Amath 586: Homework 5

Due on May 26, 2015

Brian de Silva

Problem 1

Let $U = [U_0,\ U_1,\ \ldots,\ U_m]^T$ be a vector of function values at equally spaced points on the interval $0 \le x \le 1$, and suppose the underlying function is periodic and smooth. Then we can approximate the first derivative u_x at all of these points by DU, where D is circulant matrix such as

$$D_- = \frac{1}{h}\begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{bmatrix}, \qquad D_+ = \frac{1}{h}\begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{bmatrix}$$

for first-order accurate one-sided approximations or

$$D_0 = \frac{1}{2h}\begin{bmatrix} 0 & 1 & & & -1 \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & -1 & 0 & 1 \\ 1 & & & -1 & 0 \end{bmatrix}$$

for a second-order accurate centered approximation. (These are illustrated for a grid with $m + 1 = 5$ unknowns and $h = 1/5$.)
The advection equation $u_t + au_x = 0$ on the interval $0 \le x \le 1$ with periodic boundary conditions gives rise to the MOL discretization $U'(t) = -aDU(t)$ where D is one of the matrices above.

(a) Discretizing $U' = -aD_-U$ by forward Euler gives the first order upwind method

$$U_j^{n+1} = U_j^n - \frac{ak}{h}(U_j^n - U_{j-1}^n),$$

where the index i runs from 0 to m with addition of indices performed mod $m + 1$ to incorporate the periodic boundary conditions.

Suppose instead we discretize the MOL equation by the second-order Taylor series method,

$$U^{n+1} = U^n - akD_-U^n + \frac{1}{2}(ak)^2 D_-^2 U^n.$$

Compute D_-^2 and also write out the formula for U_j^n that results from this method.

(b) How accurate is the method derived in part (a) compared to the Beam-Warming method, which is also a 3-point one-sided method?

(c) Suppose we make the method more symmetric:

$$U^{n+1} = U^n - \frac{ak}{2}(D_+ + D_-)U^n + \frac{1}{2}(ak)^2 D_+D_-U^n.$$

Write out the formula for U_j^n that results from this method. What standard method is this?

Solution:

(a) D_-^2 is easily computed:

$$D_-^2 = \frac{1}{h^2}\begin{bmatrix} 1 & & & & & & 1 & -2 \\ -2 & 1 & & & & & & 1 \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 & 1 \end{bmatrix}.$$

The formula that results from this method is given by

$$U_j^{n+1} = U_j^n - \frac{ak}{h}\left(U_j^n - U_{j-1}^n\right) + \frac{1}{2}\left(\frac{ak}{h}\right)^2\left(U_j^n - 2U_{j-1}^n + U_{j-2}\right), \quad j = 1, 2, \ldots$$

where we identify U_0^n with U_{m+1}^n and U_{-1}^n with U_m^n because of the periodic boundary conditions.

(b) This method is only $O(k^2 + h)$ as it uses a second order discretization in time and a first order one in space, whereas the Beam-Warming method is $O(k^2 + h^2)$ accurate. This method is the worse of the two 3-point one-sided methods.

(c) The formula that results from making this method symmetric is

$$U_j^{n+1} = U_j^n - \frac{ak}{2h}\left(U_{j+1}^n - U_{j-1}^n\right) + \frac{1}{2}\left(\frac{ak}{h}\right)^2\left(U_{j-1}^n - 2U_j^n + U_{j+1}^n\right),$$

which is just the Lax-Wendroff method.

Problem 2

(a) Produce a plot similar to those shown in Figure 10.1 for the upwind method (10.21) with the same values of $a = 1$, $h = 1/50$ and $k = 0.8h$ used in that figure.

(b) Produce the corresponding plot if the one-sided method (10.22) is instead used with the same values of a, h, and k.

Solution:

(a) Using the upwind method and the same values of $a = 1$, $h = 1/50$ and $k = 0.8h$ as in Figure 10.1 we produce the plot shown in Figure 1.

(b) Method (10.22) and the same values as above produces the plot shown in Figure 2.

See the appendix for the Matlab code that generated these plots.

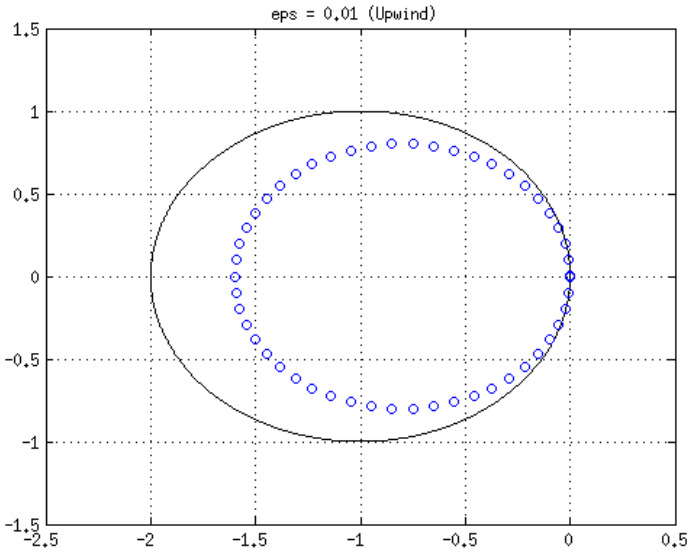


Figure 1: Eigenvalues of A_ϵ with $\epsilon = \frac{ah}{2}$

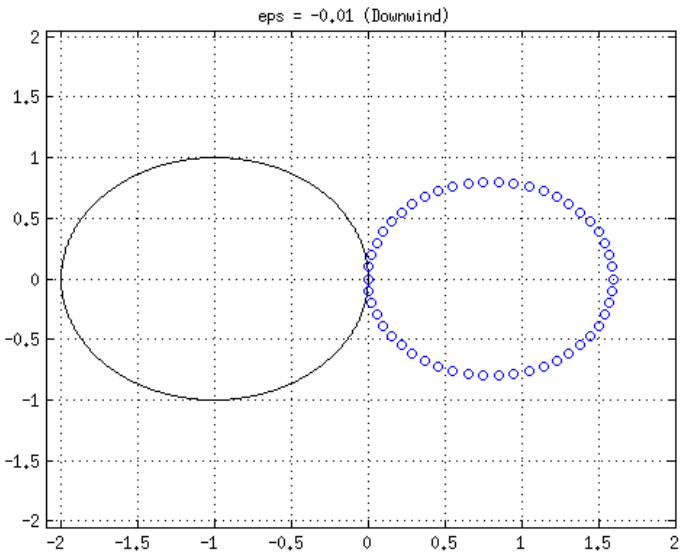


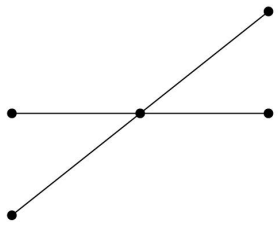
Figure 2: Eigenvalues of A_ϵ with $\epsilon = -\frac{ah}{2}$

Problem 3

Suppose $a > 0$ and consider the following *skewed leapfrog* method for solving the advection equation $u_t + au_x = 0$:

$$U_j^{n+1} = U_{j-2}^{n-1} - \left(\frac{ak}{h} - 1\right)(U_j^n - U_{j-2}^n).$$

The stencil of this method is



Note that if $ak/h \approx 1$ then this stencil roughly follows the characteristic of the advection equation and might be expected to be more accurate than standard leapfrog. (If $ak/h = 1$ the method is exact.)

(a) What is the order of accuracy of this method?

(b) For what range of Courant number ak/h does this method satisfy the CFL condition?

(c) Show that the method is in fact stable for this range of Courant numbers by doing von Neumann analysis. **Hint:** Let $\gamma(\xi) = e^{i\xi h}g(\xi)$ and show that γ satisfies a quadratic equation closely related to the equation (10.34) that arises from a von Neumann analysis of the leapfrog method.

Solution:

(a) We need the following Taylor expansions to compute the truncation error for this method:

$$\begin{aligned}
u(x, t+k) &= u + ku_t + \frac{k^2}{2}u_{tt} + \frac{k^3}{3!}u_{ttt} + O(k^4), \\
u(x-2h, t) &= u - 2hu_x + 2h^2u_{xx} - \frac{4}{3}h^3u_{xxx} + O(h^4), \\
u(x-2h, t-k) &= u - 2hu_x - ku_t + \frac{1}{2}(4h^2u_{xx} + k^2u_{tt} + 4khu_{xt}) \\
&\quad - \frac{1}{6}(8h^3u_{xxx} + k^3u_{ttt} + 6hk^2u_{xtt} + 12h^2ku_{xxt}) + \dots.
\end{aligned}$$

Rewriting the method, we obtain an expression for the truncation error:

$$\tau^n = \frac{U_j^{n+1} - U_{j-2}^{n-1}}{k} + \frac{1}{k}\left(\frac{ak}{h} - 1\right)(U_j^n - U_{j-2}^n).$$

Substituting in the above Taylor expansions, cancelling terms, and using the PDE to simplify the resulting equation further, we get

$$\tau^n = \frac{k^2}{3}u_{ttt} + hku_{ttx} + 2h^2uxxt + \frac{4}{3}ah^2u_{xxx} + \dots = O((h+k)^2).$$

Hence the method is second order accurate in both space and time. (Note: although I did above the calculations by hand, I checked my answer using a Mathematica notebook which can be found in the Appendix).

Problem 3 continued on next page...

(b) Using the skewed leapfrog formula, we can trace the dependence of an approximation U_j^n back to the points $U_{j-2n}^0, U_{j-2(n-1)}^0, \ldots, U_j^0$. Fixing $\frac{k}{h} = r$ we find that as we refine the grid, the numerical domain of dependence for a point (X,T) corresponding to U_j^n converges to $[X - 2T/r, X]$. For $a > 0$ the domain of dependence of the PDE at (X,T) is $\{X - aT\}$. Hence in order for the method to satisfy the CFL condition, we need that $X - 2T/r \leq X - aT \leq X$. For $a > 0$ this means that we must have $ar \leq 2 \;\Rightarrow\; 0 \leq \frac{ak}{h} \leq 2$.

(c) Let $\nu = \frac{ak}{h}$, the Courant number. Setting $U_j^n = g(\xi)^n e^{ijh\xi}$ in the skewed leapfrog method yields

$$g(\xi)^{n+1} e^{ijh\xi} = g(\xi)^{n-1} e^{i(j-2)h\xi} - g(\xi)^n (\nu - 1)\left(e^{ijh\xi} - e^{i(j-2)h\xi}\right).$$

Dividing by $g(\xi)^{n-1} e^{i(j-2)h\xi}$ gives

$$g(\xi)^2 e^{2ih\xi} = 1 - g(\xi)(\nu - 1)\left(e^{2ih\xi} - 1\right).$$

Finally, setting $\gamma(\xi) = g(\xi) e^{ijh\xi}$ in the above, we have

$$\begin{aligned} &\gamma(\xi)^2 = 1 - \gamma(\xi)(\nu - 1)\left(e^{ih\xi} - e^{-ih\xi}\right) \\ \Rightarrow\; &\gamma(\xi)^2 = 1 - 2i(\nu - 1) sin(h\xi)\gamma(\xi). \end{aligned}$$

Applying the same analysis as in Example 10.4, we see that in order for the method to be stable (i.e. $|\gamma(\xi)| = |g(\xi) e^{ijh\xi}| = |g(\xi)| \leq 1$), we require $|\nu - 1| \leq 1 \;\Rightarrow\; 0 \leq \nu \leq 2$, the same restriction that the CFL condition imposes. Therefore the method is stable for the range of Courant numbers satisfying the CFL condition.

Problem 4

Derive the modified equation (10.45) for the Lax-Wendroff method.

Solution:

The Lax-Wendroff method is given by

$$U_j^{n+1} = U_j^n - \frac{ak}{2h}(U_{j+1}^n - U_{j-1}^n) + \frac{a^2k^2}{2h^2}(U_{j-1}^n - 2U_j^n + U_{j+1}^n).$$

Supposing that $v(x,t)$ satisfies this difference equation exactly, we substitute it into the equation, Taylor expand about (x,t), and simplify to get

$$\left(kv_t + \frac{k^2}{2}v_{tt} + \frac{k^3}{6}v_{ttt} + \frac{k^4}{4!}v_{4t} + O(k^5)\right) + ak\left(v_x + \frac{h^2}{3!}v_{xxx} + O(h^4)\right) - \frac{a^2k^2}{2}\left(v_{xx} + \frac{h^2}{12}v_{4x} + O(h^4)\right) = 0.$$

This can be rewritten as

$$v_t + av_x = \frac{k}{2}\left(a^2 v_{xx} - v_{tt}\right) - \frac{1}{6}\left(k^2 v_{ttt} + ah^2 v_{xxx}\right) + O(k^3 + h^3 + h^2 k). \tag{1}$$

Dropping higher order terms and differentiating with respect to x and t gives

$$\begin{aligned} v_{tx} &= -av_{xx} + \frac{k}{2}\left(a^2 v_{xxx} - v_{ttx}\right) - \frac{1}{6}\left(k^2 v_{tttx} + ah^2 v_{4x}\right), \\ v_{tt} &= -av_{xt} + \frac{k}{2}\left(a^2 v_{xxt} - v_{ttt}\right) - \frac{1}{6}\left(k^2 v_{4t} + ah^2 v_{xxxt}\right). \end{aligned}$$

Problem 4 continued on next page...

Combining these, we obtain

$$v_{tt} = a^2 v_{xx} - \frac{ak}{2}\left(a^2 v_{xxx} - v_{ttx}\right) + \frac{k}{2}\left(a^2 v_{xxt} - v_{ttt}\right) + O(k^2 + h^2).$$

Substituting this into (1) and ignoring higher order terms yields

$$v_t + a v_x = \frac{k^2}{4}\left(a^2 v_{xxt} - v_{ttt} - a^3 v_{xxx} + a v_{ttx}\right) - \frac{1}{6}\left(k^2 v_{ttt} + a h^2 v_{xxx}\right).$$

Differentiating earlier equations with respect to x and t once more gives

$$\begin{aligned} v_{txx} &= -a v_{xxx} + O(k), \\ v_{ttx} &= a^2 v_{xxx} + O(k), \\ v_{ttt} &= a^2 v_{xxt} + O(k) = -a^3 (v_{xxx}) + O(k). \end{aligned}$$

Finally, substituting these into the above and ignoring higher order terms yields

$$\begin{aligned} v_t + a v_x &= \frac{k^2}{4}\left(a^2 v_{xxt} - (a^2 v_{xxt}) - a^3 v_{xxx} + a(a^2 v_{xxx})\right) - \frac{1}{6}\left(k^2(-a^3 v_{xxx}) + a h^2 v_{xxx}\right) \\ &= -\frac{a h^2}{6}\left(1 - \left(\frac{ak}{h}\right)^2\right) v_{xxx}. \end{aligned}$$

Hence we obtain the desired expression

$$v_t + a v_x + \frac{a h^2}{6}\left(1 - \left(\frac{ak}{h}\right)^2\right) v_{xxx} = 0.$$

Problem 5

The m-file `advection_LW_pbc.m` from the book repository implements the Lax-Wendroff method for the advection equation on $0 \le x \le 1$ with periodic boundary conditions.
The class repository contains a Python translation, `$AM586/codes/advection_LW_pbc.py`.
You might want to first experiment with these and see how Lax-Wendroff behaves for various grid resolutions. Note that the way this problem is set up, the solution advects twice around the domain over time 1 and should end up agreeing with the initial data.

(a) Modify one of these files to implement the leapfrog method and verify that this is second order accurate. Note that you will have to specify two levels of initial data. For the convergence test set $U_j^1 = u(x_j, k)$, the true solution at time k.

(b) Modify your code so that the initial data consists of a wave packet

$$\eta(x) = \exp(-\beta(x - 0.5)^2)\sin(\xi x)$$

Work out the true solution $u(x,t)$ for this data. Using $\beta = 100$, $\xi = 80$ and $U_j^1 = u(x_j, k)$, test that your code still exhibits second order accuracy for k and h sufficiently small.

(c) Using $\beta = 100$, $\xi = 150$ and $U_j^1 = u(x_j, k)$, estimate the group velocity of the wave packet computed with leapfrog using $m = 199$ and $k = 0.4h$. How well does this compare with the value (10.52) predicted by the modified equation?

Note: Early editions of the text book had a typo in (10.52). There should be a factor ν^2 in the denominator.

Problem 5 continued on next page...

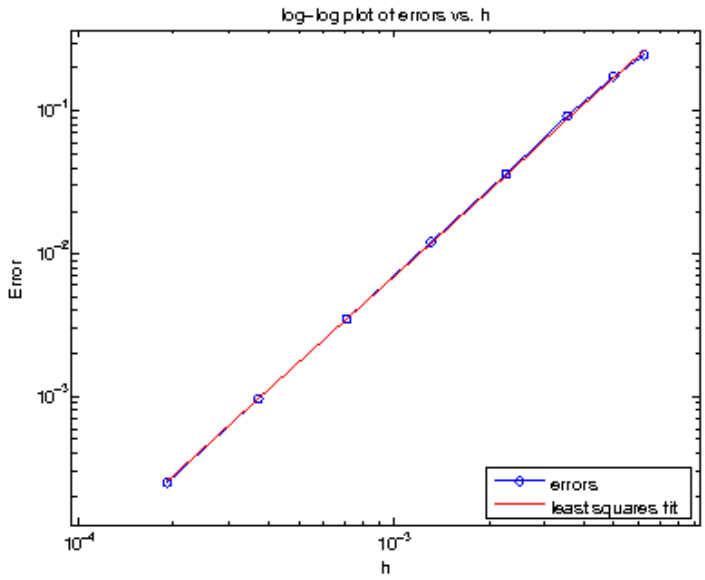


Figure 3: Error at $t = 1$ using Leapfrog method

Solution:

(a) Figure (3) shows the error at $t = 1$ of the Leapfrog method as the grid is refined. The numerically approximated order of accuracy was 1.99 so the method is exhibiting second order accuracy.

(b) For the simple advection equation the solution is simply $\eta(x - at)$ modified so that it is periodic with period 1. As shown in Figure (4), the method is still second order accurate on this new initial data (approximate order of accuracy: 1.96). Notice however that the grid spacing had to be decreased in order for the desired second order accuracy to become apparent. For larger values of h and k the method was not converging nearly as quickly.

(c) Following the peak of the wave packet of the numerical solution, we can approximate its wave speed. We see that it travels a total distance of 1.752 over 1 unit of time, so the group velocity is roughly 1.75. Equation (10.52) uses the modified equation to approximate the group velocity of the wave packet computed with leapfrog as

$$\pm \frac{acos(\xi h)}{\sqrt{(1 - \nu^2 sin^2(\xi h))}} \approx 1.746$$

using the given parameters $a = 2, \nu = \frac{4}{5}, h = \frac{1}{200}, \xi = 150$. This agrees almost perfectly with the observed group velocity.

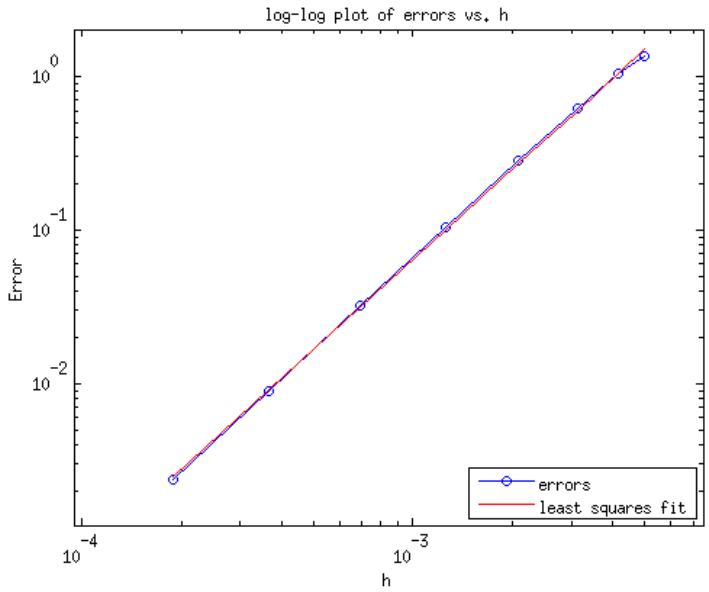


Figure 4: Error at $t = 1$ using Leapfrog method with alternate initial conditions

Appendix: Code

Amath 586 HW 5 Matlab Notebook

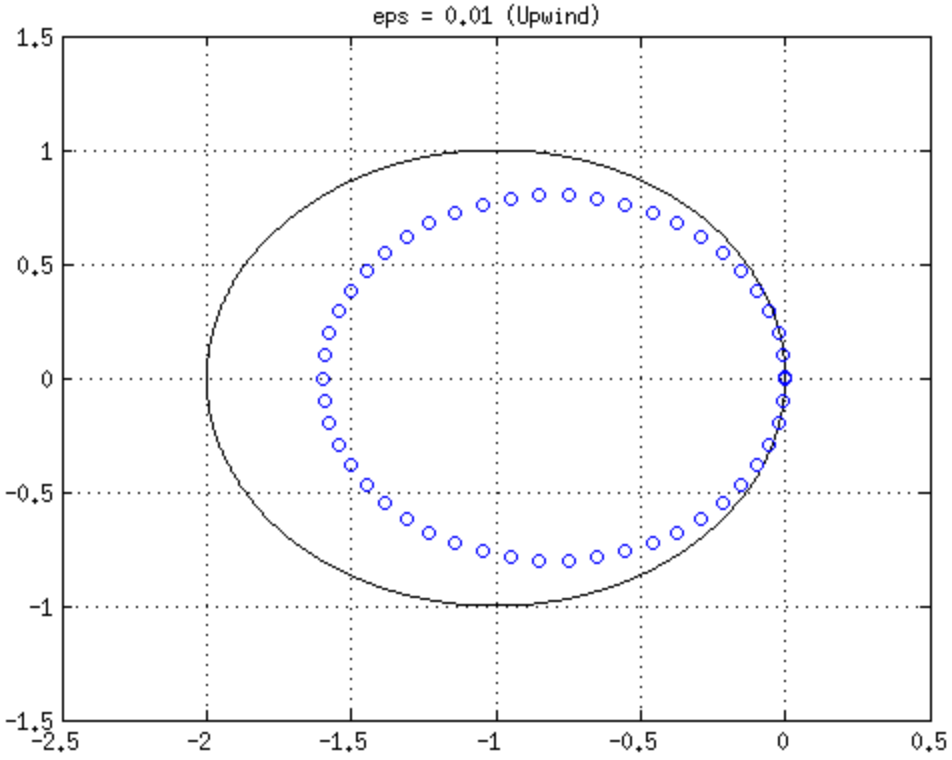
Table of Contents

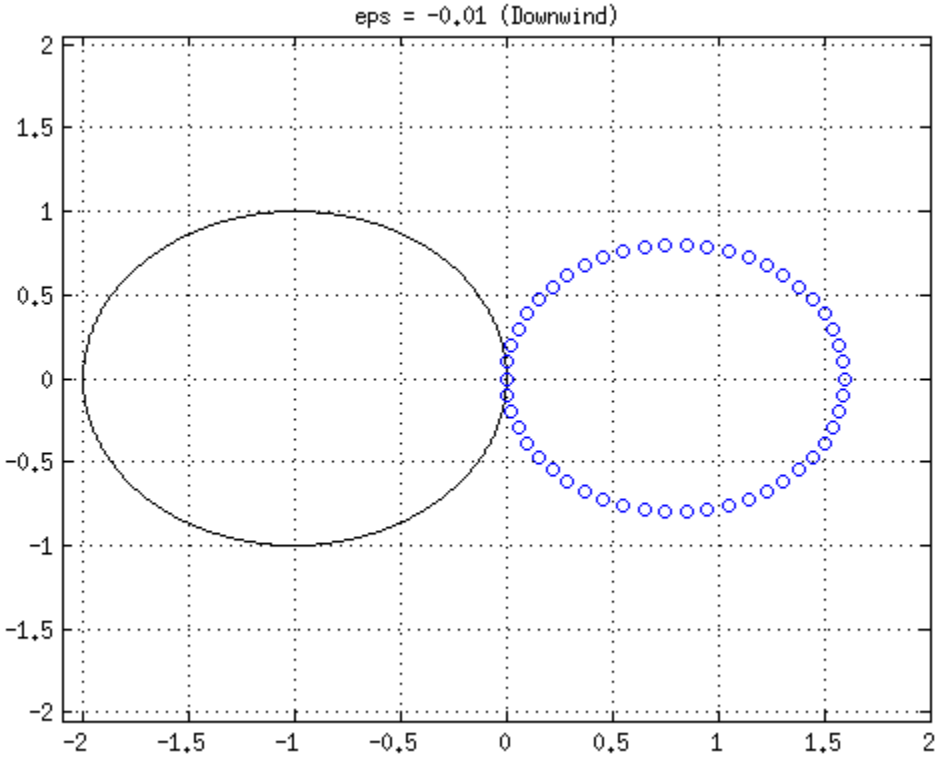
Problem 2 1
Problem 5 3

Brian de Silva

Problem 2

```
% Set the relevant parameters
a = 1;
h = 1/50;
k = 0.8*h;
p = [0:round(1/h)]';

shifted_unit_circle = [cos(0:0.001:2*pi) - 1; sin(0:0.001:2*pi) ].';


% Part (a)
epsilon = a*h/2;

% Find eigenvalues * k
kmu = k*[(-2*epsilon/(h^2))*(1-cos(2*pi*p*h)), (-a/h)*sin(2*pi*p*h)];

% Plot results
Fig1 = figure(1);
plot(shifted_unit_circle(:,1),shifted_unit_circle(:,2),'k-'), hold on
plot(kmu(:,1),kmu(:,2), 'bo');
axis([-2.5 0.5 -1.5 1.5]);
title('eps = 0.01 (Upwind) ')
grid on


% Part (b)
epsilon = -a*h/2;

% Find eigenvalues * k
kmu = k*[(-2*epsilon/(h^2))*(1-cos(2*pi*p*h)), (-a/h)*sin(2*pi*p*h)];

% Plot results
Fig2 = figure(2);
plot(shifted_unit_circle(:,1),shifted_unit_circle(:,2),'k-'), hold on
plot(kmu(:,1),kmu(:,2), 'bo');
axis([-2.1 2 -2.05 2.05]);
title('eps = -0.01 (Downwind) ')
grid on
```


eps = 0.01 (Upwind)
1.5
1
0.5
0
-0.5
-1
-1.5
-2.5
-2
-1.5
-1
-0.5
0
0.5


eps = -0.01 (Downwind)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Problem 5

```
close all

% Part (a) - Leapfrog
IC = 1;
numTrials = 8;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% advection_Leap_pbc(239);

% Approximate PDE using increasing number of grid points (160,200,280,...)
% Note: we choose k=0.4h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = advection_Leap_pbc(120+40*2^(j-1) - 1,IC);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);

% Computed order of accuracy: 1.99
% Constant is O(10^3)


% Part (b) - New initial data
IC = 2;
numTrials = 8;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);


% Approximate PDE using increasing number of grid points (200,240,320,...)
% Note: we choose k=0.4h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = advection_Leap_pbc(160+40*2^(j-1) - 1,IC);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);

% Computed order of accuracy: 1.96
% Constant is O(10^4)


% Part (c)
IC = 3;

% Note: I had advection_Leap_pbc output multiple
% plots when I ran this line originally
[h,k,~]=advection_Leap_pbc(199,IC);
```

```
Least squares fit gives E(h) = 6681.15 * h^1.9949

Least squares fit gives E(h) = 47436.5 * h^1.95563
```

Published with MATLAB® 7.14

MathematicaNotebook

Brian de Silva

Problem 3

In[7]:=
```
(* Automate  multivariable  Taylor expansion to third order *)
U[j_, n_] :=
    u + j*h*u_x + n*k*u_t + ((j*h)^2)*u_xx/2 + ((n*k)^2)*u_tt/2 + n*j*h*k*u_tx + (1/6)*
        (u_xxx (j*h)^3 + u_ttt (n*k)^3 + 3 u_xxt n*k* (j*h)^2 + 3 u_ttx j*h* (n*k)^2) + hots1;

(* Check local truncation error for skewed leapfrog method  *)
lte = Expand[(U[0, 1] - U[-2, -1])/k + ((a)/h - 1/k)*(U[0, 0] - U[-2, 0])
  ]
```

Out[8]=
$$2 u_t + \frac{k^2 u_{ttt}}{3} + h k u_{ttx} - 2 h u_{tx} + 2 a u_x - 2 a h u_{xx} + 2 h^2 u_{xxt} + \frac{4}{3} a h^2 u_{xxx}$$

Problem 4

In[9]:=
```
(* Automate  single variablte Taylor expansion to fourth order *)
U[j_] :=
    u + j*h*u_x + ((j*h)^2)*u_xx/2 + (1/6)*u_xxx (j*h)^3 + (1/24)*u_xxxx (j*h)^4 + hots2;

(* Check truncation error for modified  equation - Lax-Wendroff *)
Expand[(U[0, 1] - U[0] + (a*k/(2h))*(U[1] - U[-1]) -
        (1/2)*((a*k/h)^2)*(U[-1] - 2U[0] + U[1]))/k]
```

Out[10]=
$$\frac{hots1}{k} - \frac{hots2}{k} + u_t + \frac{k u_{tt}}{2} + \frac{k^2 u_{ttt}}{6} + a u_x - \frac{1}{2} a^2 k u_{xx} + \frac{1}{6} a h^2 u_{xxx} - \frac{1}{24} a^2 h^2 k u_{xxxx}$$