
Systems Lab: Systems of ODEs in MATLAB

Table of Contents

Student Information	1
Exercise 1	1
Exercise 2	2
Exercise 3	3
Saving Images in MATLAB	5
Exercise 4	5

In this lab, you will write your own ODE system solver for the Heun method (aka the Improved Euler method), and compare its results to those of `ode45`.

You will also learn how to save images in MATLAB.

Opening the m-file `lab4.m` in the MATLAB editor, step through each part using cell mode to see the results. Compare the output with the PDF, which was generated from this m-file.

There are four (4) exercises in this lab that are to be handed in on the due date of the lab. Write your solutions in the template, including appropriate descriptions in each step. Save the m-files and the pdf-file for Exercise 4 and submit them on Quercus.

Student Information

Student Name: Hshmat Sahak

Student Number: 1005903710

Exercise 1

Objective: Write your own ODE system solver using the Heun/Improved Euler Method and compare it to `ode45`.

Details: Consider the system of 2 ODEs:

$$x_1' = f(t, x_1, x_2), \quad x_2' = g(t, x_1, x_2)$$

This m-file should be a function which accepts as variables (t_0, t_N, x_0, h) , where t_0 and t_N are the start and end points of the interval on which to solve the ODE, h is the stepsize, and x_0 is a vector for the initial condition of the system of ODEs $x(t_0) = x_0$. Name the function `solvesystem_<UTORid>.m` (Substitute your UTORid for [UTORid](#)). You may also want to pass the functions into the ODE the way `ode45` does (check MATLAB labs 2 and 3).

Your m-file should return a row vector of times and a matrix of approximate solution values (the first row has the approximation for x_1 and the second row has the approximation for x_2).

Note: you will need to use a loop to do this exercise. You will also need to recall the Heun/Improved Euler algorithm learned in lectures.

```
% code in solvesystem_sahakhsh.m
% function x = solvesystem_sahakhsh(f,g,t0,tN,x0,h)
% numsteps = floor((tN-t0)/h);
% xvals = zeros(2, numsteps+1);
% xvals(1,1) = x0(1);
% xvals(2,1) = x0(2);
% for i = 1:numsteps
%     init_time = t0 + h*(i-1);
%     init_x = xvals(:,i);
%     end_time = t0 + h*i;
%     slope1x1 = f(init_time, init_x(1), init_x(2));
%     slope1x2 = g(init_time, init_x(1), init_x(2));
%     end_x = init_x + [slope1x1*h; slope1x2*h];
%     slope2x1 = f(end_time, end_x(1), end_x(2));
%     slope2x2 = g(end_time, end_x(1), end_x(2));
%     slopeavg = [(slope1x1+slope2x1)/2; (slope1x2+slope2x2)/2];
%     xvals(:,i+1) = xvals(:,i) + slopeavg*h;
% end
% times = linspace(t0, tN, numsteps+1);
% x = struct('t', times, 'x', xvals);
```

Exercise 2

Objective: Compare Heun with an exact solution

Details: Consider the system of ODEs

$$x_1' = x_1/2 - 2x_2, \quad x_2' = 5x_1 - x_2$$

with initial condition $x(0) = (1, 1)$.

Use your method from Exercise 1 to approximate the solution from $t=0$ to $t=4\pi$ with step size $h=0.05$.

Compute the exact solution (by hand) and plot both phase portraits on the same figure for comparison.

Your submission should show the construction of the inline function, the use of your Heun's method to obtain the solution, a construction of the exact solution, and a plot showing both. In the comments, include the exact solution.

Label your axes and include a legend.

```
% exact solutions:
% x1 = (1/151)*e^(-
t/4)*(151*cos(sqrt(151)*t/4)-5*sqrt(151)*sin(sqrt(151)*t/4)
% x2 = (1/151)*e^(-
t/4)*(151*cos(sqrt(151)*t/4)+17*sqrt(151)*sin(sqrt(151)*t/4))

% Inline function
f = @(t, x1, x2) x1/2 - 2*x2;
g = @(t, x1, x2) 5*x1 - x2;
t0 = 0;
tN = 4*pi;
```

```

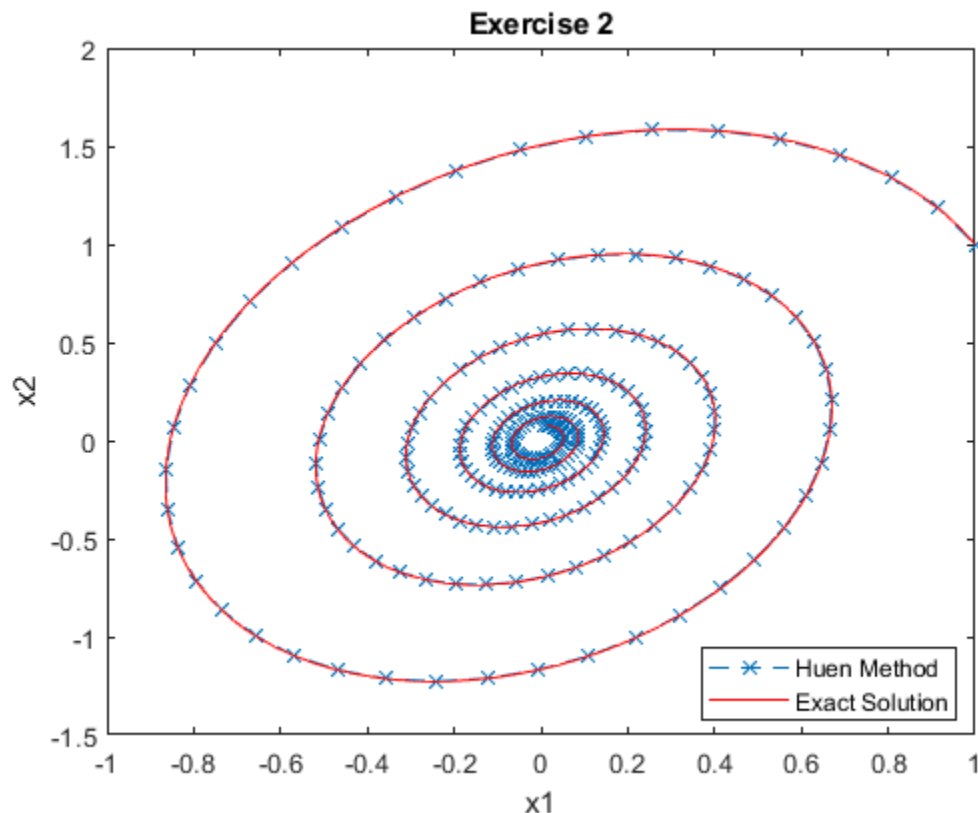
h = 0.05;
x0 = [1,1];

% Huen method
approximation = solvesystem_sahakhsh(f,g,t0,tN,x0,h);

% Analytic Solution
x2_exact = @(t) exp(-
t/4).*((3*cos(t*sqrt(151)/4)+sqrt(151)*sin(t*sqrt(151)/4))/8
+ (-sqrt(151)*cos(t*sqrt(151)/4)+3*sin(t*sqrt(151)/4))*(-5)/
(8*sqrt(151)));
x1_exact = @(t) exp(-t/4).*(cos(t*sqrt(151)/4) - 5*sin(t*sqrt(151)/4)/
sqrt(151));
t = linspace(0,4*pi,600);

% Plot both
plot(approximation.x(1,:), approximation.x(2,:), 'x--', x1_exact(t),
x2_exact(t), 'r');
title('Exercise 2');
xlabel('x1');
ylabel('x2');
legend('Huen Method', 'Exact Solution', 'Location', 'Best');

```



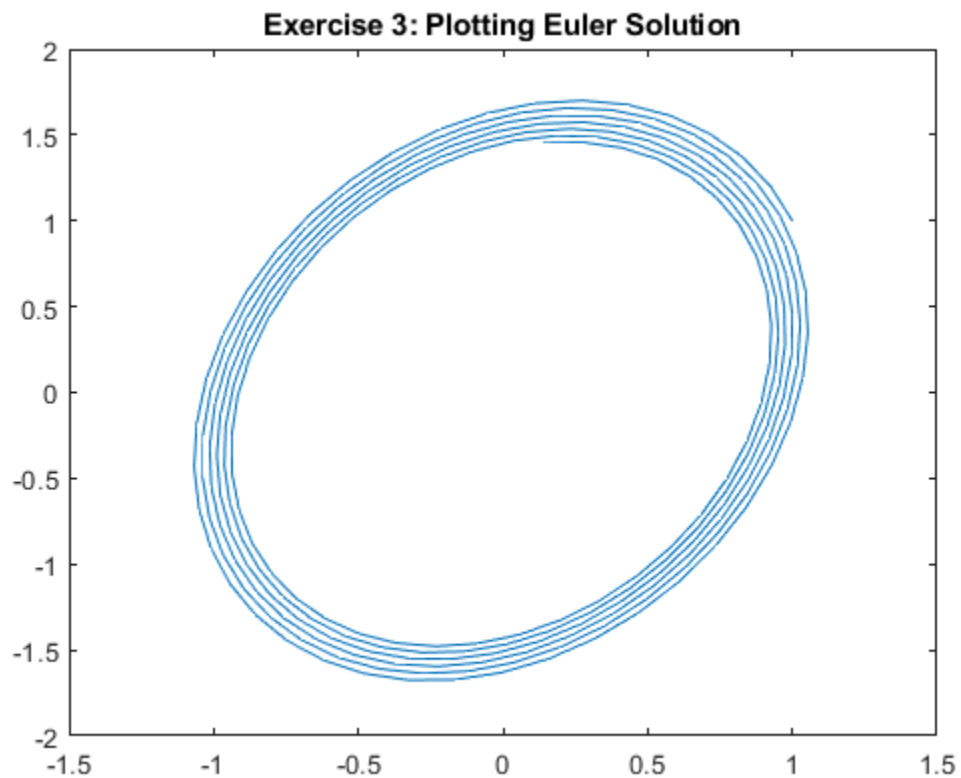
Exercise 3

Objective: Compare your method with Euler's Method (from iode).

Details: Use `ode` to plot the solution for the same problem with the same step size as on Exercise 2.

Compare your solution on exercise 2, the exact solution from exercise 2 and the approximation using Euler's method. Plot the solution for Euler's method and make note of any differences.

```
t=0:0.05:4*pi;
x0 = [1;1];
f=@(t,x) [x(1)/2-2*x(2); 5*x(1)-x(2)];
x=euler(f,x0,t);
plot(x(1,:),x(2,:))
title('Exercise 3: Plotting Euler Solution');
% The improved euler's method and plot from exact solution appear to
% be
% spiraling inwards. (0,0) appears as if it is a spiral sink. However,
% in
% euler's method, we dont seem to spiral in as fast. The radius from
% (0,0)
% appears to remain relatively constant. If we were only to use
% euler's method, we
% may be lead to beleve that the solution is really a closed path, and
% that
% deviations arise from the approximation.
% This occurs because Euler's method is a lower order method!
% Experimenting
% with ode shows that euler's method could approach the exact
% solution
% with smaller time steps
```



Saving Images in MATLAB

To do the following exercises, you will need to know how to output graphics from MATLAB. Create a folder on your Desktop (or elsewhere) to contain the files generated by these exercises. Make this folder the "Current Folder" in the left side of the main MATLAB window. This will ensure that the files output by MATLAB end up in the folder you created.

To save an image of a phase portrait, use the following steps:

1. Get the phase portrait looking the way you want in the `ode` window.
2. Leaving `ode` open, switch to the main MATLAB window.
3. Type the command `print -dpng -r300 'filename.png'` in the command window.

This command will create a PNG graphic called `filename.png` in the current folder. The `-dpng` option tells MATLAB to output the graphic in PNG format; MATLAB also allows output in other formats, such as BMP, EPS, PNG and SVG. The `-r300` option tells MATLAB to set the resolution at 300 dots per inch and can be adjusted if you wish.

Exercise 4

Objective: Analyze phase portraits.

Details: Compile the results of the following exercises into a single document (e.g. using a word processor) and export it to PDF for submission on Quercus.

For each of the first-order systems of ODEs 4.1 to 4.10 below, do the following exercises:

- (a) Generate a phase portrait for the system (centre the graph on the equilibrium point at (0,0)). Include a few trajectories.
- (b) Classify the equilibrium on asymptotic stability, and behaviour (sink, source, saddle-point, spiral, center, proper node, improper node) - check table 3.5.1 and figure 3.5.7. Classify also as for clockwise or counterclockwise movement, when relevant.
- (c) Compute the eigenvalues of the matrix (you do not need to show your calculations). Using the eigenvalues you computed, justify part (b).

To avoid numerical error, you should use Runge-Kutta solver with a step size of 0.05. Change the display parameters, if necessary, to best understand the phase portrait.

4.1. $\frac{dx}{dt} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} x$ eigenvalues: $(5+\sqrt{5})/2$, $(5-\sqrt{5})/2$ nodal source unstable

4.2. $\frac{dx}{dt} = \begin{bmatrix} -2 & -1 \\ -1 & -3 \end{bmatrix} x$ eigenvalues: $(-5+\sqrt{5})/2$, $(-5-\sqrt{5})/2$ nodal sink asymptotically stable

4.3. $\frac{dx}{dt} = \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix} x$ eigenvalues: 2, -1 saddle point unstable orientation:

4.4. $\frac{dx}{dt} = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix} x$ eigenvalues: 1, -2 saddle unstable orientation:

4.5. $\frac{dx}{dt} = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} x$ eigenvalues: $(-1+i\sqrt{3})/2$, $(-1-i\sqrt{3})/2$ spiral sink asymptotically stable counterclockwise

4.6. $\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} x$ eigenvalues: $(1+i\sqrt{3})/2$, $(1-i\sqrt{3})/2$ spiral source unstable clockwise

4.7. $\frac{dx}{dt} = \begin{bmatrix} 2 & 8 \\ -1 & -2 \end{bmatrix} x$ eigenvalues: $2i, -2i$ centre at $(0,0)$ stable (not asymptotically) clockwise

4.8. $\frac{dx}{dt} = \begin{bmatrix} -2 & -8 \\ 1 & 2 \end{bmatrix} x$ eigenvalues: $2i, -2i$ centre at $(0,0)$ stable (not asymptotically) counterclockwise

4.9. $\frac{dx}{dt} = \begin{bmatrix} -8 & 5 \\ -13 & 8 \end{bmatrix} x$ eigenvalues: $i, -i$ centre at $(0,0)$ stable (not asymptotically) clockwise

4.10. $\frac{dx}{dt} = \begin{bmatrix} 8 & -5 \\ 13 & -8 \end{bmatrix} x$ eigenvalues: $i, -i$ centre at $(0,0)$ stable (not asymptotically) counterclockwise

Published with MATLAB® R2020a