Lab 3 - Dean Styx - MAT 275 Lab

Table of Contents

Introduction to Numerical Methods for Solving ODEs

Exercise 1

Part (a)

```
% NOTE: We often define the right-hand side of an ODE as some
 function, say
% f, in terms of the input and output variables in the ODE. For
 example,
% dy/dt = f(t,y) = 3y. Although the ODE does not explicitly depend on
 t, it does
% so implicitly since y is a function of t. Therefore, t is an input
% variable of f.

% Define ODE function f for ODE.

f=@(t,y)(4*y);
t=linspace(0,.5,100);
y = 2*exp(4*t);

%inital value of third part of euler
y0 = 2;
% Define vector t of time-values over the given interval to compute
% analytical solution vector.
tspan = [0,0.5];

% Solve IVP numerically using forward Euler's method with 5 timesteps.
[t5,y5] = euler(f,tspan, y0,5);

% Solve IVP numerically using forward Euler's method with 50
 timesteps.
[t50,y50] = euler(f,tspan, y0,50);
% Solve IVP numerically using forward Euler's method with 500
 timesteps.
[t500,y500] = euler(f,tspan, y0,500);
% Solve IVP numerically using forward Euler's method with 5000
 timesteps.
[t5000,y5000] = euler(f,tspan, y0,5000);
% In the following steps, we define error as analytical (solution
 value
% - numerical solution value).


% Compute numerical solution error at t=0.5 for forward Euler with 5
% timesteps.
a5=y5(end);

% Compute numerical solution error at t=0.5 for forward Euler with 50
% timesteps.
a50=y50(end);

% Compute numerical solution error at t=0.5 for forward Euler with 500
% timesteps.
a500=y500(end);

% Compute numerical solution error at t=0.5 for forward Euler with
 5000
% timesteps.
a5000=y5000(end);

% Compute ratio of errors between N=5 and N=50.
e5=y(end)-y5(end);

% Compute ratio of errors between N=50 and N=500.
e50=y(end)-y50(end);

% Compute ratio of errors between N=500 and N=5000.
e500=y(end)-y500(end);
e5000=y(end)-y5000(end);

% NOTE: To complete the following table, simply run this section after
% you've finished the steps above and copy and paste your errors and
 ratios
% into the follwing display commands. Make sure you re-align each
 column
% after entering in values so that it looks pretty.
r50=e5/e50;
r500=e50/e500;
r5000=e500/e5000;
% Display table of errors (and ratios of consecutive errors) for the
% numerical solution at t=0.5 (the last element in the solution
 vector).

disp('---------------------------------------------')
disp('| N    |  approximation  |  error    | ratio  |')
disp('|------|-----------------|-----------|--------|')
disp('| 5    |     10.7565     |  4.0216   |  N/A   |')
disp('| 50   |     14.2134     |  0.5647   | 7.1211 |')
disp('| 500  |     14.7193     |  0.0588   | 9.5983 |')
disp('| 5000 |     14.7722     |  0.0059   | 9.9582 |')
disp('---------------------------------------------')
```

```
---------------------------------------------
| N    |  approximation  |  error    | ratio  |
|------|-----------------|----------|--------|
| 5    |      10.7565    |  4.0216  |  N/A   |
| 50   |      14.2134    |  0.5647  | 7.1211 |
| 500  |      14.7193    |  0.0588  | 9.5983 |
| 5000 |      14.7722    |  0.0059  | 9.9582 |
----------------------------------------------
```

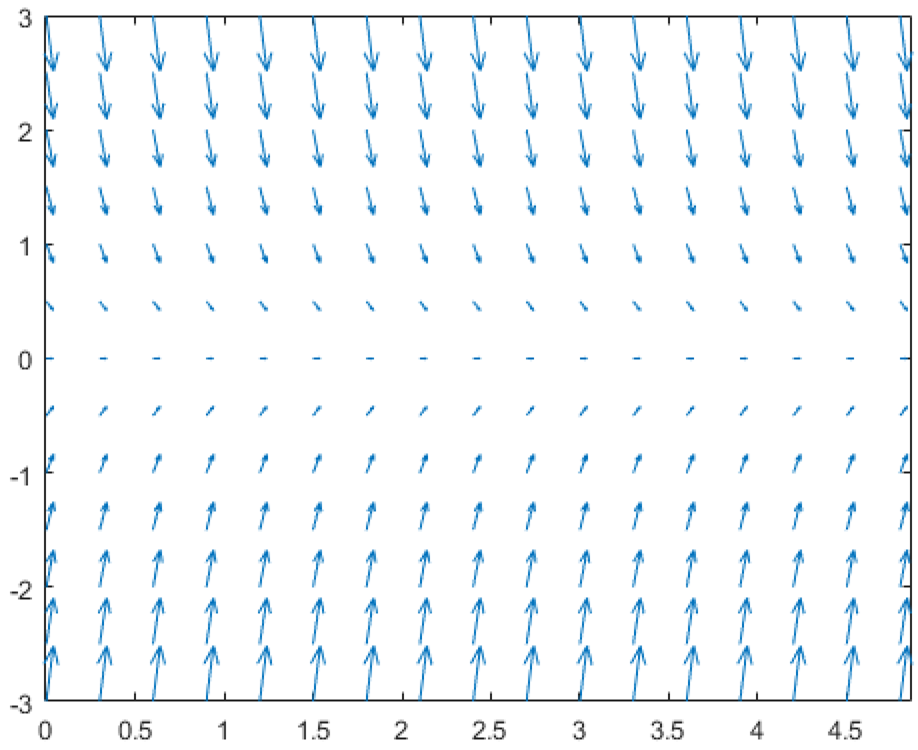
Part (b). Your answer here. While looking at the last row, it can be seen by looking from bottom to top (5 to 5000), that it appears the ratio is approaching the limit of 10. And considering the step value is 1/10, it can be considered that the higher value of N, the closer we get to the error ratio of the inverse of our step value.

Part (c). In the ODE given, it graphs out to be an exponentially rising curve . Given that as every infidecimal step upwards increases the slope, unless the step size given to the computer is infinity, it will always be an underestimation of the real value.

Exercise 2

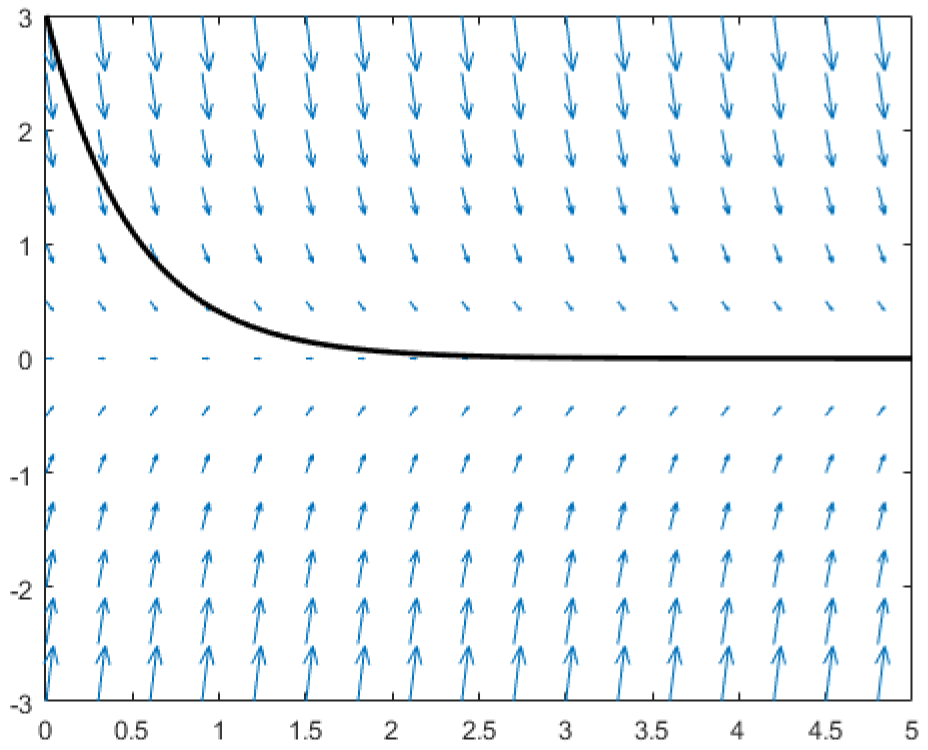
Part (a)

```
figure
% Plot slopefield for the new ODE  using the given commands.
t = 0:.3:5; y = -3:0.5:3 ; % define grid of values in t and y
 direction
[T,Y]=meshgrid(t,y); % creates 2d matrices of points in the ty-plane
dT = ones(size(T)); % dt=1 for all points
dY = -4*Y; % dy = -3*y; this is the ODE
quiver(T,Y,dT,dY) % draw arrows (t,y)->(t+dt, t+dy)
axis tight % adjust look
hold on
```



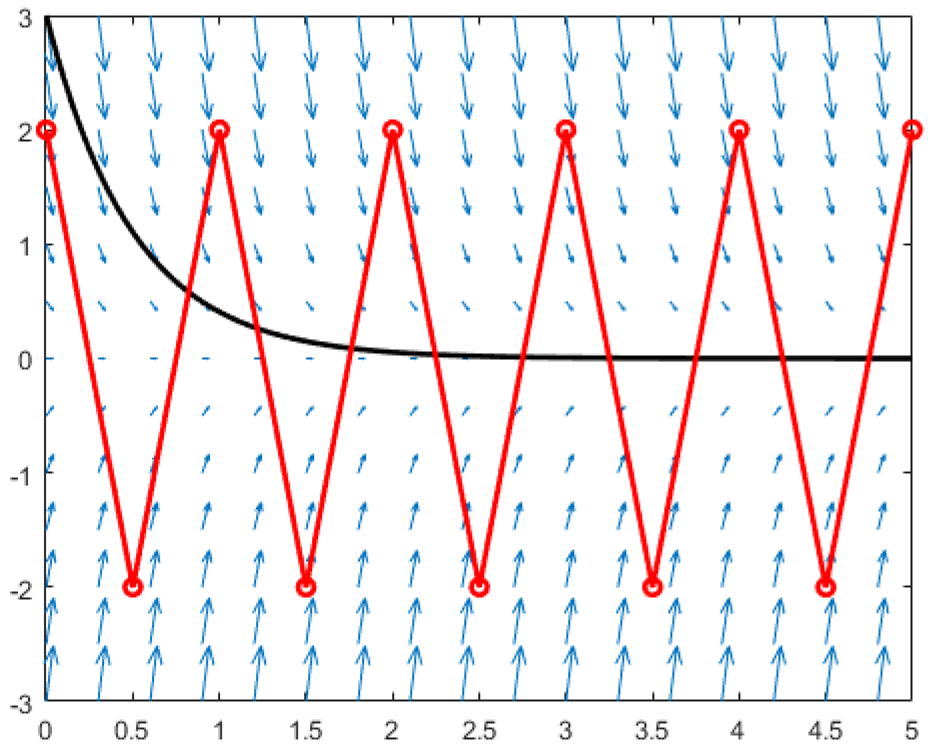
Part (b)

```
% Define vector t of time-values over the  given interval  to define
% analytical solution vector.
t= linspace(0,5, 100);
y=3*exp(-2*t);
% Create vector of analytical solution values at corresponding t
 values.

% NOTE: The "hold on" command in the segment of code to plot the slope
% field indicates to MATLAB to add future plots to the same window
 unless
% otherwise specified using the "hold off" command.

% Plot analytical solution vector with slopefield from (a).
plot(t,y, 'k-', 'linewidth',2)
```



Part (c)

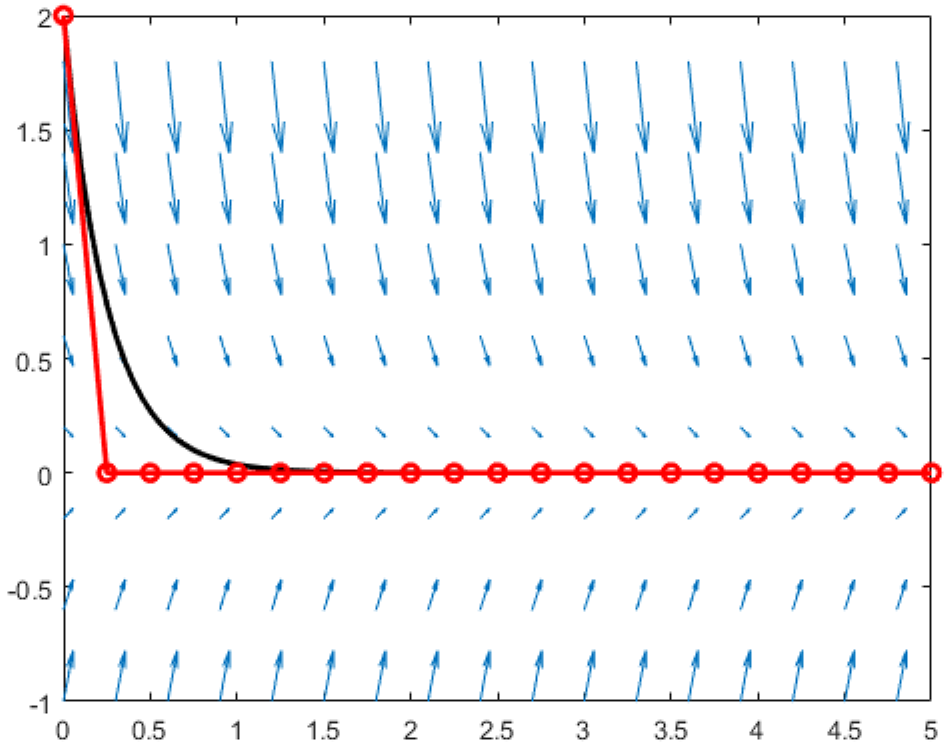
```
% NOTE: Recall we can define the right-hand side of an ODE as some
 function
% f. In the case of a one-dimensional autonomous ODE, we may define f
 such
% that dy/dt = f(t,y).

% Define ODE function.
f=@(t,y) (-4*y);

% Compute numerical solution to IVP with N timesteps using forward
 Euler.
y0=2;
tspan = [0,5];
N=10;
[t2,y2] = euler(f,tspan,y0,N);
plot(t2,y2, 'ro-', 'linewidth', 2)
% Plot numerical solution with analytical solution from (b) and
 slopefield
% from (a) using circles to distinguish between the approximated data
% (i.e., the numerical solution values) and actual (analytical)
 solution.

hold off;  % end plotting in this figure window
figure
```


3
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

```
%Because with this equation, exact solution is equal to 2/(e^(4t)). So
 as t
%increases, the value of y gets smaller leading it to be closer and
 closer
%to 0 as it gets exponentially larger.
```

Part (d)

```
% Define new grid of t and y values at which to plot vectors for slope
% field.
t = 0:.3:5; y = -1:0.4:2 ;
[T,Y]=meshgrid(t,y);
dT = ones(size(T));
dY = -4*Y;
quiver(T,Y,dT,dY)
axis tight
hold on

t = linspace(0,5,100);

y=2*exp(-4*t);

plot(t,y, 'k-', 'linewidth',2)

f=@(t,y) (-4*y);
tspan = [0,5];
y0=2;
N=20;
[t3,y3] = euler(f,tspan,y0,N);
plot(t3,y3, 'ro-', 'linewidth', 2)

hold off
% Compute numerical solution to IVP with the given timesteps using
 forward Euler.

% plot numerical solution together with slope field and analytical
 solution
```



The timestep eventually gets two values close to zero, thus making the bar drop down to zero very quickly.

Exercise 3

```
% Display contents of impeuler M-file.
type 'impeuler.m'

% NOTE: Make sure you document your code in impeuler.m appropriately
 for
% full credit.
f=@(t,y) (4*y);
y0 = 2;
% Define ODE function.
[t5,y5] = impeuler(f,[0,.5],y0,5);
% Compute numerical solution to IVP with 5 timesteps using "improved
% Euler."
[t5,y5]


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 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: euler(@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
    y(:,n+1) = y(:,n)+(h/2)*(f(t(n),y(:,n))+f(t(n
+1),y(:,n)+h*f(t(n),y(:,n))));   % implement improved Euler's method
end
t = t'; y = y';    % change t and y from row to column vectors
end

ans =

         0    2.0000
    0.1000    2.9600
    0.2000    4.3808
    0.3000    6.4836
    0.4000    9.5957
    0.5000   14.2016
```

Exactly the same answers.

Exercise 4

```
figure

tspan=[0, 0.5];
f=@(t,y)(4*y);
t=linspace(0,.5,100);
y=2*exp(4*t); %define the exact solution of ODE
y0=2;

%solve IVP numerically using Improcved Eulers method with 5 steps.
[t5,y5] = impeuler(f,tspan, y0,5);
%solve IVP numerically using Improcved Eulers method with 50 steps.
```

```
[t50,y50] = impeuler(f,tspan, y0,50);
%solve IVP numerically using Improcved Eulers method with 500 steps.
[t500,y500] = impeuler(f,tspan, y0,500);
%solve IVP numerically using Improcved Eulers method with 5000 steps.
[t5000,y5000] = impeuler(f,tspan, y0,5000);


% Compute numerical solution error at t=0.5 for Improved Euler with 5
% timesteps.
a5=y5(end)
% Compute numerical solution error at t=0.5 for Improved Euler with 50
% timesteps.
a50=y50(end)
% Compute numerical solution error at t=0.5 for Improved Euler with
 500
% timesteps.
a500=y500(end)
% Compute numerical solution error at t=0.5 for Improved Euler with
 5000
% timesteps.
a5000=y5000(end)




% Compute ratio of errors between N=5 and N=50.
e5=y(end)-y5(end)
% Compute ratio of errors between N=50 and N=500.
e50=y(end)-y50(end)
% Compute ratio of errors between N=500 and N=5000.
e500=y(end)-y500(end)
e5000=y(end)-y5000(end)


% Ratios for results below
r50=e5/e50
r500=e50/e500
r5000=e500/e5000

disp('----------------------------------------------')
disp('| N     |  approximation   |  error     | ratio  |')
disp('|------|-----------------|----------|--------|')
disp('| 5     |     14.2016      |  0.5765   |  N/A   |')
disp('| 50    |     14.7705      |  0.0076   |75.3850 |')
disp('| 500   |     14.7780      |7.8580e-05|97.3146 |')
disp('| 5000  |     14.7781      |7.8793e-07|99.7301 |')
disp('----------------------------------------------')

% NOTE: Here, we repeat all the steps from exercise 1, except using
% "improved Euler" instead of forward Euler (or regular Euler). Make
 sure
% to document your code appropriately.
```

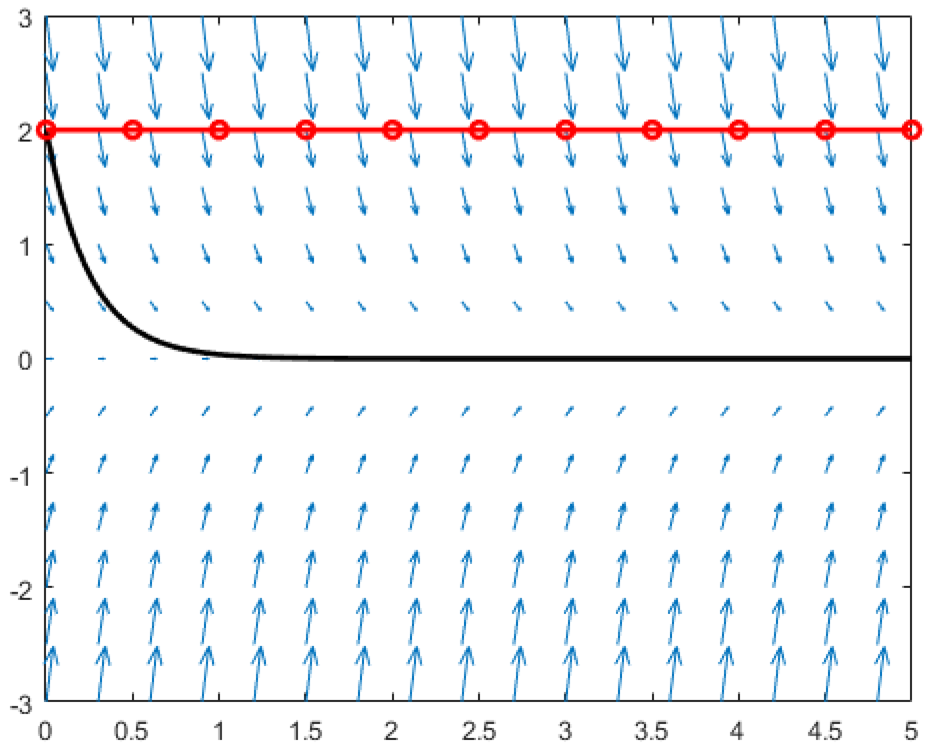
```
a5 =

   14.2016


a50 =

   14.7705


a500 =

   14.7780


a5000 =

   14.7781


e5 =

    0.5765


e50 =

    0.0076


e500 =

   7.8580e-05


e5000 =

   7.8793e-07


r50 =

   75.3850


r500 =

   97.3146


r5000 =

   99.7301
```

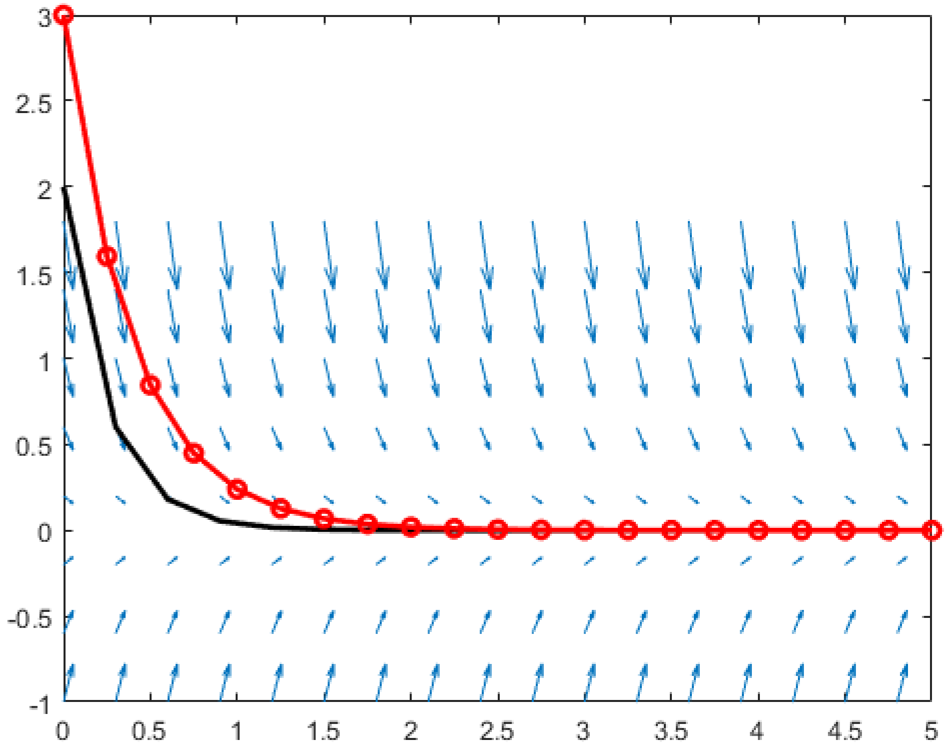
```
----------------------------------------------
| N    |  approximation   |   error   | ratio  |
|------|------------------|-----------|--------|
| 5    |     14.2016      |  0.5765   |  N/A   |
| 50   |     14.7705      |  0.0076   |75.3850 |
| 500  |     14.7780      |7.8580e-05|97.3146 |
| 5000 |     14.7781      |7.8793e-07|99.7301 |
----------------------------------------------
```

Exercise 5

```
t = 0:.3:5; y = -3:0.5:3 ; % define grid of values in t and y
 direction
[T,Y]=meshgrid(t,y); % creates 2d matrices of points in the ty-plane
dT = ones(size(T)); % dt=1 for all points
dY = -4*Y; % dy = -3*y; this is the ODE
quiver(T,Y,dT,dY) % draw arrows (t,y)->(t+dt, t+dy)
axis tight % adjust look
hold on

t= 0:.045:5;
y=2*exp(-4*t);

plot(t,y, 'k-', 'linewidth',2)
```

```
f=@(t,y) (-4*y);
tspan = [0,5];
y0=2;
N=10;
[t10,y10] = impeuler(f,tspan,y0,N);
plot(t10,y10, 'ro-', 'linewidth', 2)

hold off;  % end plotting in this figure window
figure

t = 0:.3:5; y = -1:0.4:2 ; % define grid of values in t and y
 direction
[T,Y]=meshgrid(t,y); % creates 2d matrices of points in the ty-plane
dT = ones(size(T)); % dt=1 for all points
dY = -4*Y; % dy = -3*y; this is the ODE
quiver(T,Y,dT,dY) % draw arrows (t,y)->(t+dt, t+dy)
axis tight
hold on

t = 0:.3:5;
y = -1:0.4:2 ;
y=2*exp(-4*t);

plot(t,y, 'k-', 'linewidth',2)

f=@(t,y) (-3*y);
tspan = [0,5];
y0=3;
N=20;
[t20,y20] = impeuler(f,tspan,y0,N);
plot(t20,y20, 'ro-', 'linewidth', 2)

hold off
% NOTE: Here, we repeat all the steps from exercise 2, except using
% improved Euler rather than forward Euler. Remember code doc.
```


3
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Congratulations, you're done!

Published with MATLAB® R2017a