Table of Contents

G12ISC 2019-2020 Coursework 6

Subject: Numerical solution of IVP Name: Jake Denton Student ID:14322189

```
clear all
close all
clc
```

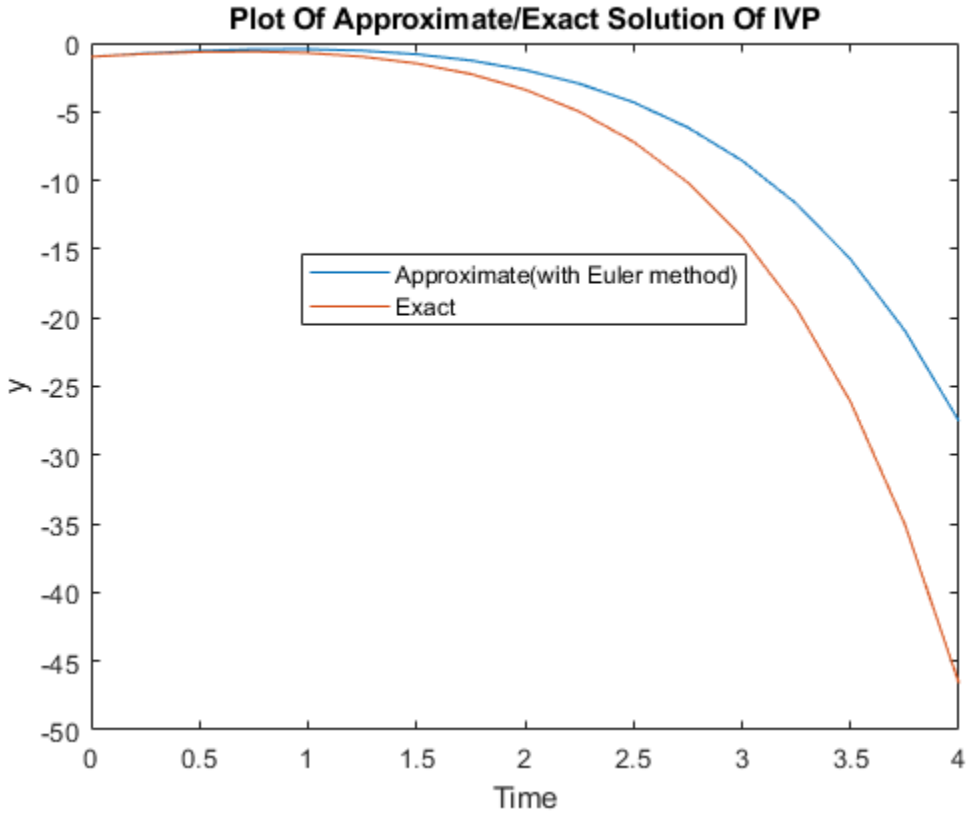
Question 1

This code creates a table and plot which show the approximate and exact solutions to the initial value problem using 16 time steps in the interval [0,4]. The table also has a column displaying the error in each approximation.

```
format short e %initially used format long but this wouldn't fit all
 data in table on the published pdf
f=@(t,y) 2*(1-t)+y;%defines the function f(t,y) as given
[Time,ApproxEul]=Euler(0,4,16,-1,f);%calls the Euler function to
 approximate the solution along with the times of each estimate
g=@(t) 2*t-exp(t);%defines the exact solution y
Exact=g(Time);%evaluates the exact solution y at the times it has been
 approximated above
Error=abs(Exact-ApproxEul);%finds the absolute error of the
 approximation
T=table(Time,ApproxEul,Exact,Error);%creates a table with columns as
 required
disp(T);
plot(Time,ApproxEul);%next few lines plot the approximate/exact
 solutions
hold on
plot(Time,Exact);
hold off
title('Plot Of Approximate/Exact Solution Of IVP');%rest of this code
 formats the plot
xlabel('Time');
ylabel('y');
legend('Approximate(with Euler method)','Exact','Location','Best');
```

Time	ApproxEul	Exact	Error

0.0000e+00	-1.0000e+00	-1.0000e+00	0.0000e+00
2.5000e-01	-7.5000e-01	-7.8403e-01	3.4025e-02
5.0000e-01	-5.6250e-01	-6.4872e-01	8.6221e-02
7.5000e-01	-4.5313e-01	-6.1700e-01	1.6388e-01
1.0000e+00	-4.4141e-01	-7.1828e-01	2.7688e-01
1.2500e+00	-5.5176e-01	-9.9034e-01	4.3859e-01
1.5000e+00	-8.1470e-01	-1.4817e+00	6.6699e-01
1.7500e+00	-1.2684e+00	-2.2546e+00	9.8623e-01
2.0000e+00	-1.9605e+00	-3.3891e+00	1.4286e+00
2.2500e+00	-2.9506e+00	-4.9877e+00	2.0372e+00
2.5000e+00	-4.3132e+00	-7.1825e+00	2.8693e+00
2.7500e+00	-6.1415e+00	-1.0143e+01	4.0011e+00
3.0000e+00	-8.5519e+00	-1.4086e+01	5.5336e+00
3.2500e+00	-1.1690e+01	-1.9290e+01	7.6004e+00
3.5000e+00	-1.5737e+01	-2.6115e+01	1.0378e+01
3.7500e+00	-2.0922e+01	-3.5021e+01	1.4099e+01
4.0000e+00	-2.7527e+01	-4.6598e+01	1.9071e+01



Question 2

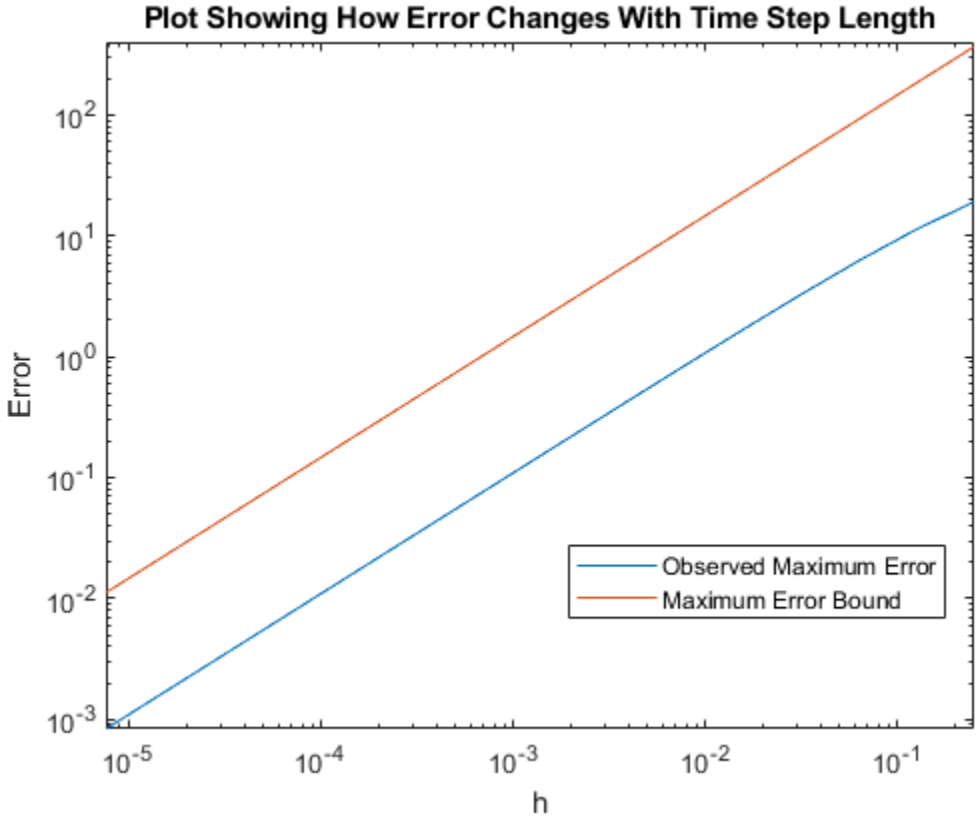
This code calculates the maximum absolute error of Euler's method for different lengths of time step h, producing a table then a double log plot, alongside the curve that represents the maximum theoretical error bound for each h. Clearly the two curves shown in the plot have the same slope (they appear parallel), and

thus share the same order of convergence. The error bound is known to converge linearly in h therefore we can conclude that the observed asymptotic rate of convergence is as expected (O(h)).

```
format long g
f=@(t,y) 2*(1-t)+y;%first two lines define functions as given
g=@(t) 2*t-exp(t);
H=zeros(16,1);%these two lines initialise vectors to go into for loop
 below
Emax=zeros(16,1);
M=exp(4);%bound on the absolute value of second derivative of the
 exact solution y
L=1;%this is the Lipschitz constant for the IVP
c=(M/(2*L))*[exp(L*(4-0))-1];%this is the constant multiplying h in
 the error bound
Bound=zeros(16,1);
for i=1:16
    h=2^(-1*(i+1));%this tells us the length of each time step for
 this iteration
    N=4/h;%this is the number of time steps in total in the interval
    [Time,Approx]=Euler(0,4,N,-1,f);%calls Euler function to give
 approximate solution with N time steps
    Exact=g(Time);%finds exact solution at each time step
    Error=abs(Exact-Approx);%defines a vector with elements being the
 absolute error for each time step
    Emax(i)=max(Error);%assigns ith element of Emax with the max.
 absolute value of the error for this h
    H(i)=h;%assigns ith element of H with the h used in this iteration
    Bound(i)=c*h;%assigns ith element of Bound with the theoretical
 error bound
end
T=table(H,Emax);%creates a table with two columns as required
T.Properties.VariableNames={'h','MaxAbsError'};%renames the headings
 in the table
disp(T);
loglog(H,Emax);%next few lines plot the two curves on double log axes
hold on
loglog(H,Bound);
hold off
xlim([2^(-17),0.25]);%rest of this code formats the plot
ylim([0,400]);
xlabel('h');
ylabel('Error');
title('Plot Showing How Error Changes With Time Step Length')
legend('Observed Maximum Error','Maximum Error
 Bound','Location','Best');
```

h	MaxAbsError
0.25	19.0710132451392
0.125	11.2589672056451
0.0625	6.17510699512211
0.03125	3.24263613512311
0.015625	1.66277909450294

```
    0.0078125       0.842116830884862
    0.00390625      0.423787224333665
    0.001953125     0.212581800563136
    0.0009765625    0.106463702833189
    0.00048828125   0.0532751469391215
    0.000244140625  0.0266484092392432
    0.0001220703125 0.0133269150498023
    6.103515625e-05 0.0066641353186867
    3.0517578125e-05 0.00333223713023756
    1.52587890625e-05 0.00166616093532213
    7.62939453125e-06 0.000833091060528091
```



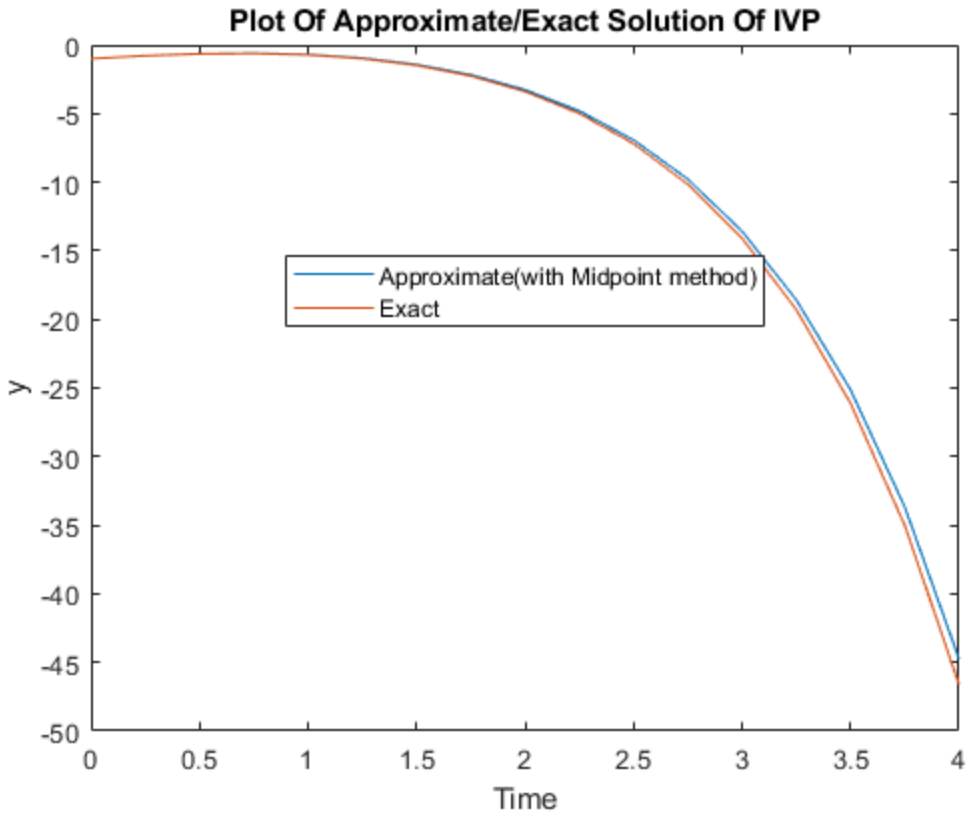
Question 4

This code creates a table and a plot as in Q1, but instead uses the Midpoint method to obtain the approximations with 16 time steps.

```
format short e %again had issues with format long on the pdf
f=@(t,y) 2*(1-t)+y;%defines required functions as previously
g=@(t) 2*t-exp(t);
[Time,ApproxMid]=Midpoint(0,4,16,-1,f);%uses the Midpoint method to
 obtain times/approximations
Exact=g(Time);%defines vector of y values for the exact solution
Error=abs(Exact-ApproxMid);%calculates absolute value of error
T=table(Time,ApproxMid,Exact,Error);%creates table using the above
```

```
disp(T);
plot(Time,ApproxMid);%plots the approximate solution against time
hold on
plot(Time,Exact);%plots exact solution against time
hold off
title('Plot Of Approximate/Exact Solution Of IVP');%rest of this code
 formats the plot
xlabel('Time');
ylabel('y');
legend('Approximate(with Midpoint method)','Exact','Location','Best');
```

Time	*ApproxMid*	*Exact*	*Error*
0.0000e+00	*-1.0000e+00*	*-1.0000e+00*	*0.0000e+00*
2.5000e-01	*-7.8125e-01*	*-7.8403e-01*	*2.7754e-03*
5.0000e-01	*-6.4160e-01*	*-6.4872e-01*	*7.1197e-03*
7.5000e-01	*-6.0330e-01*	*-6.1700e-01*	*1.3698e-02*
1.0000e+00	*-6.9486e-01*	*-7.1828e-01*	*2.3426e-02*
1.2500e+00	*-9.5278e-01*	*-9.9034e-01*	*3.7559e-02*
1.5000e+00	*-1.4239e+00*	*-1.4817e+00*	*5.7810e-02*
1.7500e+00	*-2.1681e+00*	*-2.2546e+00*	*8.6507e-02*
2.0000e+00	*-3.2622e+00*	*-3.3891e+00*	*1.2681e-01*
2.2500e+00	*-4.8048e+00*	*-4.9877e+00*	*1.8298e-01*
2.5000e+00	*-6.9217e+00*	*-7.1825e+00*	*2.6078e-01*
2.7500e+00	*-9.7747e+00*	*-1.0143e+01*	*3.6793e-01*
3.0000e+00	*-1.3571e+01*	*-1.4086e+01*	*5.1483e-01*
3.2500e+00	*-1.8575e+01*	*-1.9290e+01*	*7.1537e-01*
3.5000e+00	*-2.5127e+01*	*-2.6115e+01*	*9.8815e-01*
3.7500e+00	*-3.3663e+01*	*-3.5021e+01*	*1.3580e+00*
4.0000e+00	*-4.4740e+01*	*-4.6598e+01*	*1.8579e+00*



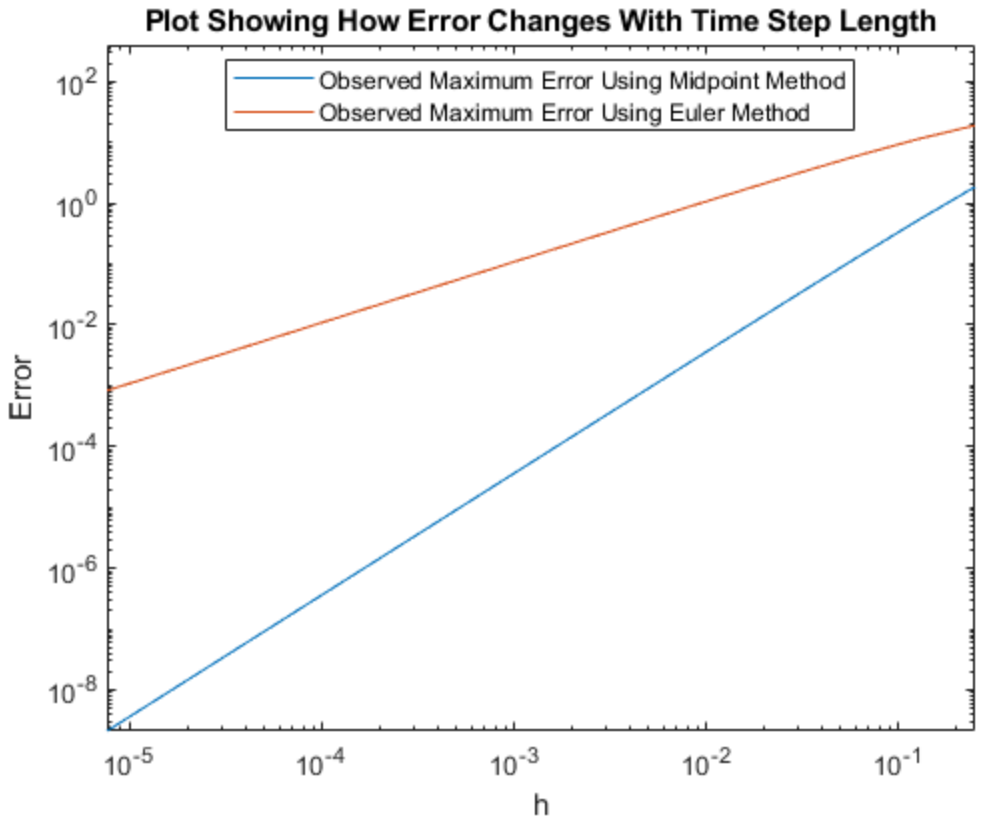
Question 5

This code repeats question 2 with the midpoint method, keeping the line that represents observed error in Euler method so that I can compare the rates of convergence. It's clear from the plot that the rate of convergence of the midpoint method is higher than that of Euler's method since the slope is steeper, and this is as expected since the midpoint method has error of order h squared.

```
format long g
f=@(t,y) 2*(1-t)+y;%first two lines define functions as given
g=@(t) 2*t-exp(t);
H=zeros(16,1);%these three lines initialise vectors to go into for
 loop below
EmaxMid=zeros(16,1);
EmaxEul=zeros(16,1);%kept in because I'm interested in comparing rates
 of convergence between the two methods
for i=1:16
    h=2^(-1*(i+1));%this tells us the length of each time step for
 this iteration
    N=4/h;%this is the number of time steps in total in the interval
    [~,ApproxEul]=Euler(0,4,N,-1,f);%calls Euler function to give
 approximate solution using Euler method
    [Time,ApproxMid]=Midpoint(0,4,N,-1,f);%calls Midpoint function to
 give approximate solution with N time steps
    Exact=g(Time);%finds exact solution at each time step
    ErrorEul=abs(Exact-ApproxEul);
```

```
    ErrorMid=abs(Exact-ApproxMid);%defines a vector with elements
 being the absolute error for each time step
    EmaxEul(i)=max(ErrorEul);
    EmaxMid(i)=max(ErrorMid);%assigns ith element of EmaxMid with the
 max. absolute value of the error for this h
    H(i)=h;%assigns ith element of H with the h used in this iteration
end
T=table(H,EmaxMid);%creates a table with two columns as required
T.Properties.VariableNames={'h','MaxAbsErrorMid'};%renames the
 headings in the table
disp(T);
loglog(H,EmaxMid);%next few lines plot the two curves on double log
 axes
hold on
loglog(H,EmaxEul);%decided to keep this line so we could compare the
 observed rates of convergence
hold off
xlim([2^(-17),0.25]);%rest of this code formats the plot
ylim([0,400]);
xlabel('h');
ylabel('Error');
title('Plot Showing How Error Changes With Time Step Length')
legend('Observed Maximum Error Using Midpoint Method','Observed
 Maximum Error Using Euler Method','Location','Best');
```

h	MaxAbsErrorMid
0.25	1.85791578537409
0.125	0.515614687312308
0.0625	0.135515811478555
0.03125	0.0347119400686751
0.015625	0.00878222435137133
0.0078125	0.00220858320512463
0.00390625	0.000553773685368242
0.001953125	0.00013864686471976
0.0009765625	3.46871435965568e-05
0.00048828125	8.67496430601022e-06
0.000244140625	2.16913809936159e-06
0.0001220703125	5.42333722819421e-07
6.103515625e-05	1.35590006777875e-07
3.0517578125e-05	3.38994468052078e-08
1.52587890625e-05	8.47555270411249e-09
7.62939453125e-06	2.11816342243765e-09



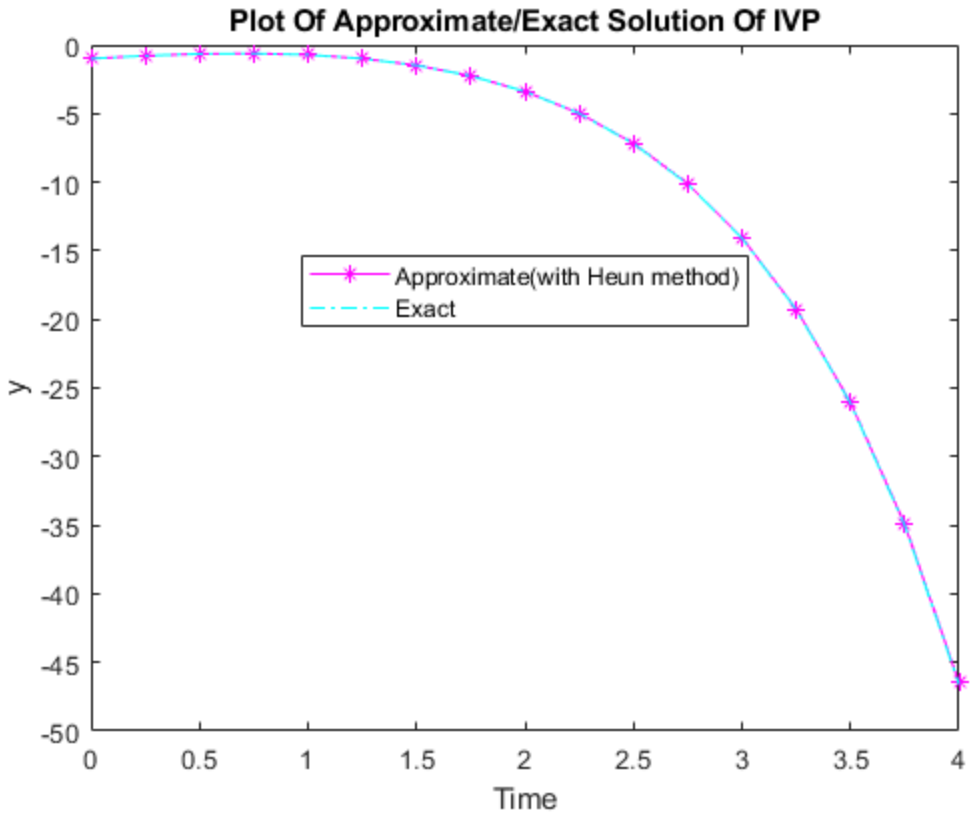
Question 7

This code creates a table and plot as in Q1, but uses Heun's method to obtain the approximations with N=16 time steps.

```
format short e
f=@(t,y) 2*(1-t)+y;%defines the function f(t,y) as given
[Time,ApproxHeun]=Heun(0,4,16,-1,f);%calls the Heun function to
 approximate the solution along with the times of each estimate
g=@(t) 2*t-exp(t);%defines the exact solution y
Exact=g(Time);%evaluates the exact solution y at the times it has been
 approximated above
Error=abs(Exact-ApproxHeun);%finds the absolute error of the
 approximation
T=table(Time,ApproxHeun,Exact,Error);%creates a table with columns as
 required
disp(T);
plot(Time,ApproxHeun,'-*m');%next few lines plot the approximate/exact
 solutions
hold on
plot(Time,Exact,'-.c');%plots the exact line as dotted so we can see
 the approximation clearly
hold off
title('Plot Of Approximate/Exact Solution Of IVP');%rest of this code
 formats the plot
xlabel('Time');
```

```
ylabel('y');
legend('Approximate(with Heun method)','Exact','Location','Best');
```

Time	ApproxHeun	Exact	Error
0.0000e+00	-1.0000e+00	-1.0000e+00	0.0000e+00
2.5000e-01	-7.8385e-01	-7.8403e-01	1.7125e-04
5.0000e-01	-6.4828e-01	-6.4872e-01	4.3975e-04
7.5000e-01	-6.1615e-01	-6.1700e-01	8.4692e-04
1.0000e+00	-7.1683e-01	-7.1828e-01	1.4499e-03
1.2500e+00	-9.8802e-01	-9.9034e-01	2.3269e-03
1.5000e+00	-1.4781e+00	-1.4817e+00	3.5851e-03
1.7500e+00	-2.2492e+00	-2.2546e+00	5.3703e-03
2.0000e+00	-3.3812e+00	-3.3891e+00	7.8801e-03
2.2500e+00	-4.9764e+00	-4.9877e+00	1.1382e-02
2.5000e+00	-7.1663e+00	-7.1825e+00	1.6238e-02
2.7500e+00	-1.0120e+01	-1.0143e+01	2.2933e-02
3.0000e+00	-1.4053e+01	-1.4086e+01	3.2122e-02
3.2500e+00	-1.9246e+01	-1.9290e+01	4.4680e-02
3.5000e+00	-2.6054e+01	-2.6115e+01	6.1779e-02
3.7500e+00	-3.4936e+01	-3.5021e+01	8.4986e-02
4.0000e+00	-4.6482e+01	-4.6598e+01	1.1639e-01



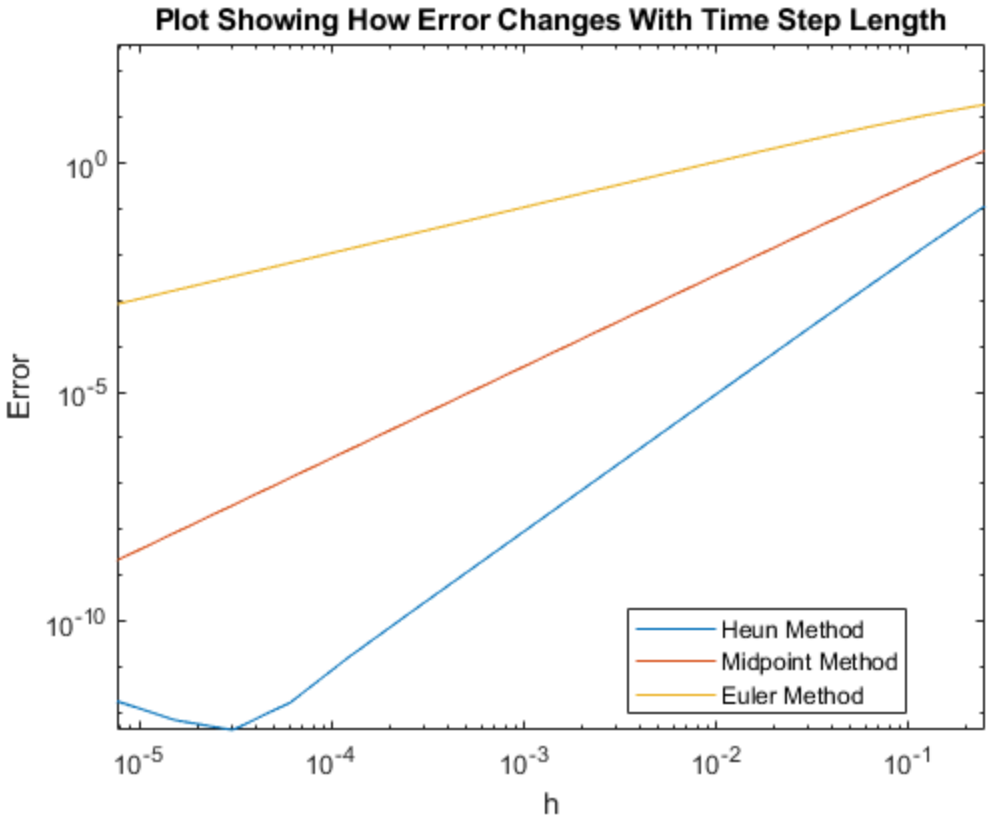
Question 8

This code repeats Q2 with Heun's method, keeping both the lines that represent observed error in the previous two methods so that I can compare the rates of convergence. It's clear from the plot that the rate of convergence of Heun's method is higher than that of the other methods since the slope is steeper. This is as expected since Heun's method has an error of order h cubed. However, for h<2^-15 (optimum time step length), error begins to increase. This is due to round-off error (which is inversely proportional to h).

```
format long g
f=@(t,y) 2*(1-t)+y;%first two lines define functions as given
g=@(t) 2*t-exp(t);
H=zeros(16,1);%these four lines initialise vectors to go into for loop
 below
EmaxHeun=zeros(16,1);
EmaxMid=zeros(16,1);%keeping the other two lines so I can compare the
 observed rates of convergence
EmaxEul=zeros(16,1);
for i=1:16
    h=2^(-1*(i+1));%this tells us the length of each time step for
 this iteration
    N=4/h;%this is the number of time steps in total in the interval
    [~,ApproxEul]=Euler(0,4,N,-1,f);%calls Euler function to give
 approximate solution using Euler method
    [t2,ApproxMid]=Midpoint(0,4,N,-1,f);%calls Midpoint function to
 give approximate solution
    [Time,ApproxHeun]=Heun(0,4,N,-1,f);%calls Heun function to give
 approximate solutions with time step
    Exact=g(Time);%finds exact solution at each time step
    ErrorEul=abs(Exact-ApproxEul);
    ErrorMid=abs(Exact-ApproxMid);
    ErrorHeun=abs(Exact-ApproxHeun);%finds absolute error for Heun's
 method
    EmaxEul(i)=max(ErrorEul);
    EmaxMid(i)=max(ErrorMid);
    EmaxHeun(i)=max(ErrorHeun);%assigns ith element of EmaxHeun with
 the max. absolute value of the error for this h
    H(i)=h;%assigns ith element of H with the h used in this iteration
end
T=table(H,EmaxHeun);%creates a table with two columns as required
T.Properties.VariableNames={'h','MaxAbsErrorHeun'};%renames the
 headings in the table
disp(T);
loglog(H,EmaxHeun);%next few lines plot the curves on double log axes
hold on
loglog(H,EmaxMid);%decided to keep these lines so we could compare the
 observed rates of convergence
loglog(H,EmaxEul);
hold off
xlim([2^(-17),0.25]);%rest of this code formats the plot
ylim([0,400]);
xlabel('h');
ylabel('Error');
title('Plot Showing How Error Changes With Time Step Length');
```

```
legend('Heun Method','Midpoint Method','Euler
 Method','Location','Best');
```

h	MaxAbsErrorHeun
0.25	0.11639131200667
0.125	0.016082605989979
0.0625	0.00211332568635214
0.03125	0.000270846966849092
0.015625	3.42814634564093e-05
0.0078125	4.31203979900374e-06
0.00390625	5.40691708295071e-07
0.001953125	6.769202087753e-08
0.0009765625	8.46809911081436e-09
0.00048828125	1.0589005228212e-09
0.000244140625	1.32338584535319e-10
0.0001220703125	1.61648472385423e-11
6.103515625e-05	1.68398628375144e-12
3.0517578125e-05	4.19220214098459e-13
1.52587890625e-05	6.89226453687297e-13
7.62939453125e-06	1.78346226675785e-12

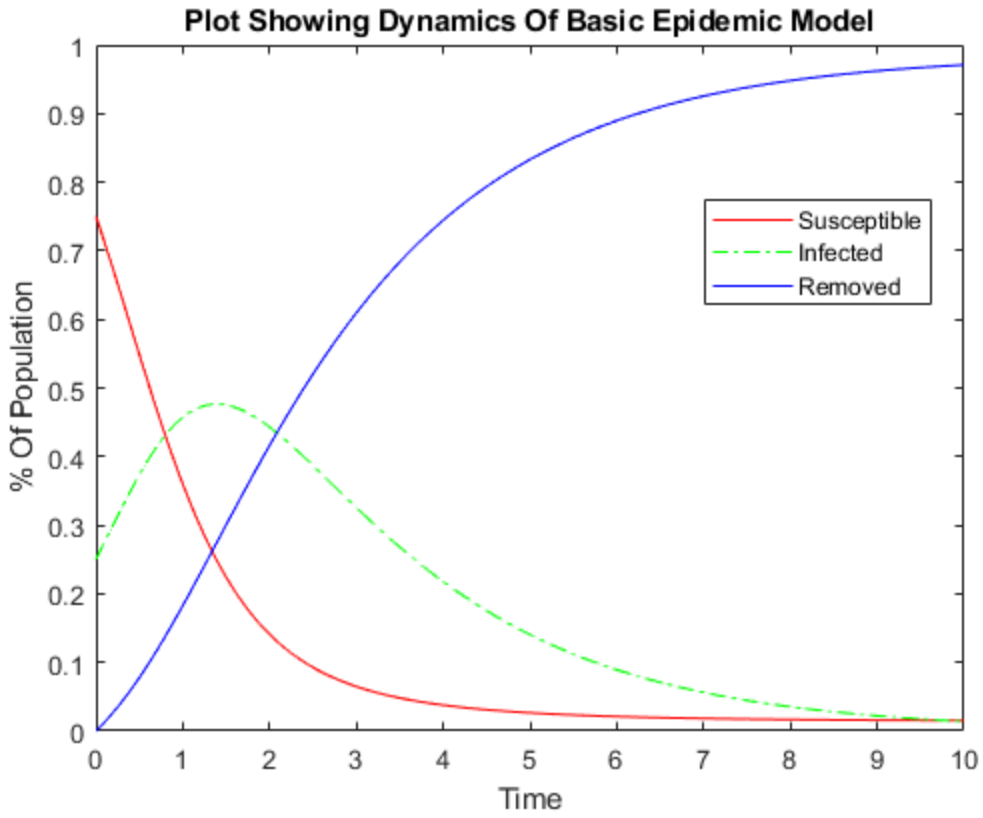


Question 10

This code uses Epidemic.m with the input data as given by the question to produce a plot showing the dynamics of the epidemic over t in (0,10]. It also tells us the peak of the infected proportion of the population for these data. Then, using a for loop and if condition, it finds the minimum percentage of individuals that need to be vaccinated for the epidemic (with 25% of population initially infected) to peak at t=0.

```
[t,w1,w2,w3]=Epidemic(10,1000,0.75,0.25,0,2,0.5);%uses the input
 vector given to obtain the approx. proportion vectors
peak=100*max(w2);%finds the maximum of the infected proportion vector
 and converts to a percentage
disp(['Maximum percentage of population infected at any one time with
 (S0,I0,R0)=(0.75,0.25,0) is ',num2str(peak),'%']);
for S0=0:0.001:0.3%this for loop investigates what percentage requires
 vaccination to keep epidemic under control
    R0=1-S0-0.25;%R0 represents the proportion of the population
 that's vaccinated
    [~,S,I,R]=Epidemic(10,1000,S0,0.25,R0,2,0.5);%calls Epidemic
 function with the appropriate initial proportions for this iteration
    peakI=max(I);%again finds peak of the infected curve
    VacProp=100*R0;%converts R0 to a percentage
    if peakI~=0.25&&peakI>0.25%this if loop breaks as soon as the peak
 of the infected curve is greater than 0.25(I0)
        disp(['The minimum percentage of individuals that
 should be vaccinated to ensure epidemic kept under control is
 ',num2str(VacProp),'%']);
        break
    end
end
plot(t,w1,'-r');%the next chunk of code plots the 3 approx. vectors
 against time for (S0,I0,R0)=(0.75,0.25,0)
hold on
plot(t,w2,'-.g');
plot(t,w3,'-b');
hold off
title('Plot Showing Dynamics Of Basic Epidemic Model');%rest of this
 code formats the graph
xlabel('Time');
ylabel('% Of Population');
legend('Susceptible','Infected','Removed','Location','Best');
```

```
Maximum percentage of population infected at any one time with
 (S0,I0,R0)=(0.75,0.25,0) is 47.647%
The minimum percentage of individuals that should be vaccinated to
 ensure epidemic kept under control is 49.9%
```



Published with MATLAB® R2019a