Mario L. Gutierrez Abed
364009832
mlg3843@rit.edu

Problem Set 4
Numerical Methods for PDEs

04-28-2021

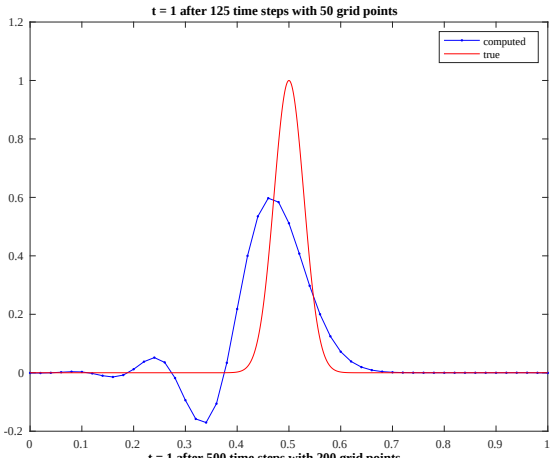
Problem 1. *The m-file* `advection_LW_pbc.m` *implements the Lax-Wendroff method for the advection equation* $u_t + au_x = 0$ *on* $0 \leq x \leq 1$ *with periodic boundary conditions.*

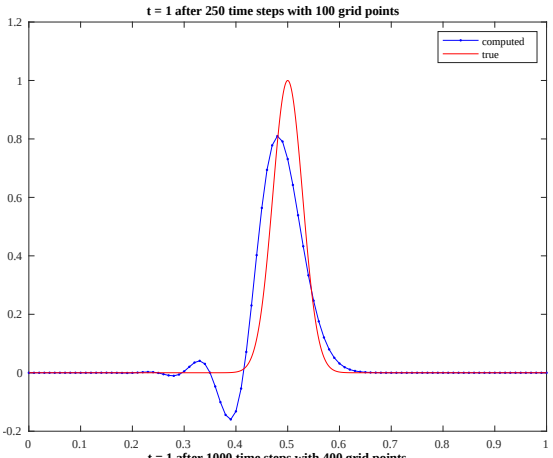
a) Observe how this behaves with $m + 1 = 50,\ 100,\ 200,\ 400$ *grid points. Include plots of the solution in your results. Change the final time to* $t_{final} = 0.1$ *and include a numerical verification of second order accuracy (create a table using* `error_table.m` *).*

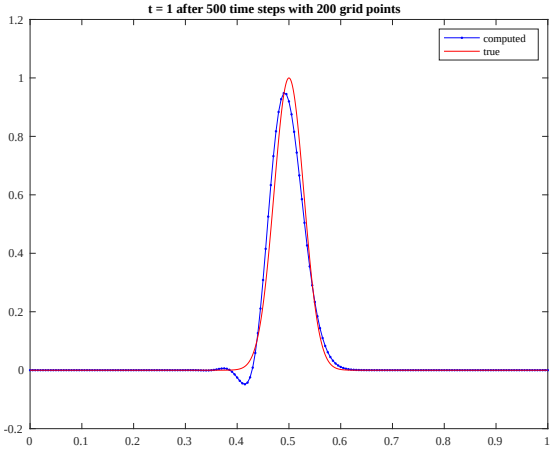
b) Modify the file to create a version `advection_up_pbc.m` *implementing the upwind method and include a numerical verification of first order accuracy.*

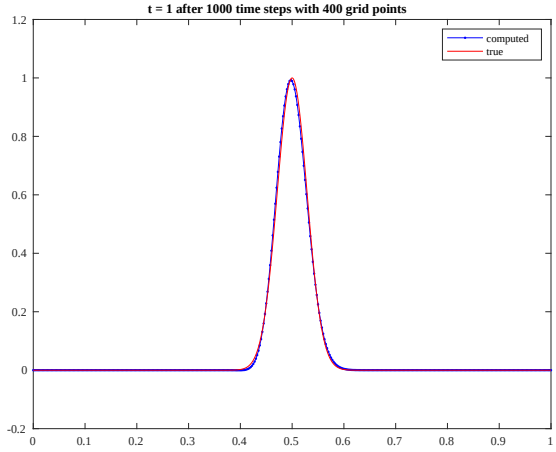
c) Keep m fixed and observe what happens with `advection_up_pbc.m` *if the time step* Δt *is reduced, e.g. try* $\Delta t = 0.4\Delta x$, $\Delta t = 0.2\Delta x$, $\Delta t = 0.1\Delta x$. *When a convergent method is applied to an ODE we expect better accuracy as the time step is reduced and we can view the upwind method as an ODE solver applied to an MOL system. However, you should observe decreased accuracy as* $\Delta t \to 0$ *with* Δx *fixed. Include an error table here and explain this apparent paradox.* **Hint:** *Which ODE system are you solving by using method of lines? Follow the* modified equation *idea and write down the Taylor expansion of the spatial derivative* u_x *for either* $a > 0$ *or* $a < 0$ *case.*

Solution to a). Here are the solution plots for $m + 1 = 50,\ 100,\ 200,\ 400$ grid points, all with $t_{\text{final}} = 1$:









The following snippet creates the desired table, with $t_{\text{final}} = 0.1$:

```
m_vec = [49; 99; 199; 399];
h_vec = zeros(length(m_vec),1);
error_vec = zeros(length(m_vec),1);

for i = 1 : length(m_vec)
    [h_vec(i), ~, error_vec(i)] = advection_LW_pbc(m_vec(i));
end

error_table(h_vec, error_vec)
```

The output shows second-order accuracy:

h	error	ratio	observed order
0.02000	1.66622e-01	NaN	NaN
0.01000	6.27543e-02	2.65514	1.40879
0.00500	1.70902e-02	3.67195	1.87654
0.00250	4.30103e-03	3.97352	1.99042

We can also see the second-order accuracy by choosing any two error values, say ${}^{h=0.0025}\|e\|_\infty = 4.30103 \times 10^{-3}$ and ${}^{h=0.005}\|e\|_\infty = 1.70902 \times 10^{-2}$, and looking at their ratio

$$\frac{4.30103 \times 10^{-3}}{1.70902 \times 10^{-2}} = 0.251666 \approx \frac{1}{4},$$

we see that cutting h in half cuts the error by 1/4; thus we have $O(h^2)$ convergence. ♠

Solution to b). The modified code yields the following table of errors (with $t_{\text{final}} = 0.1$):

h	error	ratio	observed order
0.02000	2.81062e-01	NaN	NaN
0.01000	1.78422e-01	1.57526	0.65559
0.00500	1.02013e-01	1.74902	0.80654
0.00250	5.50908e-02	1.85172	0.88887

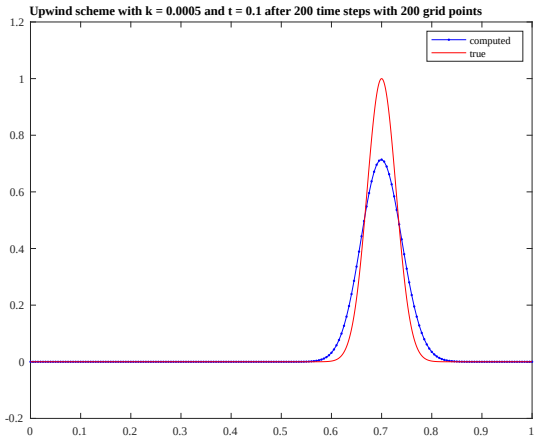
We can also see the first-order accuracy by choosing any two error values, say ${}^{h=0.0025}\|e\|_\infty = 5.50908 \times 10^{-2}$ and ${}^{h=0.005}\|e\|_\infty = 1.02013 \times 10^{-1}$, and looking at their ratio

$$\frac{5.50908 \times 10^{-2}}{1.02013 \times 10^{-1}} = 0.540037 \approx \frac{1}{2},$$

we see that cutting h in half cuts the error (roughly) in half also; thus we have $O(h)$ convergence. ♠

Solution to c). Here is the table of errors (for $m = 199$ fixed, and $t_{\text{final}} = 0.1$):

k	$\|e\|_\infty$
$0.4h$	0.1020
$0.2h$	0.2378
$0.1h$	0.2859



The table shows that the error is, in fact, increasing with decreasing k. The figure shows the relatively large error output when we set $k = 0.1h$. This apparent "paradox" is due to the smearing of the numerical solution, which increases with decreasing k because the diffusive term in the *modified equation* increases with smaller k. We will show this next:

The upwind method implements the Forward Euler time stepping; it is given by

$$U_i^{n+1} = U_i^n - \frac{ak}{h}\left(U_i^n - U_{i-1}^n\right). \tag{1}$$

We then insert a new function $v(t,x)$ into this difference equation that agrees exactly with U_i^n at the grid points; i.e., Eq. (1) is satisfied exactly by

$$v(t+k,x) = v(t,x) - \frac{ak}{h}\left[v(t,x) - v(t,x-h)\right]. \tag{2}$$

Taylor-expanding this expression, we get

$$\begin{aligned} v + kv_t + \frac{1}{2}k^2 v_{tt} + \cdots &= v - \frac{ak}{h}\left[v - \left(v - hv_x + \frac{1}{2}h^2 v_{xx} + \cdots\right)\right] \\ v_t + \frac{1}{2}kv_{tt} + \cdots &= -a\left(v_x - \frac{1}{2}hv_{xx} + \cdots\right) \\ v_t + av_x &= \frac{1}{2}\left(ahv_{xx} - kv_{tt}\right) + \cdots \end{aligned} \tag{3}$$

From the derivatives of this resulting expression,

$$\begin{aligned} v_{tt} &= -av_{xt} + \frac{1}{2}\left(ahv_{xxt} - kv_{ttt}\right) + \cdots \\ v_{tx} &= -av_{xx}\frac{1}{2}\left(ahv_{xxx} - kv_{ttx}\right) + \cdots, \end{aligned}$$

we get

$$v_{tt} = a^2 v_{xx} + \cdots$$

We then plug back into Eq. (3) and end up with

$$\begin{aligned} v_t + av_x &= \frac{1}{2}\left(ahv_{xx} - a^2 kv_{xx}\right) + \cdots \\ &= \frac{1}{2}ah\left(1-\nu\right)v_{xx} + \cdots \end{aligned}$$

where $\nu \equiv ak/h$ is the *Courant number*. We know that for stability we require that $0 < \nu < 1$; thus making $k \to 0$ makes the diffusion coefficient (i.e., the coefficient of v_{xx}) larger, which in turn yields that unwanted numerical dissipation we saw on the figure above. On the other hand, taking $k \to 1$ the function v gets closer and closer to satisfying our original advection equation $v_t + av_x = 0$. ♠

Problem 2. *Boundary value problems of the type*

$$-\varepsilon u'' + \beta u' = 0, \quad 0 < x < 1$$
$$u(0) = 0,\ u(1) = 1, \quad \beta, \varepsilon > 0$$

are used to describe problems with diffusion processes with advection (u stands for certain concentration). The term $-\varepsilon u''$ describes diffusion and $\beta u'$ describes advection (or transport). The analytical solution of the problem is given by

$$u(x) = \frac{e^{(\beta/\varepsilon)x} - 1}{e^{\beta/\varepsilon} - 1}. \tag{4}$$

We define the global Péclet number *as* $\mathbb{P}\acute{e}_{gl} = \frac{\beta}{\varepsilon}$.

a) *Examine the behavior of the analytical solution for very small global Péclet number ($\mathbb{P}\acute{e}_{gl} \ll 1$) and very large global Péclet number ($\mathbb{P}\acute{e}_{gl} \gg 1$).* **Hint**: *Taylor expansions.*

b) *Plot the exact solutions for $\varepsilon = 0.1$ and $\beta = 0.1,\ 1.1,\ 2.1,\ 3.1,\ 4.1$ (all in one plot, label the plots by β values (if using MATLAB, use legend).*

c) *Discretize the above boundary value problem by using the second-order centered difference formulas for both derivatives. Use mesh size $h = 1/(m+1)$ (where m is the number of interior grid points) and set up the linear system $\boldsymbol{AU} = \boldsymbol{F}$ for the solution vector $U = [U_0, U_1, \ldots, U_{m+1}]^\top$. Here, U_j is the numerical approximation of u at grid point $x_j = jh$. Specify the entries of the extended matrix A and the right-hand side F of the linear system.*

d) *One can show that the numerical approximation U_j of the solution at grid point x_j satisfies*

$$U_j = \frac{1 - \left(\frac{1+\mathbb{P}\acute{e}}{1-\mathbb{P}\acute{e}}\right)^j}{1 - \left(\frac{1+\mathbb{P}\acute{e}}{1-\mathbb{P}\acute{e}}\right)^{m+1}},\ j = 0, \ldots, m+1 \tag{5}$$

where $\mathbb{P}\acute{e} = (\beta h)/(2\varepsilon)$ denotes the local Péclet number. *Plot the numerical solutions for $\beta = 1$ and $\varepsilon = 0.01$ for $m = 31, 41, 49, 51, 61$ (all in one plot). Label the graphs by the corresponding local Péclet numbers. What do you observe in the plots? Some of these m values correspond to $\mathbb{P}\acute{e} > 1$ and some to $\mathbb{P}\acute{e} < 1$ and you should be able to observe different behaviors for these two cases.*

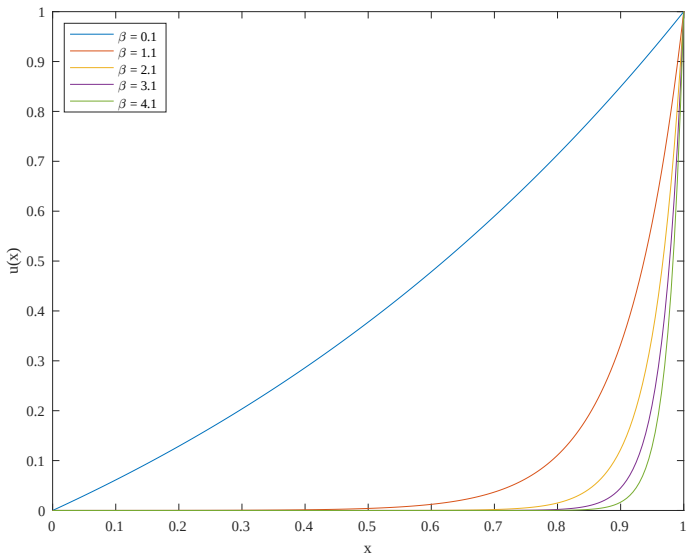
e) *Write a code for part c) and plot the numerical solutions for $\varepsilon = 8 \cdot 10^{-4}$, $\beta = 1$, and $h = 1/200, 1/400, 1/600, 1/800, 1/1000$ (either make separate plots or use subplot to make several plots in one window, don't overlap them).*

f) *Repeat the discretization process from part c), but approximate the first derivative by a one-sided difference approximation*

$$u'(x_j) \approx \frac{U_j - U_{j-1}}{h}$$

instead of the centered differences. Write a code and plot the numerical solutions for $\varepsilon = 8 \cdot 10^{-4}$, $\beta = 1$, and $h = 1/200$, $1/400$, $1/600$, $1/800$, $1/1000$ (either make separate plots or use subplot to make several plots in one window, don't overlap them). Compare your results with those from part e). What do you notice in the numerical solutions?

Solution to a) and b). The following plot shows the analytical solution (4) for the given different values of β:



Firstly, we note that $u(0) = 0$ and $u(1) = 1$. Moreover, for $x \in (0, 1)$, we note that in the limit $\mathbb{P}\acute{e}_{gl} \to \infty$ the denominator in Eq. (4) grows larger than the numerator, since x acts as a damping factor in the numerator. Thus for $\mathbb{P}\acute{e}_{gl} \gg 1$, $u(x) \approx 0$, except as $x \to 1$, in which case $u(x) \to 1$. On the other hand, in the limit $\mathbb{P}\acute{e}_{gl} \to 0$, the function u behaves as $u(x) \approx x$; this would be even more noticeable if we included a smaller β (hence smaller $\mathbb{P}\acute{e}_{gl}$) in the graph above. Let us examine this behavior by Taylor-expanding. We consider a small perturbation $\widehat{\mathbb{P}\acute{e}_{gl}} = \mathbb{P}\acute{e}_{gl} + \delta\mathbb{P}\acute{e}_{gl}$, so that

$$u(\widehat{\mathbb{P}\acute{e}_{gl}}) = \frac{e^{\widehat{\mathbb{P}\acute{e}_{gl}}x} - 1}{e^{\widehat{\mathbb{P}\acute{e}_{gl}}} - 1} \approx \frac{\left(e^{\mathbb{P}\acute{e}_{gl}x} + xe^{\mathbb{P}\acute{e}_{gl}x}\delta\mathbb{P}\acute{e}_{gl} + O(\delta\mathbb{P}\acute{e}_{gl}^2)\right) - 1}{\left(e^{\mathbb{P}\acute{e}_{gl}} + e^{\mathbb{P}\acute{e}_{gl}}\delta\mathbb{P}\acute{e}_{gl} + O(\delta\mathbb{P}\acute{e}_{gl}^2)\right) - 1}.$$

Dropping higher order terms and considering the limit $\mathbb{P}\acute{e}_{gl} \to 0$,

$$\lim_{\mathbb{P}\acute{e}_{gl}\to 0} \frac{e^{\mathbb{P}\acute{e}_{gl}x} + xe^{\mathbb{P}\acute{e}_{gl}x}\delta\mathbb{P}\acute{e}_{gl} - 1}{e^{\mathbb{P}\acute{e}_{gl}} + e^{\mathbb{P}\acute{e}_{gl}}\delta\mathbb{P}\acute{e}_{gl} - 1} = \frac{1 + x\delta\mathbb{P}\acute{e}_{gl} - 1}{1 + \delta\mathbb{P}\acute{e}_{gl} - 1} = \frac{x\delta\mathbb{P}\acute{e}_{gl}}{\delta\mathbb{P}\acute{e}_{gl}} = x. \qquad \surd \qquad \spadesuit$$

Solution to c) and e). The centered discretization of the BVP is given by

$$-\varepsilon\left(\frac{U_{j-1} - 2U_j + U_{j+1}}{h^2}\right) + \beta\left(\frac{U_{j+1} - U_{j-1}}{2h}\right) = 0 \qquad j = 1, \ldots, m.$$

Multiplying this equation by h^2/ε and using the local Péclet number $\mathbb{P}\acute{e} = (\beta h)/(2\varepsilon)$, we get

$$\begin{aligned} -\left(U_{j-1} - 2U_j + U_{j+1}\right) + \mathbb{P}\acute{e}\left(U_{j+1} - U_{j-1}\right) &= 0 \\ -\left(\mathbb{P}\acute{e} + 1\right)U_{j-1} + 2U_j + \left(\mathbb{P}\acute{e} - 1\right)U_{j+1} &= 0. \end{aligned}$$

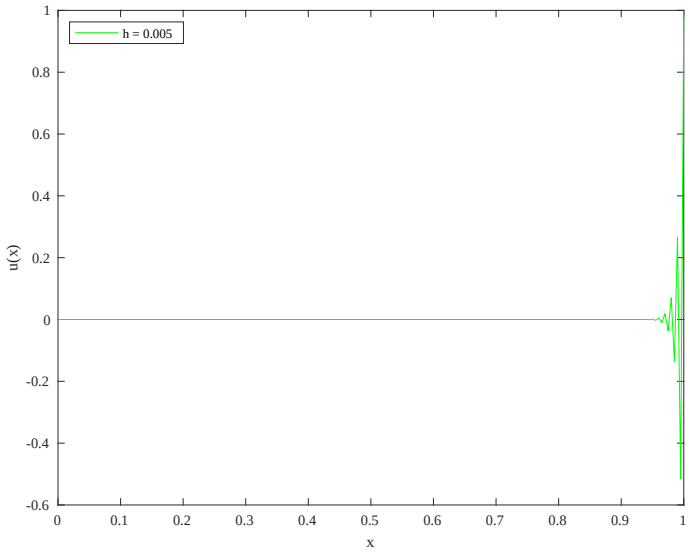
In matrix form, we get the linear system $\boldsymbol{AU} = \boldsymbol{F}$:

$$\underbrace{\begin{bmatrix} 2 & \mathbb{P}\acute{e}-1 & & & & & \\ -\mathbb{P}\acute{e}-1 & 2 & \mathbb{P}\acute{e}-1 & & & & \\ & -\mathbb{P}\acute{e}-1 & 2 & \mathbb{P}\acute{e}-1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\mathbb{P}\acute{e}-1 & 2 & \mathbb{P}\acute{e}-1 & \\ & & & & -\mathbb{P}\acute{e}-1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_j \\ \vdots \\ U_m \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 1 - \mathbb{P}\acute{e} \end{bmatrix}}_{F}, \tag{6}$$

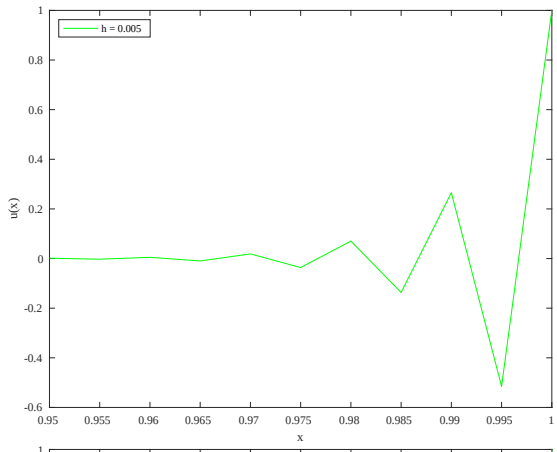
where the last entry of $\boldsymbol{F}$ came from the right boundary condition $U_{m+1} = u(1) = 1$. The Matlab code that solves this system is given here:

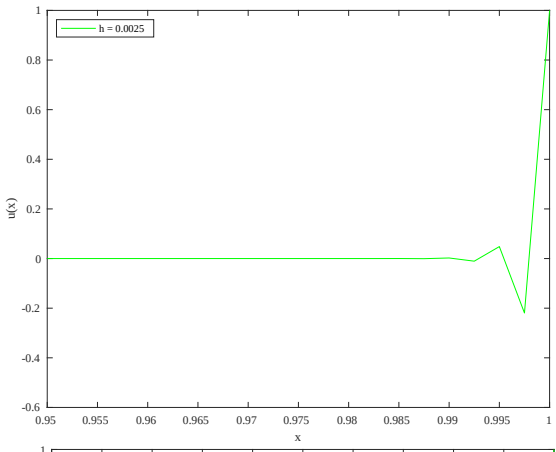
```
function Peclet_2e(m)

    a = 0;
    b = 1;
    e = 8e-04;              %\epsilon
    bet = 1;                %\beta
    h = (b-a)/(m+1);        %h=\Delta x
    P = (bet*h)/(2*e);      %local Peclet number

    %Generate arrays:
    A = zeros(m);
    F = zeros(m,1);
    F(m) = 1 - P;             %assign only nonzero entry to F

    %fill up A (could also use sparse allocation)
    for i = 1:m
        A(i,i)  = 2;
        if i ~= m
            A(i,i+1)  =   P-1;
            A(i+1,i)  =  -P-1;
        end
    end

    U = A\F;          %solve the system
    U = [0;U;1];     %complete solution vector

    %Plot results:
    x = linspace(a,b,m+2)';
    plot(x,U, "g-")
    ylabel('u(x)')
    xlabel('x')
    legend(['h = ' , num2str(h)], 'Location','northwest')
    exportgraphics(gcf,'Peclet_2e_m199.pdf')
end
```

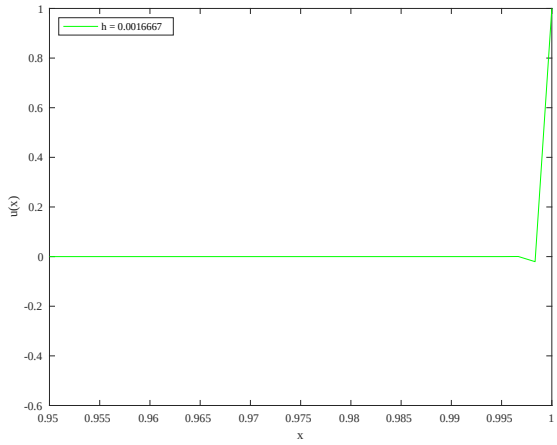
We run this code for different values of m corresponding to the h-values provided in the exercise; the results are shown in the following plots. We will see that for the lower resolutions (larger h) we see some oscillatory behavior near the right boundary $x = 1$; for instance:

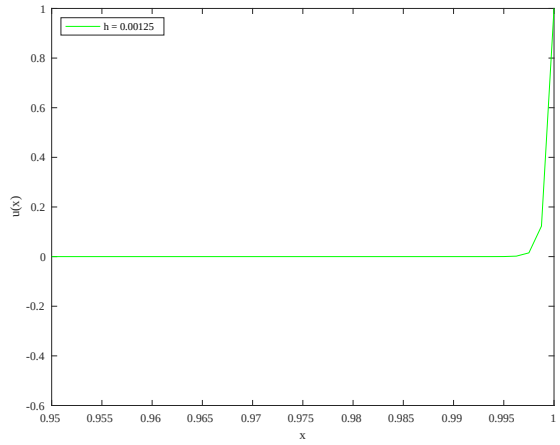


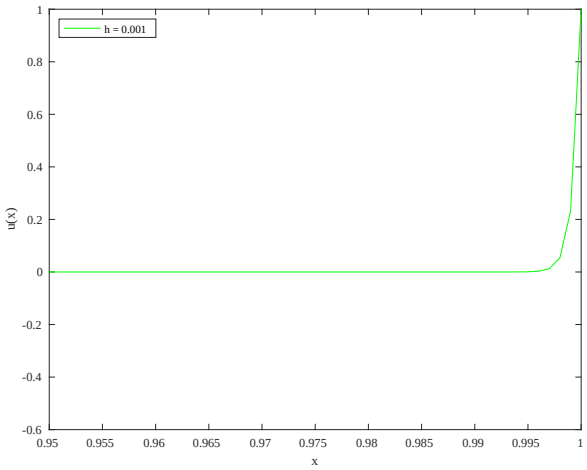
Since the function vanishes identically almost everywhere in $(0, 1)$, except near and at the boundary $x = 1$, we will zoom in to take a closer look at the behavior there (there is nothing really interesting happening away from $x = 1$ anyway):











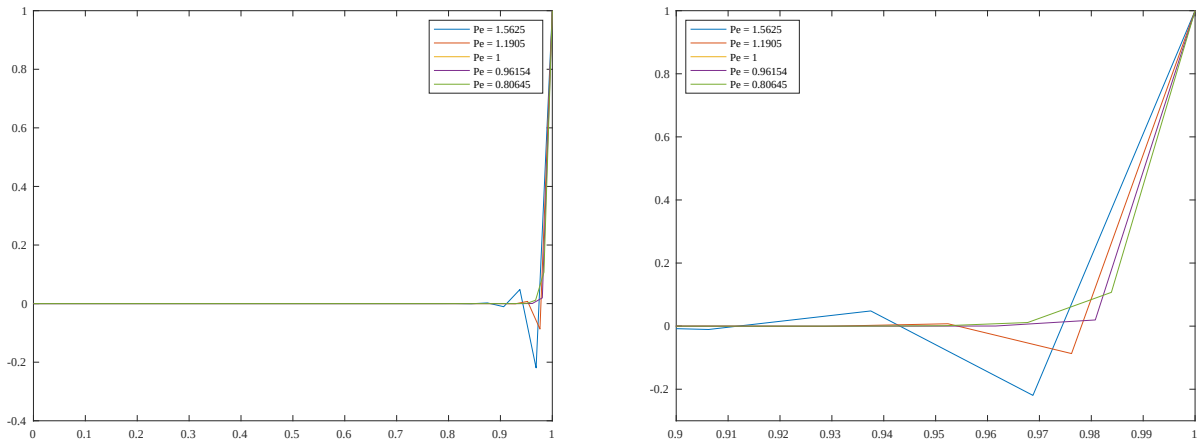
♠

Solution to d). The following Matlab code shows the numerical approximation given by Eq. (5) for the given values of m:

```
a   = 0;
b   = 1;
bet = 1;        %\beta
e = 0.01;       %\epsilon
M = [31,41,49,51,61];

for m = M

```

```
    U = zeros(m+2,1);          %Initilalize solution vector
    h = (b-a)/(m+1);

    for j = 1:m+2
        P = (bet*h)/(2*e);     %local Peclet number
        P_f = ( 1+P)/(1-P);
        U(j) = (1-(P_f)^(j-1))/(1-(P_f)^(m+1));
    end

    %save all solution vectors
    if m == M(1)
        x1 = linspace(a,b,m+2)';
        U1 = U;
        P1 = P;
    elseif m==M(2)
        x2 = linspace(a,b,m+2)';
        U2 = U;
        P2 = P;
    elseif m==M(3)
        x3 = linspace(a,b,m+2)';
        U3 = U;
        P3 = P;
    elseif m==M(4)
        x4 = linspace(a,b,m+2)';
        U4 = U;
        P4 = P;
    elseif m==M(5)
        x5 = linspace(a,b,m+2)';
        U5 = U;
        P5 = P;
    end
end

clf
plot(x1,U1, 'DisplayName', ['Pe = ', num2str(P1)])
hold on
plot(x2,U2, 'DisplayName', ['Pe = ', num2str(P2)])
hold on
plot(x3,U3, 'DisplayName', ['Pe = ', num2str(P3)])
hold on
plot(x4,U4, 'DisplayName', ['Pe = ', num2str(P4)])
hold on
plot(x5,U5, 'DisplayName', ['Pe = ', num2str(P5)])
hold off

exportgraphics(gcf,'Peclet_diff_m.pdf')
```

Here is the resulting plot (for clarity, we also include a zoom-in on the right, as we did on the previous exercise):



We can see that for the cases where $\mathbb{P}\text{é} > 1$ the numerical solution dips below 0 as we approach the right ($x = 1$) boundary. In fact, we see some oscillatory behavior that is quite noticeable for the largest $\mathbb{P}\text{é}$ value. On the other hand, the $\mathbb{P}\text{é} < 1$ values show a more stable behavior. ♠

Solution to f). This time we have the discretization

$$-\varepsilon\left(\frac{U_{j-1}-2U_j+U_{j+1}}{h^2}\right)+\beta\left(\frac{U_j-U_{j-1}}{h}\right)=0\, j=1,\ldots,m.$$

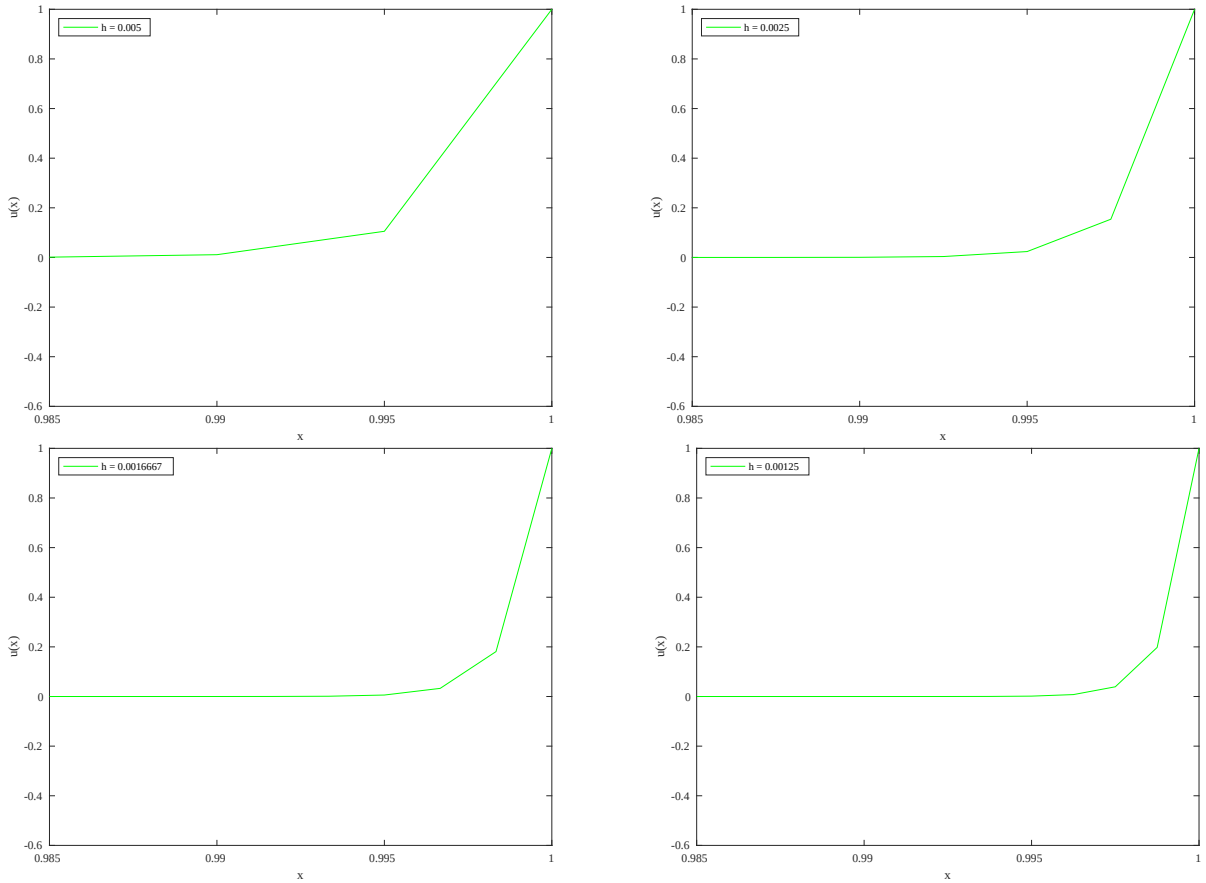
As before, we multiply this equation by h^2/ε and use the local Péclet number:

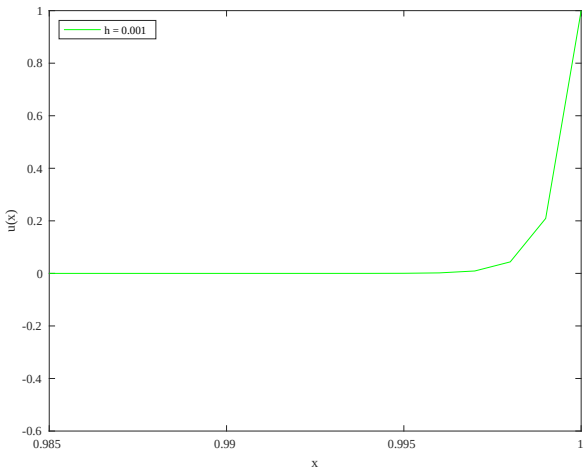
$$\begin{aligned}-\left(U_{j-1}-2U_j+U_{j+1}\right)+2\mathbb{P}\text{é}\left(U_j-U_{j-1}\right)&=0\\ -\left(2\mathbb{P}\text{é}+1\right)U_{j-1}+2\left(\mathbb{P}\text{é}+1\right)U_j-U_{j+1}&=0.\end{aligned}$$

In matrix form, we get the linear system $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{U}}=\tilde{\boldsymbol{F}}$:

$$\underbrace{\begin{bmatrix} 2\mathbb{P}\text{é}+2 & -1 & & & & & \\ -2\mathbb{P}\text{é}-1 & 2\mathbb{P}\text{é}+2 & -1 & & & & \\ & -2\mathbb{P}\text{é}-1 & 2\mathbb{P}\text{é}+2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -2\mathbb{P}\text{é}-1 & 2\mathbb{P}\text{é}+2 & -1 & \\ & & & & -2\mathbb{P}\text{é}-1 & 2\mathbb{P}\text{é}+2 & \end{bmatrix}}_{\tilde{A}}\underbrace{\begin{bmatrix}U_1\\U_2\\\vdots\\U_j\\\vdots\\U_m\end{bmatrix}}_{\tilde{U}}=\underbrace{\begin{bmatrix}0\\0\\\vdots\\\vdots\\0\\1\end{bmatrix}}_{\tilde{F}}, \tag{7}$$

where the last entry of $\tilde{\boldsymbol{F}}$ came from the right boundary condition $U_{m+1}=u(1)=1$. The code to implement this method is very similar to the one we showed on the solution to parts c) and e), so there is no need to repeat it here. We simply show the results (once again zooming in near the $x=1$ boundary):





As we can see from the figures, using the one-sided discretization for the first derivative turned out to give us better results than the ones we got earlier using the centered scheme. Even for the lower resolution (larger h) this one-sided scheme is well-behaved; we certainly do not see the oscillatory behavior we saw earlier with the centered approach. ♠

Problem 3. *Consider the upwind finite difference method*

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x} \begin{cases} f(U_j^n) - f(U_{j-1}^n) & \partial f/\partial u \geq 0 \\ f(U_{j+1}^n - f(U_j^n) & \partial f/\partial u < 0 \end{cases},$$

for solving the advection equation $u_t + (f(u))_x = 0$.

a) Solve the equation numerically, using $f(u) = u^2/2$, *on the period domain* $0 \leq x \leq 2$ *with initial condition*

$$u(x,0) = \begin{cases} 1 - \cos(2\pi x), & 0 \leq x \leq 1 \\ 0, & \textit{elsewhere} \end{cases}.$$

Let $\Delta x = 1/100$ *and* $\Delta t = \Delta x/2$. *Run your simulation until* $t = 0.5$. *Make a movie from the solution plots.*

b) Comment on your results. What is happening to the amplitude of the wave? Do you think the method is correct?

Solution to a) and b). The following code solves the equation $u_t + (f(u))_x = 0$ numerically. Note that we only make use of

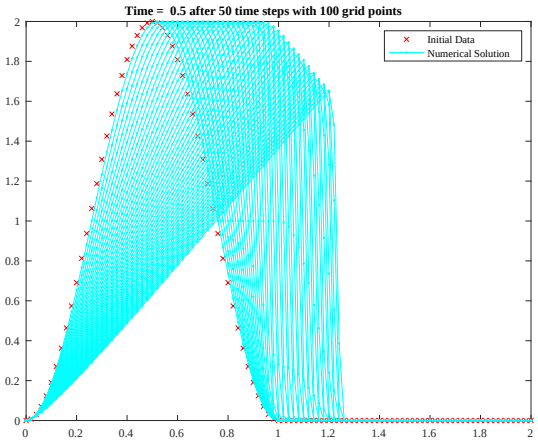
$$U_j^{n+1} = U_j^n - \frac{k}{h}\left[f(U_j^n) - f(U_{j-1}^n)\right],$$

since $\partial f/\partial u = u \geq 0 \ \forall x \in [0,2]$ initially, and it remains so for the remainder of the time-evolution.

```
function advection_up_Prob3(m)

global a
a = 1;      %advection velocity
clf         %clear graphics

ax = 0;
bx = 2;
tfinal = 0.5;              %final time

h = (bx-ax)/(m+1);         %h = delta x    (m=99 for this exercise)
k = 0.5*h;                 %time step
nu = a*k/h;                %Courant number
x = linspace(ax,bx,m+2)';

I = 2:(m+2);               %indices of unknowns
```

```
17
18 nsteps = round(tfinal / k);    %number of time steps
19
20 if abs(k*nsteps - tfinal) > 1e-5
21    % The last step won't go exactly to tfinal.
22    disp(' ')
23    display(['  WARNING *** k does not divide tfinal, k = ', num2str(k)])
24    disp(' ')
25 end
26
27 %initial conditions:
28 tn = 0;
29 u0 = eta(x);
30 u = u0;
31
32 % periodic boundary conditions:
33 u(1) = u(m+2);   % copy value from rightmost unknown to ghost cell on left
34 u(m+3) = u(2);   % copy value from leftmost unknown to ghost cell on right
35
36 %plot initial data:
37 plot(x,u0,'rx')
38 hold on
39
40 %create movie for the plot from t_0 to t_final
41 v = VideoWriter('Upwind_movie.avi');
42 open(v);
43
44 % main time-stepping loop:
45 for n = 1:nsteps
46      tnp = tn + k;   %= t_{n+1}
47
48      % Upwind:
49      u(I) = u(I) - nu*(f(u(I)) - f(u(I-1))) ;
50
51      % periodic boundary conditions:
52      u(1) = u(m+2);   % copy value from rightmost unknown to ghost cell on left
53      u(m+3) = u(2);   % copy value from leftmost unknown to ghost cell on right
54
55       uint = u(1:m+2);      %points on the interval (drop ghost cell on right)
56
57       plot(x,uint, 'c.-');
58       title(['Time =  ', num2str(tnp,6), ' after ', ...
59       num2str(n), ' time steps with ', num2str(m+1), ' grid points']);
60       legend('Initial Data', 'Numerical Solution')
61       axis
62       pause(0.1);
63       frame = getframe(gcf);
64       writeVideo(v,frame);
65
66      tn = tnp;   %update for next time step
67 end
68
69 exportgraphics(gcf,'UW_Prob3.pdf');
70 close(v);
71
72
73 function f = f(x)
74         f = 0.5 .* x.^2;
75 return
76
77
78 function eta = eta(x)                %Initial data
79
80     eta = zeros(length(x), 1);
81     for i = 1 : length(x)
82         if ( (x(i) >= 0) && (x(i) <= 1) )
83             eta(i) = 1 - cos(2*pi.*x(i));
84         end
85     end
86 return
```

The following plot shows the output (.avi file will also be uploaded to MyCourses):



We note that the wave amplitude starts to decrease after some time. This is to be expected, since –as we saw on Problem 1c)– the upwind method when applied to the advection equation introduces a diffusion term that smears out the numerical solution. This particular problem shows the same behavior as we saw on Problem 1c): the upwind method transports the solution to the right with constant velocity a (= 1 in this case), but in the process it introduces numerical dissipation that affects the wave amplitude of the Gaussian peak. ♠