaTeX','FontSize',fSize)
166
167 end
168
169 % Saving plot
170 set(gcf,'PaperPositionMode','Auto')
171 print(gcf,'-depsc',[ImgPath,'mmv_tcrit.eps'])
172
173 %=====
174 %% Plotting solution at specific times
175 %=====
176
177 % Find the indices at which t = 20, 200, 800 for first-order upwind
178 fouSoln = 9;
179 mmvSoln = 10;
180 tol = 0.5;
181 x1 = soln(fouSoln).x;
182 x2 = soln(mmvSoln).x;
183 n1_a = find(soln(fouSoln).t<=20+tol);
184 n1_b = find(soln(fouSoln).t>=20-tol);
185 n1_20 = intersect(n1_a,n1_b);
186 n1_a = find(soln(fouSoln).t<=200+tol);

```

```

187 n1_b = find(soln(fouSoln).t>=200-tol);
188 n1_200 = intersect(n1_a,n1_b);
189 n1_a = find(soln(fouSoln).t<=800+tol);
190 n1_b = find(soln(fouSoln).t>=800-tol);
191 n1_800 = intersect(n1_a,n1_b);
192 n1_800 = numel(soln(fouSoln).t)-1;
193
194 % Find the indices at which t = 20, 200, 800 for MacCormack variant
195 n2_a = find(soln(mmvSoln).t<=20+tol);
196 n2_b = find(soln(mmvSoln).t>=20-tol);
197 n2_20 = intersect(n2_a,n2_b);
198 n2_a = find(soln(mmvSoln).t<=200+tol);
199 n2_b = find(soln(mmvSoln).t>=200-tol);
200 n2_200 = intersect(n2_a,n2_b);
201 n2_a = find(soln(mmvSoln).t<=800+tol);
202 n2_b = find(soln(mmvSoln).t>=800-tol);
203 n2_800 = intersect(n2_a,n2_b);
204 n2_800 = numel(soln(mmvSoln).t)-1;
205
206 % Extracting solutions at desired time steps
207 phi_f_20 = soln(fouSoln).phi_f(n1_20(1),:);
208 phi_f_200 = soln(fouSoln).phi_f(n1_200(1),:);
209 phi_f_800 = soln(fouSoln).phi_f(n1_800(1),:);
210
211 phi_m_20 = soln(mmvSoln).phi_m(n2_20(1),:);
212 phi_m_200 = soln(mmvSoln).phi_m(n2_200(1),:);
213 phi_m_800 = soln(mmvSoln).phi_m(n2_800(1),:);
214
215
216 % Setting up figure
217 figure('Position',[70 10 560 640])
218
219 %-----
220 % t = 20
221 %-----
222
223 % Plotting
224 subplot(3,1,1)
225 hold on
226 plot(x1,phi_f_20,'-k','LineWidth',1.4)
227 plot(x2,phi_m_20,'--r','LineWidth',1.4)
228
229 % Adding labels and improving aesthetics
230 set(gca,'YLim',[0 1.2],'Box','on')
231 xlabel('$x$', 'Interpreter','LaTeX','FontSize',fSize);
232 ylabel('$\phi(x,20)$', 'Interpreter','LaTeX','FontSize',fSize)
233 [lh1,lh2] = legend('First-order upwind','MacCormack variant');
234 txtobj = findall(lh1,'type','text');
235 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
236 txtobj = findall(lh2,'type','text');
237 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
238 lPos = get(lh1,'Position');
239 set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
240
241 %-----
242 % t = 200
243 %-----
244
245 % Plotting

```

```

246 subplot(3,1,2)
247 hold on
248 plot(x1,phi_f_200,'-k','LineWidth',1.4)
249 plot(x2,phi_m_200,'--r','LineWidth',1.4)
250
251 % Adding labels and improving aesthetics
252 set(gca,'YLim',[0 1.2],'Box','on')
253 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
254 ylabel('$\phi(x,200)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
255 [lh1,lh2] = legend('First-order upwind', 'MacCormack variant');
256 txtobj = findall(lh1,'type','text');
257 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
258 txtobj = findall(lh2,'type','text');
259 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
260 lPos = get(lh1, 'Position');
261 set(lh1, 'Position', [lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
262
263 %-----
264 % t = 800
265 %-----
266
267 % Plotting
268 subplot(3,1,3)
269 hold on
270 plot(x1,phi_f_800,'-k','LineWidth',1.4)
271 plot(x2,phi_m_800,'--r','LineWidth',1.4)
272
273 % Adding labels and improving aesthetics
274 set(gca,'YLim',[0 1.2],'Box','on')
275 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
276 ylabel('$\phi(x,800)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
277 [lh1,lh2] = legend('First-order upwind', 'MacCormack variant');
278 txtobj = findall(lh1,'type','text');
279 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
280 txtobj = findall(lh2,'type','text');
281 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
282 set(lh1, 'Location', 'West')
283 lPos = get(lh1, 'Position');
284 set(lh1, 'Position', [lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])
285
286 % Saving figure
287 set(gcf, 'PaperPositionMode', 'Auto')
288 print(gcf, '-depsc', [ImagePath, 'Hw3PartE.eps'])
289
290 %-----
291 % Comparing with different time steps
292 %-----
293
294 % Records for solns with time step decreased by 10
295 a = 11;
296 b = 12;
297 x1p = soln(a).x;
298 x2p = soln(b).x;
299
300 % Finding indices
301 n1_20 = find(soln(a).t==20);
302 n1_200 = find(soln(a).t==200);
303 n1_800 = find(soln(a).t==800);
304 n2_20 = find(soln(b).t==20);

```

```

305 n2_200 = find(soln(b).t==200);
306 n2_800 = find(soln(b).t==800);
307
308 % Separating solutions
309 phi_fp_20 = soln(a).phi_f(n1_20,:);
310 phi_fp_200 = soln(a).phi_f(n1_200,:);
311 phi_fp_800 = soln(a).phi_f(n1_800,:);
312 phi_mp_20 = soln(b).phi_m(n2_20,:);
313 phi_mp_200 = soln(b).phi_m(n2_200,:);
314 phi_mp_800 = soln(b).phi_m(n2_800,:);
315
316 %+++++
317 % First-order upwind comparison
318 %+++++
319
320 % Setting up figure
321 figure('Position',[70 10 560 640])
322
323 % Plotting
324 subplot(3,1,1)
325 hold on
326 plot(x1,phi_f_20,'-k','LineWidth',1.4)
327 plot(x1p,phi_fp_20,'--r','LineWidth',1.4)
328
329 % Adding labels and improving aesthetics
330 set(gca,'YLim',[0 1.2],'Box','on')
331 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize);
332 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
333 ylabel('$\phi(x,20)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
334 [lh1,lh2] = legend('$\Delta t = 9.99$', '$\Delta t = 1$');
335 txtobj = findall(lh1,'type','text');
336 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
337 txtobj = findall(lh2,'type','text');
338 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
339 lPos = get(lh1, 'Position');
340 set(lh1, 'Position', [lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
341
342 % Plotting
343 subplot(3,1,2)
344 hold on
345 plot(x1,phi_f_200,'-k','LineWidth',1.4)
346 plot(x1p,phi_fp_200,'--r','LineWidth',1.4)
347
348 % Adding labels and improving aesthetics
349 set(gca,'YLim',[0 1.2],'Box','on')
350 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize);
351 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
352 ylabel('$\phi(x,200)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
353 [lh1,lh2] = legend('$\Delta t = 9.99$', '$\Delta t = 1$');
354 txtobj = findall(lh1,'type','text');
355 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
356 txtobj = findall(lh2,'type','text');
357 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
358 lPos = get(lh1, 'Position');
359 set(lh1, 'Position', [lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
360
361 % Plotting
362 subplot(3,1,3)
363 hold on

```

```

364 plot(x1,phi_f_800,'-k','LineWidth',1.4)
365 plot(x1p,phi_fp_800,'--r','LineWidth',1.4)
366
367 % Adding labels and improving aesthetics
368 set(gca,'YLim',[0 1.2],'Box','on')
369 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize);
370 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
371 ylabel('$\phi(x,800)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
372 [lh1,lh2] = legend('$\Delta t = 9.99$', '$\Delta t = 1$');
373 set(lh1, 'Location', 'West')
374 txtobj = findall(lh1, 'type', 'text');
375 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
376 txtobj = findall(lh2, 'type', 'text');
377 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
378 lPos = get(lh1, 'Position');
379 set(lh1, 'Position', [lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])
380
381 % Saving figure
382 set(gcf, 'PaperPositionMode', 'Auto')
383 print(gcf, '-depsc', [ImagePath, 'Hw3PartE1.eps'])
384
385 %+++++
386 % MacCormack method variant comparison
387 %+++++
388
389 % Setting up figure
390 figure('Position',[70 10 560 640])
391
392 % Plotting
393 subplot(3,1,1)
394 hold on
395 plot(x2,phi_m_20,'-k','LineWidth',1.4)
396 plot(x2p,phi_mp_20,'--r','LineWidth',1.4)
397
398 % Adding labels and improving aesthetics
399 set(gca,'YLim',[-0.2 1.2],'Box','on')
400 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize);
401 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
402 ylabel('$\phi(x,20)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
403 [lh1,lh2] = legend('$\Delta t = 19.99$', '$\Delta t = 2$');
404 txtobj = findall(lh1, 'type', 'text');
405 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
406 txtobj = findall(lh2, 'type', 'text');
407 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
408 lPos = get(lh1, 'Position');
409 set(lh1, 'Position', [lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
410
411 % Plotting
412 subplot(3,1,2)
413 hold on
414 plot(x2,phi_m_200,'-k','LineWidth',1.4)
415 plot(x2p,phi_mp_200,'--r','LineWidth',1.4)
416
417 % Adding labels and improving aesthetics
418 set(gca,'YLim',[-0.2 1.2],'Box','on')
419 xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize);
420 set(txtobj, 'Interpreter', 'LaTeX', 'FontSize', fSize)
421 ylabel('$\phi(x,200)$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
422 [lh1,lh2] = legend('$\Delta t = 19.99$', '$\Delta t = 2$');

```

```

423 txtobj = findall(lh1,'type','text');
424 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
425 txtobj = findall(lh2,'type','text');
426 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
427 lPos = get(lh1,'Position');
428 set(lh1,'Position',[lPos(1)*0.92 lPos(2) lPos(3)*1.15 lPos(4)])
429
430 % Plotting
431 subplot(3,1,3)
432 hold on
433 plot(x2,phi_m_800,'-k','LineWidth',1.4)
434 plot(x2p,phi_mp_800,'--r','LineWidth',1.4)
435
436 % Adding labels and improving aesthetics
437 set(gca,'YLim',[-0.2 1.2],'Box','on')
438 xlabel('$x$', 'Interpreter','LaTeX','FontSize',fSize);
439 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
440 ylabel('$\phi(x,800)$', 'Interpreter','LaTeX','FontSize',fSize)
441 [lh1,lh2] = legend('$\Delta t = 19.99$', '$\Delta t = 2$');
442 set(lh1,'Location','West')
443 txtobj = findall(lh1,'type','text');
444 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
445 txtobj = findall(lh2,'type','text');
446 set(txtobj,'Interpreter','LaTeX','FontSize',fSize)
447 lPos = get(lh1,'Position');
448 set(lh1,'Position',[lPos(1) lPos(2) lPos(3)*1.15 lPos(4)])
449
450 % Saving figure
451 set(gcf,'PaperPositionMode','Auto')
452 print(gcf,'-depsc',[ImgPath,'Hw3PartE2.eps'])

```

Post-processing

```

1  %=====
2  % Clearing workspace
3  %=====
4
5  clc,clear,close all
6
7  %=====
8  % Inputs
9  %=====
10
11 % Path for images
12 ImgPath = '../Images/';
13
14 % Font size
15 fSize = 11;
16
17 % Path to data file
18 MainPath = '../C++Files/DataFiles/';
19
20 %=====
21 % Determining critical time step for first-order upwind method
22 %=====

```

```

23
24 % Paths to first-order upwind data
25 fouData{1} = 'Hw3_tStep4.9_xStep0.5.dat';
26 fouData{2} = 'Hw3_tStep5.1_xStep0.5.dat';
27 fouData{3} = 'Hw3_tStep5.2_xStep0.5.dat';
28 fouData{4} = 'Hw3_tStep5.35_xStep0.5.dat';
29
30 % Vector of time steps
31 tStep = [4.9 5.1 5.2 5.35];
32
33 % Setting up figure
34 figure('Position',[50 40 680 450])
35
36 for I = 1:numel(tStep)
37
38     % Importing data
39     DataPath = [MainPath,fouData{I}];
40     data = csvread(DataPath);
41
42     % Separating data
43     x = data(:,1);
44     n = data(:,2);
45     phi_f = data(:,3);
46
47     % Gridding the data
48     [X,N,Phi] = griddata(x,n,phi_f,unique(x),unique(n)');
49
50     % Creating contour plot
51     subplot(2,2,I)
52     [c,h] = contour(X,N,Phi);
53
54     % Adding labels
55     str = sprintf('%1.2f',tStep(I));
56     title(['Time step: ',str]);
57     xlabel('$x$', 'Interpreter','LaTeX','FontSize',fSize)
58     ylabel('$t$', 'Interpreter','LaTeX','FontSize',fSize)
59
60 end
61
62 % Saving figure
63 set(gcf,'PaperPositionMode','Auto')
64 print(gcf,'-depsc',[ImgPath,'fou_tcrit.eps'])
65
66 %=====
67 % Determining critical time step for MacCormack method variant
68 %=====
69
70 % Paths to first-order upwind data
71 mmvData{1} = 'Hw3_tStep4.9_xStep0.5.dat';
72 mmvData{2} = 'Hw3_tStep5.1_xStep0.5.dat';
73 mmvData{3} = 'Hw3_tStep5.2_xStep0.5.dat';
74 mmvData{4} = 'Hw3_tStep10_xStep0.5.dat';
75
76 % Vector of time steps
77 tStep = [4.9 5.1 5.2 8];
78
79 % Setting up figure
80 figure('Position',[50 40 680 450])
81

```

```

82 for I = 1:numel(tStep)
83
84     % Importing data
85     DataPath = [MainPath,mmvData{I}];
86     data = csvread(DataPath);
87
88     % Separating data
89     x = data(:,1);
90     n = data(:,2);
91     phi_m = data(:,4);
92
93     % Gridding the data
94     [X,N,Phi] = griddata(x,n,phi_m,unique(x),unique(n)');
95
96     % Creating contour plot
97     subplot(2,2,I)
98     [c,h] = contour(X,N,Phi);
99
100    % Adding labels
101    str = sprintf('%1.2f',tStep(I));
102    title(['Time step: ',str]);
103    xlabel('$x$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
104    ylabel('$t$', 'Interpreter', 'LaTeX', 'FontSize', fSize)
105
106 end
107
108 % Saving figure
109 set(gcf, 'PaperPositionMode', 'Auto')
110 print(gcf, '-depsc', [ImgPath, 'mmv_tcrit.eps'])

```

C++ code

```

1  #include "cptstd.hpp"
2  #include <fstream>
3  #include <sstream>
4  #include <vector>
5  #include <string>
6  #include <cmath>
7  #include "matrix.hpp"
8  #include <iomanip>
9
10 using namespace cpt;
11
12 /*-----
13     This program must be compiled using g++ -o Hw3 -std=c++0x Hw3.cpp because
14     of a num to string conversion that requires C++ 11 capabilities.
15     -----*/
16
17 int main() {
18
19     //=====
20     // Declaring variables and generating grid
21     //=====
22

```

```

23 cout << "\n===== " << endl;
24 cout << " | AE 5327 -- Homework 3 | " << endl;
25 cout << "===== \n" << endl;
26
27 // Asking the user for time step
28 double dt,dx;
29 cout << "Enter dt: ";
30 cin >> dt;
31 cout << "Enter dx: ";
32 cin >> dx;
33
34 // Setting up the time vector
35 double tStart = 0.0;
36 double tStop = 800.0;
37 double tPoints;
38 tPoints = ceil((tStop - tStart)/dt);
39 vector<double> t;
40 t.reserve((int) tPoints);
41
42 // Setting up x vector
43 double xStart = 0.0;
44 double xStop = 100.0;
45 double xPoints;
46 xPoints = ceil((xStop - xStart)/dx);
47 vector<double> x;
48 x.reserve((int) xPoints);
49
50 // Generating the grid
51 for (int i=0; i<=((int) tPoints); i++){
52
53     if (i==0){
54         t[i] = tStart;
55     }
56     else {
57         t[i] = t[i-1] + dt;
58     }
59
60 }
61 for (int j=0; j<=((int) xPoints); j++){
62
63     if (j==0) {
64         x[j] = xStart;
65     }
66     else {
67         x[j] = x[j-1] + dx;
68     }
69
70 }
71
72 double c = 0.1;
73 double v;
74 int nRow = (int) tPoints;
75 int nCol = (int) xPoints;
76
77 Matrix<double,2> phi_f(nRow,nCol);
78 Matrix<double,2> phi_s(nRow,nCol);
79 Matrix<double,2> phi_m(nRow,nCol);
80 phi_f = Matrix<double,2>(nRow,nCol);
81 phi_s = Matrix<double,2>(nRow,nCol);

```

```

82  phi_m = Matrix<double,2>(nRow,nCol);
83  cout << "\n" << nRow << " by " << nCol << " grid with " << phi_f.size()
84      << " cells." << endl;
85
86  //=====
87  // First-order upwind solution
88  //=====
89
90  // Enforcing boundary condition
91  for (int n=0; n<nRow; n++) {
92      phi_f[n][0] = 1.0;
93  }
94
95  // Enforcing initial condition
96  for (int i=0; i<nCol; i++) {
97      phi_f[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
98  }
99
100  for (int n=0; n<(nRow-1); n++) {
101
102      for (int i=1; i<nCol; i++) {
103
104          phi_f[n+1][i] = phi_f[n][i] - c*dt/dx*(phi_f[n][i] - phi_f[n][i-1]);
105
106      }
107
108  }
109
110  //=====
111  // Second-order upwind solution
112  //=====
113
114  // Enforcing boundary condition
115  for (int n=0; n<nRow; n++) {
116      phi_s[n][0] = 1.0;
117  }
118
119  // Enforcing initial condition
120  for (int i=0; i<nCol; i++) {
121      phi_s[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
122  }
123
124  for (int n=0; n<(nRow-1); n++) {
125
126      for (int i=1; i<nCol; i++) {
127
128          if (i==1) {
129
130              phi_s[n+1][i] = phi_s[n][i] - c*dt/(2.0*dx)*(3.0*phi_s[n][i] -
131                  3.0*phi_s[n][i-1]);
132
133          }
134          else {
135
136              phi_s[n+1][i] = phi_s[n][i] - c*dt/(2.0*dx)*(3.0*phi_s[n][i] -
137                  4.0*phi_s[n][i-1] + phi_s[n][i-2]);
138
139          }
140

```

```

141     }
142
143 }
144
145 //=====
146 // MacCormack method variant
147 //=====
148
149 // Enforcing boundary condition
150 for (int n=0; n<nRow; n++) {
151     phi_m[n][0] = 1.0;
152 }
153
154 // Enforcing initial condition
155 for (int i=0; i<nCol; i++) {
156     phi_m[0][i] = 1.0/(1.0 + exp(x[i] - 10.0));
157 }
158
159 // CFL number
160 v = c*dt/dx;
161
162 for (int n=0; n<(nRow-1); n++) {
163
164     for (int i=1; i<nCol; i++) {
165
166         if (i==1) {
167
168             phi_m[n+1][i] = phi_m[n][i] - v*(phi_m[n][i] - phi_m[n][i-1]) +
169                 1/2*v*(v-1)*(phi_m[n][i] - phi_m[n][i-1]);
170
171         }
172         else {
173
174             phi_m[n+1][i] = phi_m[n][i] - v*(phi_m[n][i] - phi_m[n][i-1]) +
175                 1/2*v*(v-1)*(phi_m[n][i] - 2.0*phi_m[n][i-1] + phi_m[n][i-2]);
176
177         }
178
179     }
180
181 }
182
183 //=====
184 // Writing data to file
185 //=====
186
187 // Constructing file name
188 stringstream str1;
189 stringstream str2;
190 str1 << setprecision(3) << dt;
191 str2 << setprecision(3) << dx;
192 string tStepStr = str1.str();
193 string xStepStr = str2.str();
194 string fName = "Hw3_tStep" + tStepStr + "_xStep" + xStepStr + ".dat";
195 string fullfName = "./DataFiles/" + fName;
196
197 // Outputting some info
198 cout << "\nWriting data file...\n\n";
199 cout << "Solution in file " << fName << endl;

```

```

200
201 // Opening stream and writing data file
202 ofstream data_file;
203 data_file.open(fullfName);
204 double prog = 0;
205 for (int i=0; i<nCol; i++) {
206     for (int j=0; j<nRow; j++) {
207
208         data_file << setprecision(10) << x[i] << ", " << t[j] << ", " << phi_f[j][i]
209             << ", " << phi_m[j][i] << endl;
210
211     }
212     data_file << "\n";
213
214 }
215
216 // Closing data file
217 data_file.close();
218
219 return 0;
220 }

```