Lab 3 - Joseph Riley Guest - MAT 275 Lab

Table of Contents

Introduction to Numerical Methods for Solving ODEs

Exercise 1

Part (a)

```
clc                        % clearing command window
f=@(t,y) 3*y;              % Defining ODE function f for this lab
t=linspace(0,0.6,100);     % Defining vector t of time values
y=exp(3*t);                % Solution of the the function f
Nsmall = 6;Nmed = 60;Nlarge = 600;Nhuge = 6000; % declaring the
 different step increments
[t6, y6]=euler(f,[0,0.6],1,Nsmall);  %solve the ODE using Euler w/ 6
 steps
[t60,y60]=euler(f,[0,0.6],1,Nmed);        %solve the ODE using Euler
 w/ 60 steps
[t600,y600]=euler(f,[0,0.6],1,Nlarge);  %solve the ODE using Euler w/
 600 steps
[t6000,y6000]=euler(f,[0,0.6],1,Nhuge);     %solve the ODE using Euler
 w/ 6000 steps
e6 = y(end) - y6(end)       % error when N = 6
e60 = y(end) - y60(end)        % error when N = 60
e600 = y(end) - y600(end)      % error when N = 600
e6000 = y(end) - y6000(end)      % error when N = 6000
ratio60 = e6/e60;                 % ratio of Nsmall/Nmed
ratio600 = e60/e600;              % ratio of Nmed/Nlarge
ratio6000 = e600/e6000;           % ratio of Nlarge/Nhuge
% Giving the approximations of Nsmall thorugh Nhuge.
Nsmall(end);Nmed(end);Nlarge(end);Nhuge(end);
disp('---------------------------------------------------')
disp('|     N      |  approximation  |  error     | ratio  |')
disp('|------------|-----------------|------------|--------|')
disp('| 6          |     4.8268      |  1.2228    |  N/A   |')
disp('| 60         |     5.8916      |  0.1580    | 7.7373 |')
```

```
disp('| 600        |    6.0334        |  0.0163   | 9.7082 |')
disp('| 6000       |    6.0480        |  0.0016   | 9.9699 |')
disp('----------------------------------------------------')
```

```
e6 =

    1.2228


e60 =

    0.1580


e600 =

    0.0163


e6000 =

    0.0016

---------------------------------------------------
|     N      |  approximation   |  error    | ratio  |
|-----------|-----------------|----------|--------|
| 6          |    4.8268        |  1.2228  |  N/A   |
| 60         |    5.8916        |  0.1580  | 7.7373 |
| 600        |    6.0334        |  0.0163  | 9.7082 |
| 6000       |    6.0480        |  0.0016  | 9.9699 |
---------------------------------------------------
```

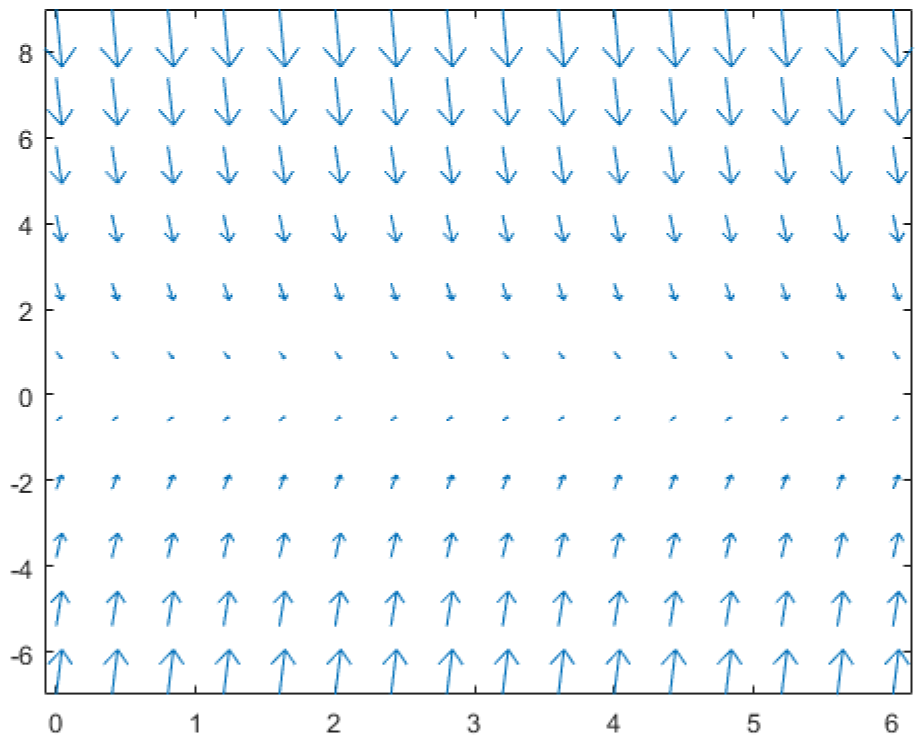
Part b.) If you look at the column titled error, the ratio is roughly ten. So as the number of N steps increases by a factor of 10, the error decreases by a factor of ten.

Part c.) The tangent line falls below the curve, so this means it is underestimating the answer for Euler's method

Exercise 2

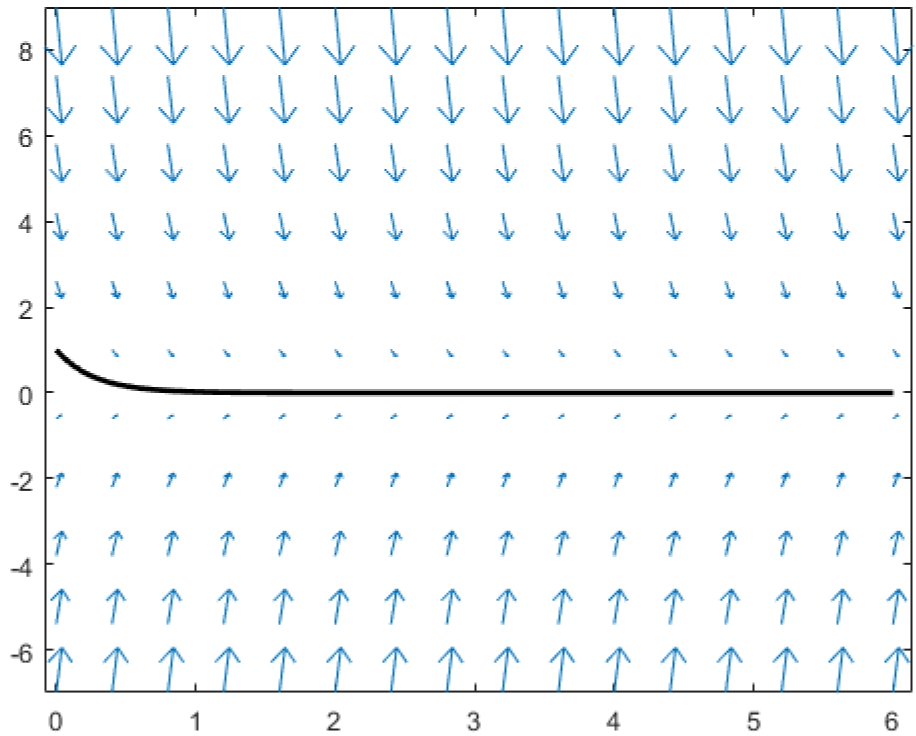
Part (a)

```
clc % clearing command window
t = 0:0.4:6; y = -7:1.6:9; % define a grid in t & y directions
[T,Y] = meshgrid (t,y); % create 2d matrices of points in ty - plane
dT = ones ( size (T)); %dt =1 for all points
dY = -3.7*Y; %dy = -3.7*y; this is the ODE
quiver (T,Y,dT ,dY) % draw arrows (t,y)->(t+dt , t+dy)
axis tight % adjust look
hold on % holding all plots on this figure until holding off
```



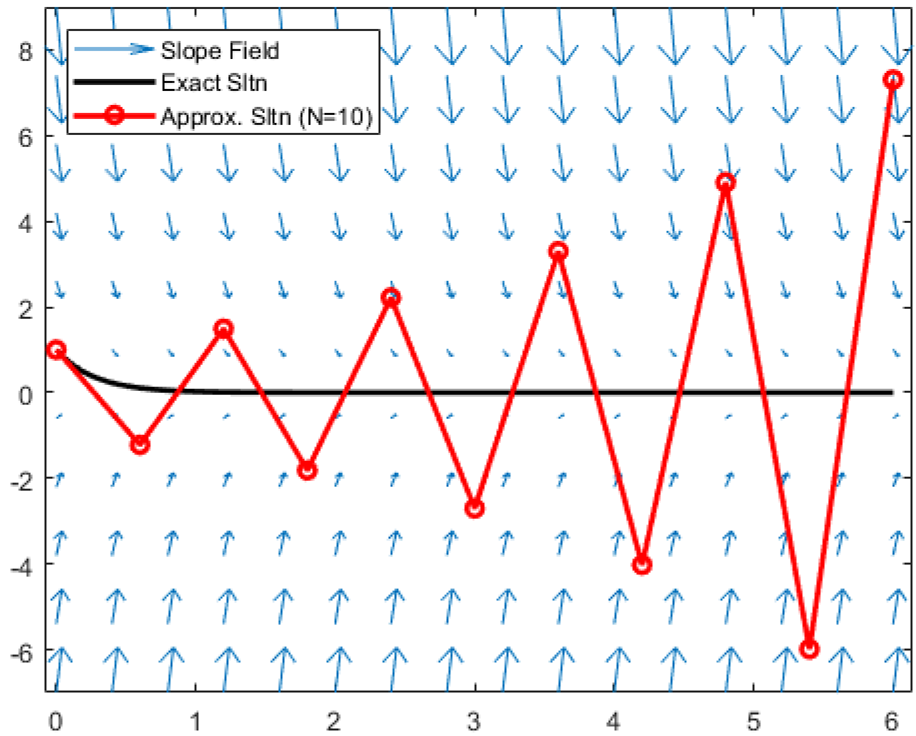
Part (b)

```
t = linspace(0, 6, 100); % defining vector t of given interval for
 exact solution
y = exp(-3.7.*t); % defining the exact solution to dydt
plot(t, y, 'k', "Linewidth",2) % plotting analytical solution vector
 with slopefield
```



Part (c)

```
f = @(t,y) -3.7*y; % defining the ODE as an anonymous function
[t10, y10] = euler(f, [0,6], 1, 10); % Solving ODE with Euler's method
 using 10 steps
plot(t10, y10, 'ro-', 'linewidth', 2) % Plotting the approx solution
 N=10
legend('Slope Field','Exact Sltn','Approx. Sltn
 (N=10)','location','northwest')
hold off;  % end plotting in this figure window
```



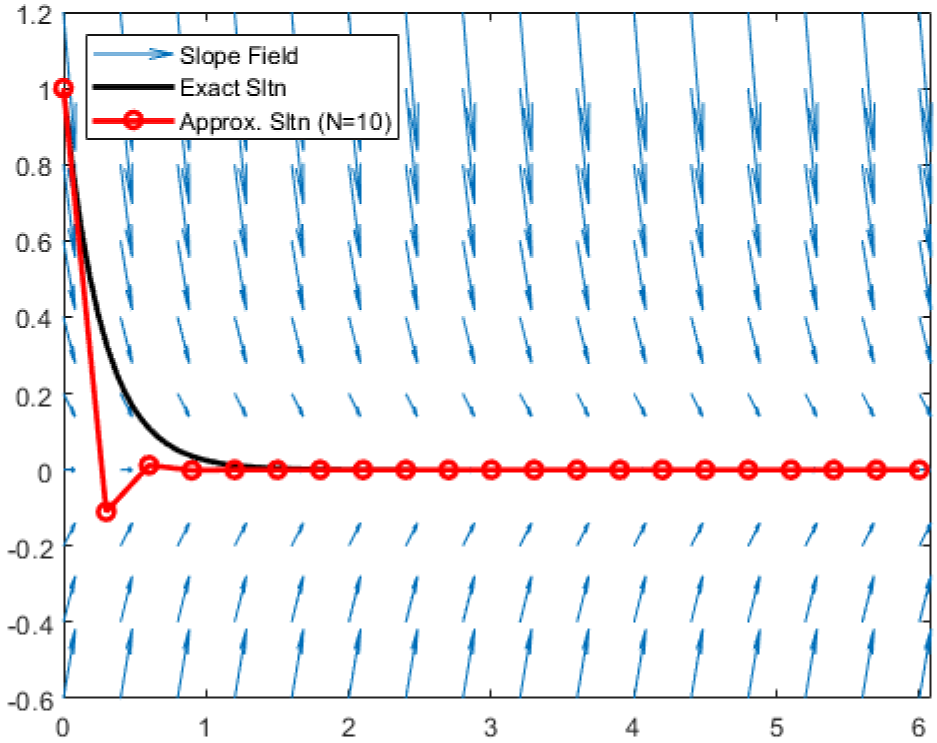
Essay Question

It's obvious that the more steps you have, the better the approximation. What makes this so inaccurate is that the coefficient of the solution is negative. Which in turn makes the solution unstable.

Part (d)

```
figure; % calling a new figure
t = 0:0.4:6; % defining the t vector (time values)
y = -0.6:0.2:1.3; % defining the y vector
[T,Y] = meshgrid (t,y); % defining new grid of t and y values for
 plotting
dT = ones ( size (T)); % dt = 1 for all points
dY = -3.7*Y; % dy = -3.7*y; this is the ODE
quiver (T,Y,dT ,dY) % draw arrows (t,y)->(t+dt , t+dy)
axis tight % adjust look
hold on; %holding all plots on this figure
t = linspace(0, 6, 100); % defining t vector for exact solution
y = exp(-3.7.*t); % defining the exact solution to dydt
plot(t, y, 'k', "Linewidth",2) % plotting solution vector
f = @(t,y) -3.7*y; % defining the ODE as an anonymous function
[t20, y20] = euler(f, [0,6], 1, 20); % Solving ODE with Euler's using
 20 steps
plot(t20, y20, 'ro-', 'linewidth', 2) % Plotting the approx solution
 N=10
```

```
legend('Slope Field','Exact Sltn','Approx. Sltn
 (N=10)','location','northwest')
hold off;  % end plotting in this figure window
```



With the stepsize at N = 20, it would make this solution stable.

Exercise 3

```
type 'impeuler.m'; % dislpaying impeuler.m
f = @(t,y) 3*y; % defining the ODE as an anonymous function
[t6 ,y6] = impeuler(f ,[0 ,0.6] ,1 ,6); %solve the ODE using ImpEuler
 w/ 6 steps
[t6 ,y6] % diplaying the outputs of the t and y vector at Nsmall = 6
```

```
 function [t,y] = impeuler(f,tspan,y0,N)

    % Solves the IVP y' = f(t,y), y(t0) = y0 in the time interval
 tspan = [t0,tf]
    % using Improved Euler's method with N time steps.
    %  Input:
    %  f = name of inline function or function M-file that evaluates
 the ODE
    %          (if not an inline function,  use:
 impeuler(@f,tspan,y0,N))
    %          For a system, the f must be given as column vector.
```

```
    %  tspan = [t0, tf] where t0 = initial time value and tf = final
 time value
    %  y0  = initial value of the dependent variable. If solving a
 system,
    %           initial conditions must be given as a vector.
    %  N   = number of steps used.
    % Output:
    %  t = vector of time values where the solution was computed
    %  y = vector of computed solution values.

    m = length(y0);
    t0 = tspan(1);
    tf = tspan(2);
    h = (tf-t0)/N;              % evaluate the time step size
    t = linspace(t0,tf,N+1);     % create the vector of t values
    y = zeros(m,N+1);   % allocate memory for the output y
    y(:,1) = y0';               % set initial condition
    for n=1:N
        f1 = f(t(n), y(:,n)); % declaring f1 = (t_n,y_n)
        f2 = f(t(n) + h, y(:,n) + h*f1); % declaring f2 = (f(t_n+h,
 y_n+hf_1)
        y(:,n+1) = y(:,n) + (h/2)*(f1+f2); % implement Improved
 Euler's method
    end
    t = t'; y = y';    % change t and y from row to column vectors
end

ans =

         0    1.0000
    0.1000    1.3450
    0.2000    1.8090
    0.3000    2.4331
    0.4000    3.2726
    0.5000    4.4016
    0.6000    5.9202
```

My output matches the output of the protocol.

Exercise 4

Part (a)

```
clc                        % clearing command window
'impeuler.m';              % calling impeuler.m file
f=@(t,y) 3*y;              % Defining ODE function f for this lab
t=linspace(0,0.6,100);     % Defining vector t of time values
y=exp(3*t);                % Defining solution for y
[t6,y6]=impeuler(f,[0,0.6],1,6); % solve the ODE using impeuler w/ 60
 steps
[t60,y60]=impeuler(f,[0,0.6],1,60); % solve the ODE using impeuler w/
 60 steps
```

```
[t600,y600]=impeuler(f,[0,0.6],1,600); % solve the ODE using impeuler
 w/ 600 steps
[t6000,y6000]=impeuler(f,[0,0.6],1,6000);% solve the ODE using
 impeuler w/ 6000 steps
y6(end), y60(end), y600(end), y6000(end) % evaluating approximations @
 6, 60, 600, 6000
e6 = y(end) - y6(end)           % error when N = 6
e60 = y(end) - y60(end)         % error when N = 60
e600 = y(end) - y600(end)       % error when N = 600
e6000 = y(end) - y6000(end)     % error when N = 6000
ratio60 = e6 / e60              % ratio of Nsmall/Nmed
ratio600 = e60/e600             % ratio of Nmed/Nlarge
ratio6000 = e600/e6000          % ratio of Nlarge/Nhuge

disp('----------------------------------------------------')
disp('|     N      |  approximation  |  error      | ratio  |')
disp('|-----------|-----------------|----------   |--------|')
disp('| 6          |    5.9202       |  0.1295     |  N/A   |')
disp('| 60         |    6.0481       |  0.0016     | 81.0854|')
disp('| 600        |    6.0496       |  1.6297e-5 | 97.9844|')
disp('| 6000       |    6.0496       |  1.6330e-7 | 99.7976|')
disp('----------------------------------------------------')


ans =

    5.9202


ans =

    6.0481


ans =

    6.0496


ans =

    6.0496


e6 =

    0.1295


e60 =

    0.0016
```

```
e600 =

   1.6297e-05


e6000 =

   1.6330e-07


ratio60 =

   81.0854


ratio600 =

   97.9844


ratio6000 =

   99.7976

---------------------------------------------------
|     N     |  approximation   |  error      | ratio   |
|-----------|------------------|----------   |--------|
| 6         |     5.9202       |  0.1295     |  N/A    |
| 60        |     6.0481       |  0.0016     | 81.0854|
| 600       |     6.0496       |  1.6297e-5  | 97.9844|
| 6000      |     6.0496       |  1.6330e-7  | 99.7976|
---------------------------------------------------
```

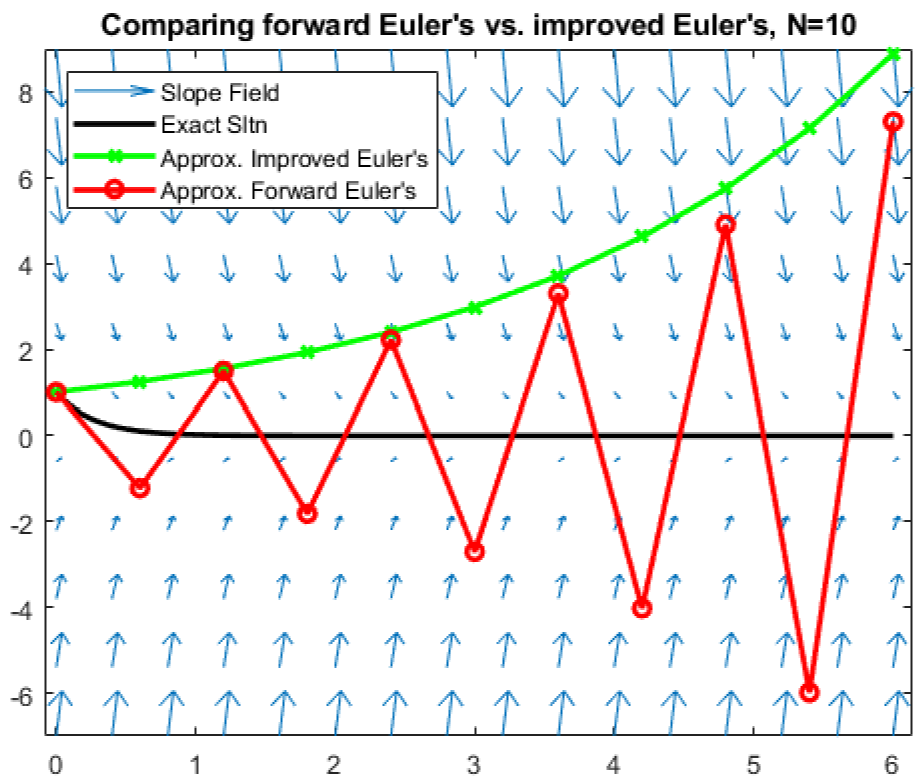
Part (b)

The ratio of errors is proportional to the number of N steps. The higher you go with the number of steps, the closer the ratio will get to 1. At N=60000, the ratio is 1.

Exercise 5

```
clc % clearing command window
t = 0:0.4:6; y = -7:1.6:9; % define a grid in t & y directions
[T,Y] = meshgrid (t,y); % create 2d matrices of points in ty - plane
dT = ones ( size (T)); %dt =1 for all points
dY = -3.7*Y; %dy = -3.7*y; this is the ODE
quiver (T,Y,dT ,dY) % draw arrows (t,y)->(t+dt , t+dy)
axis tight % adjust look
hold on % holding all plots on this figure until holding off
t = linspace(0, 6, 100); % defining vector t of given interval for
 exact solution
y = exp(-3.7.*t); % defining the exact solution to dydt
plot(t, y, 'k', "Linewidth",2) % plotting analytical solution vector
 with slopefield
```

```
f = @(t,y) -3.7*y; % defining the ODE as an anonymous function
[t10, y10] = impeuler(f, [0,6], 1, 10); % Solving ODE with Improved
 Euler's method using 10 steps
plot(t10, y10, 'gx-', 'linewidth', 2) % Plotting the approx solution
 N=10
[t10forward, y10forward] = euler(f, [0,6], 1, 10); % Solving ODE with
 forward Euler's method using 10 steps
plot(t10forward, y10forward, 'ro-', 'linewidth', 2) % Plotting the
 approx solution N=10
legend('Slope Field','Exact Sltn',"Approx. Improved Euler's","Approx.
 Forward Euler's",'location','northwest')
title("Comparing forward Euler's vs. improved Euler's, N=10") % Adding
 a title to the plot
```



Comparing results

If you compare the results of the forward euler's method to improved eulers method. The improved eulers approximation follows along the linear fit of the positive points of the forward eulers method.

Published with MATLAB® R2019b