Table of Contents


```
%Approximating ODES
```

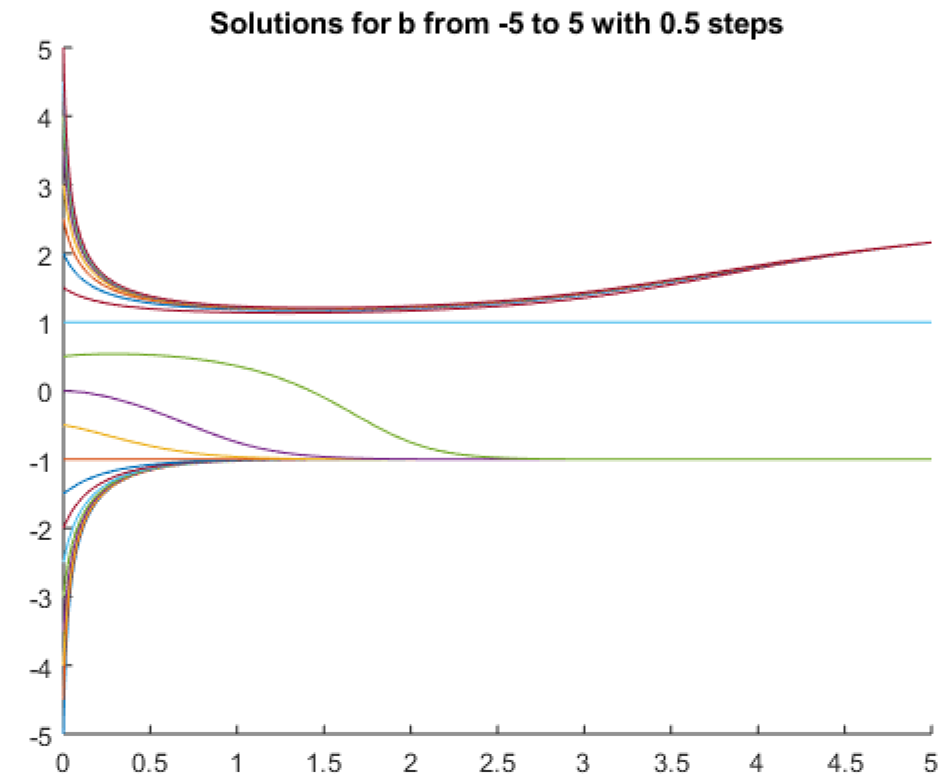
QUESTION

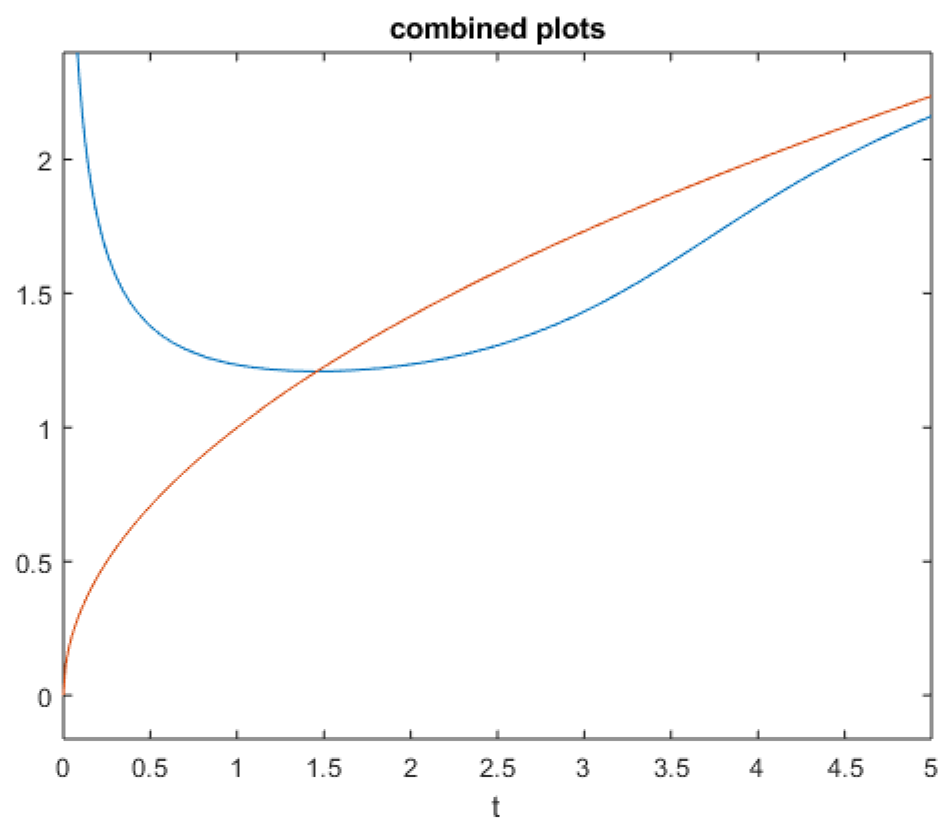
(a)

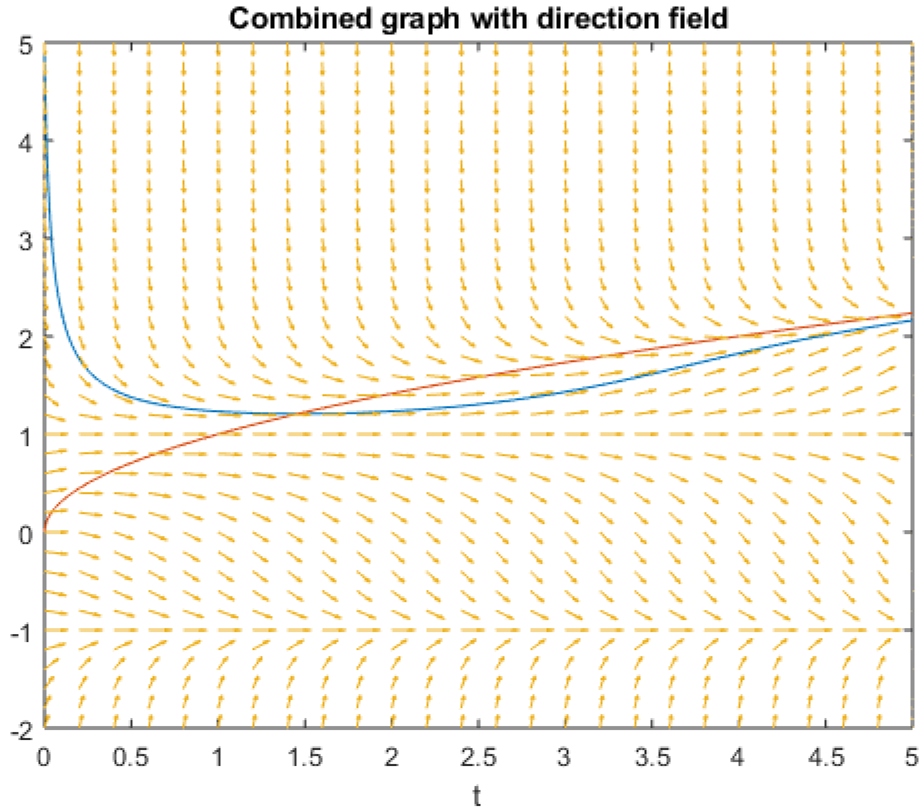
```
f = @(t,y) (y - t.^0.5).*(1 - y.^2);
figure; hold on
for b = -5:0.5:5
    [t,y] = ode45(f,[0,5],b);
    disp(['For b = ',num2str(b),' integration terminates at t = ',
 num2str(t(end))])
    plot(t,y)
end
hold off
axis([0 5 -5 5])
title 'Solutions for b from -5 to 5 with 0.5 steps'
% (b)

% The plot indicates that if b = -1, the corresponding solution is y =
 -1
% If b is in the interval (-inf, -1), the solution decreases and
 converges
% to -1. If b is in the interval (-1, 1) the solution initially
 increases
% but starts decreasing and it converges to -1. If b = 1, the
 corresponding
% solution is y = 1. Finally, when b is in the interval (1,inf), the
% solution first decreases but then increases to +inf.

% (c)
figure
for b = -5:0.5:5
    [t,y] = ode45(f,[0,5],b);
    plot(t,y)
end
hold on
axis([0 5 -5 5])
hold on
```

```
ezplot('t.^0.5', [0 5])
title 'combined plots'

% This is the combined graph of part (a) and y = t.^0.5

% (d)
figure
for b = -5:0.5:5
    [t,y] = ode45(f,[0,5],b);
    plot(t,y)
end
hold on
axis([0 5 -5 5])
hold on
ezplot('t.^0.5',[0,5])
% In the interval (1, inf), the solution curves are asymptotic to y=
 t^0.5. This is because along
% the line y= t^0.5, y' = 0
% above the line y= t^0.5, y' is negative and below the line
% y=t^0.5, y' is positive.
% Therefore the solutions are pushed towards the line y=t^0.5

[T,Y] = meshgrid([0:.2:5], [-2:.2:5]);
S = (Y - T.^0.5).*(1 - Y.^2);
L = sqrt(1+S.^2);
quiver(T,Y,1./L,S./L,0.5)
axis([0 5 -2 5])
title 'Combined graph with direction field'
```

```
For b = -5 integration terminates at t = 5
For b = -4.5 integration terminates at t = 5
For b = -4 integration terminates at t = 5
For b = -3.5 integration terminates at t = 5
For b = -3 integration terminates at t = 5
For b = -2.5 integration terminates at t = 5
For b = -2 integration terminates at t = 5
For b = -1.5 integration terminates at t = 5
For b = -1 integration terminates at t = 5
For b = -0.5 integration terminates at t = 5
For b = 0 integration terminates at t = 5
For b = 0.5 integration terminates at t = 5
For b = 1 integration terminates at t = 5
For b = 1.5 integration terminates at t = 5
For b = 2 integration terminates at t = 5
For b = 2.5 integration terminates at t = 5
For b = 3 integration terminates at t = 5
For b = 3.5 integration terminates at t = 5
For b = 4 integration terminates at t = 5
For b = 4.5 integration terminates at t = 5
For b = 5 integration terminates at t = 5
```


Solutions for b from -5 to 5 with 0.5 steps
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


combined plots
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
t

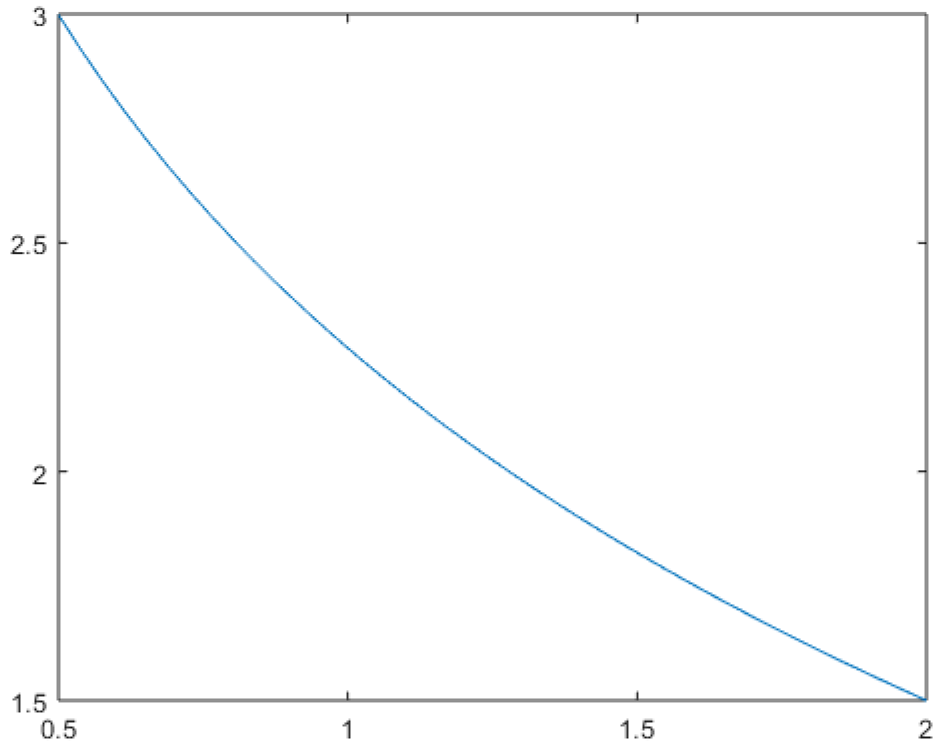


QUESTION

```
%7.a
syms t y
sol=ode45(@(t,y) -exp(y)/((t*exp(y))-sin(y)), [2 0.5], 1.5)
deval(sol, 1)
deval(sol, 1.5)
sol1=ode45(@(t,y) -exp(y)/((t*exp(y))-sin(y)), [2 4], 1.5)
deval(sol1, 3)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
figure (1)
[t, y]=ode45(g, [2 0.5], [1.5]);
plot(t, y)
```

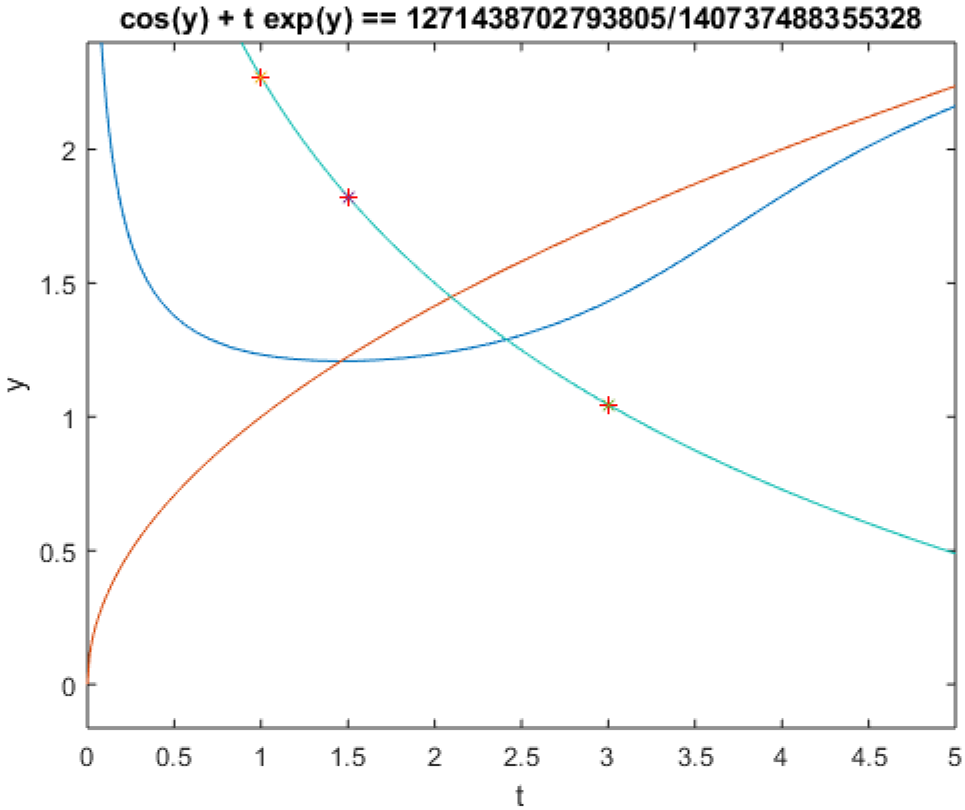
```
sol =

  struct with fields:

     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×11 double]
          y: [1×11 double]
      stats: [1×1 struct]
      idata: [1×1 struct]
```

```
ans =

    2.2698


ans =

    1.8228


sol1 =

  struct with fields:

     solver: 'ode45'
    extdata: [1×1 struct]
          x: [1×11 double]
          y: [1×11 double]
      stats: [1×1 struct]
      idata: [1×1 struct]


ans =

    1.0453
```

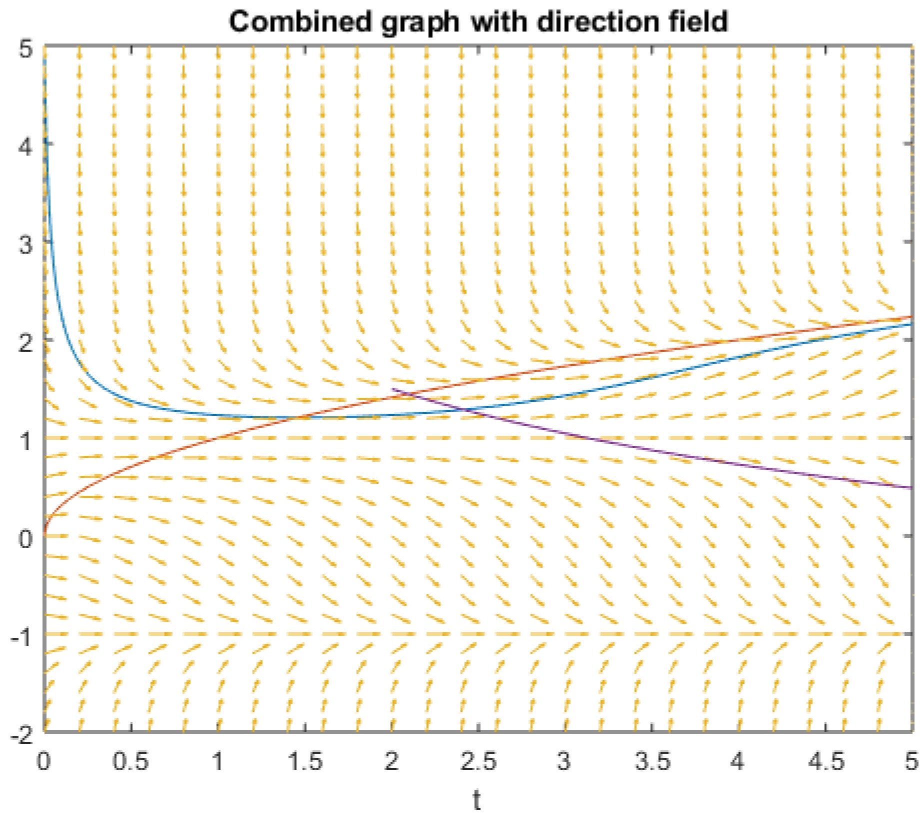



```
%7.b
syms y t
figure (2)
eq=cos(y) + t*exp(y) == cos(3/2) + 2*exp(3/2);
ezplot(eq, [0, 5])
hold on
plot(1, solve(subs(eq, t, 1), y), 'r*')
plot(1.5, solve(subs(eq, t, 1.5), y), 'r*')
plot(3, solve(subs(eq, t, 3), y), 'r*')
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [0.5 4], [1.5]);
plot(1, 2.2698, 'x')
plot(1.5, 1.8228, 'x')
plot(3, 1.0453, 'x')
hold off

%The actual solution is very similar to the numerical solution.The x's
 fall
%almost directly on top of the *'s.
```

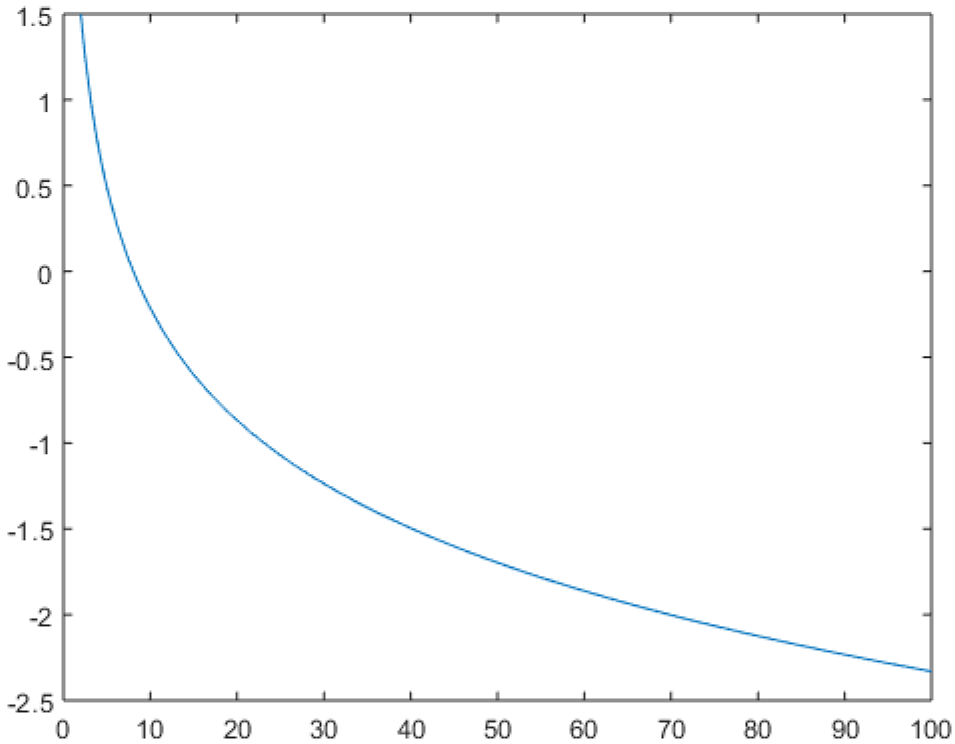
Warning: Unable to solve symbolically. Returning a numeric approximation instead.
Warning: Unable to solve symbolically. Returning a numeric approximation instead.
Warning: Unable to solve symbolically. Returning a numeric approximation instead.



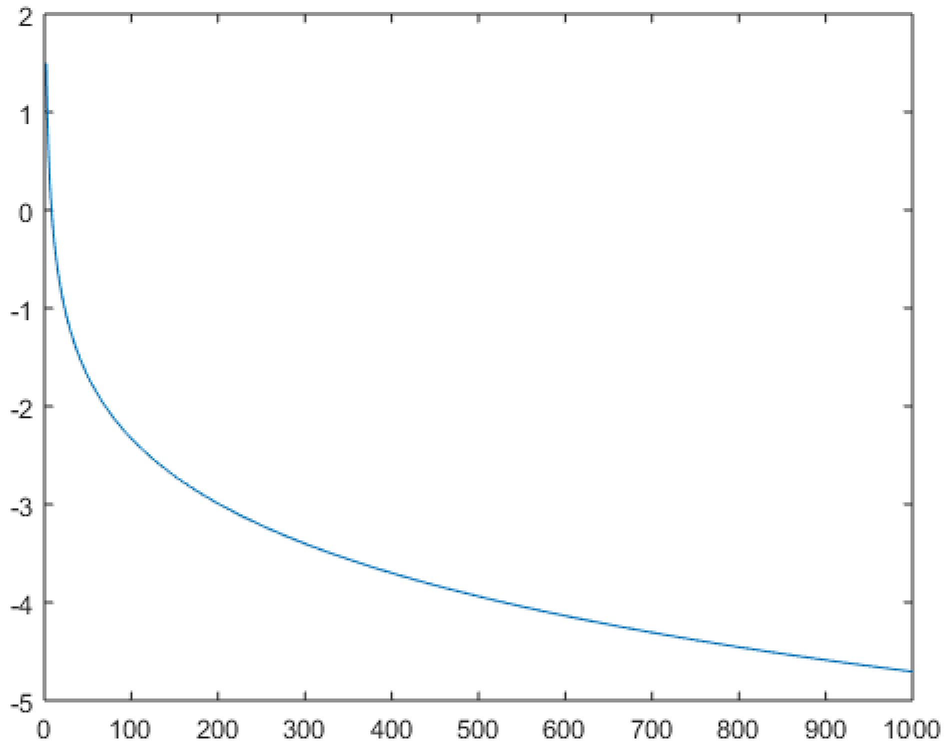

```
%7.c
%interval 2<t<10
figure (3)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [2 10], [1.5]);
plot(t, y)
```



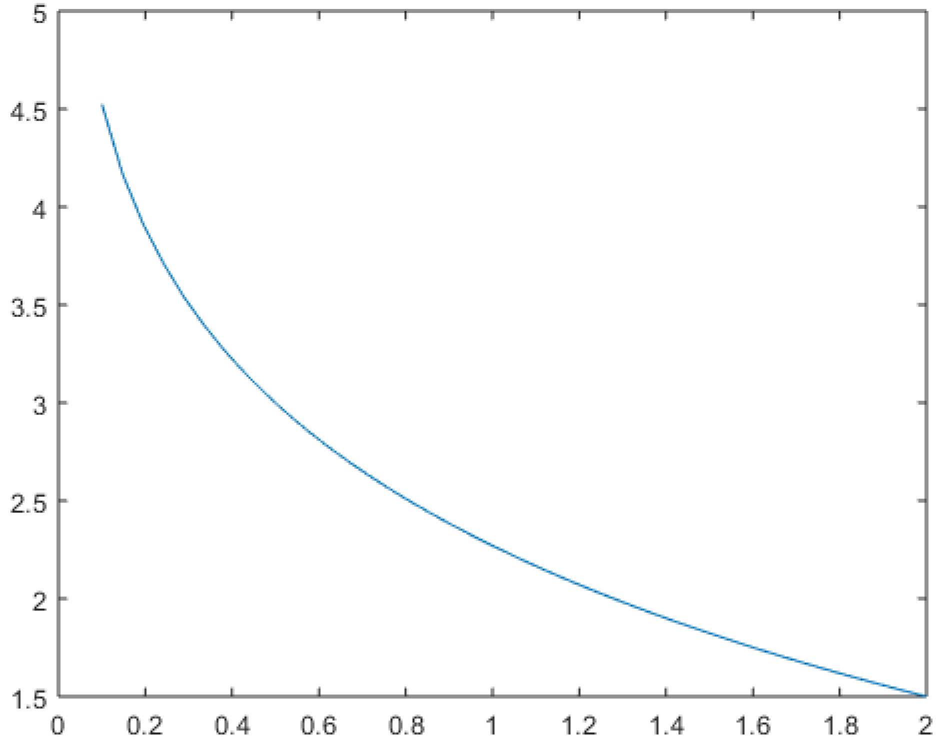

```
%7.c
%interval 2<t<100
figure (4)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [2 100], [1.5]);
plot(t, y)
```



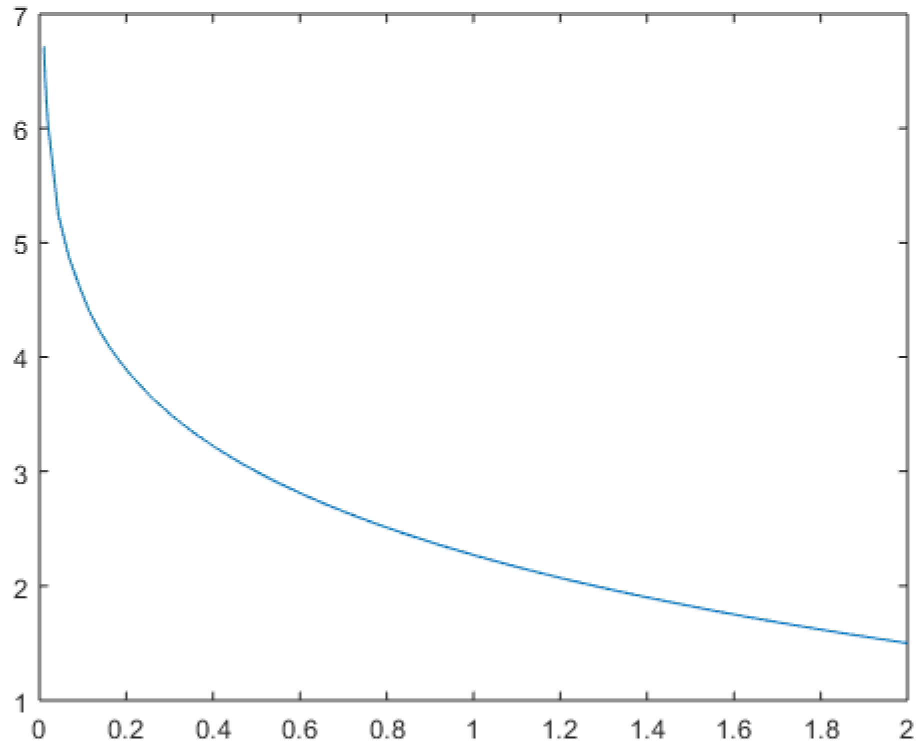

```
%7.c
%inerval 2<t<1000
figure (5)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [2 1000], [1.5]);
plot(t, y)


%As t approaches infinity, the solutions continuously decrease. T is
 in
%the denominator of the differential equation meaning that as t
 increases,
%the differential equation decreases.
```



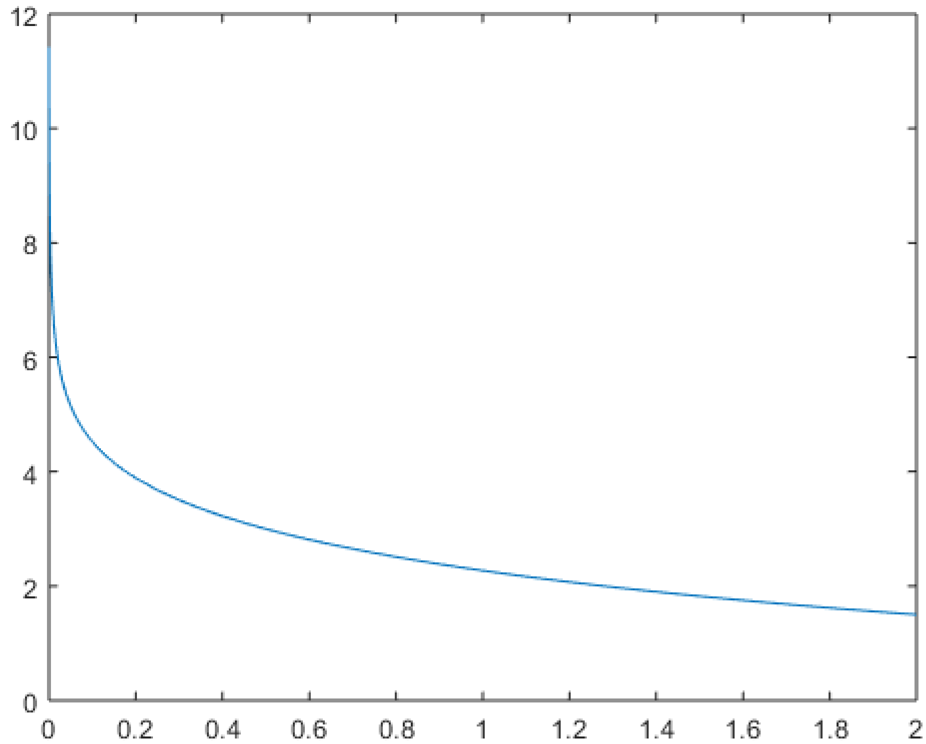

```
%7.c
%interval 0.1<t<2
g = @(t,y) -exp(y)./(t.*exp(y)-sin(y));
figure (6)
[t, y]=ode45(g, [2 0.1], [1.5]);
plot(t, y)
```




```
%7.c
%interval 0.01<t<2
figure (7)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [2 0.01], [1.5]);
plot(t, y)
```




```
%7.c
%interval 0.0001<t<2
figure (8)
g=@(t, y) -exp(y)./((t.*exp(y))-sin(y));
[t, y]=ode45(g, [2 0.0001], [1.5]);
plot(t, y)

%as t approaches 0, the solution approaches infinity
```

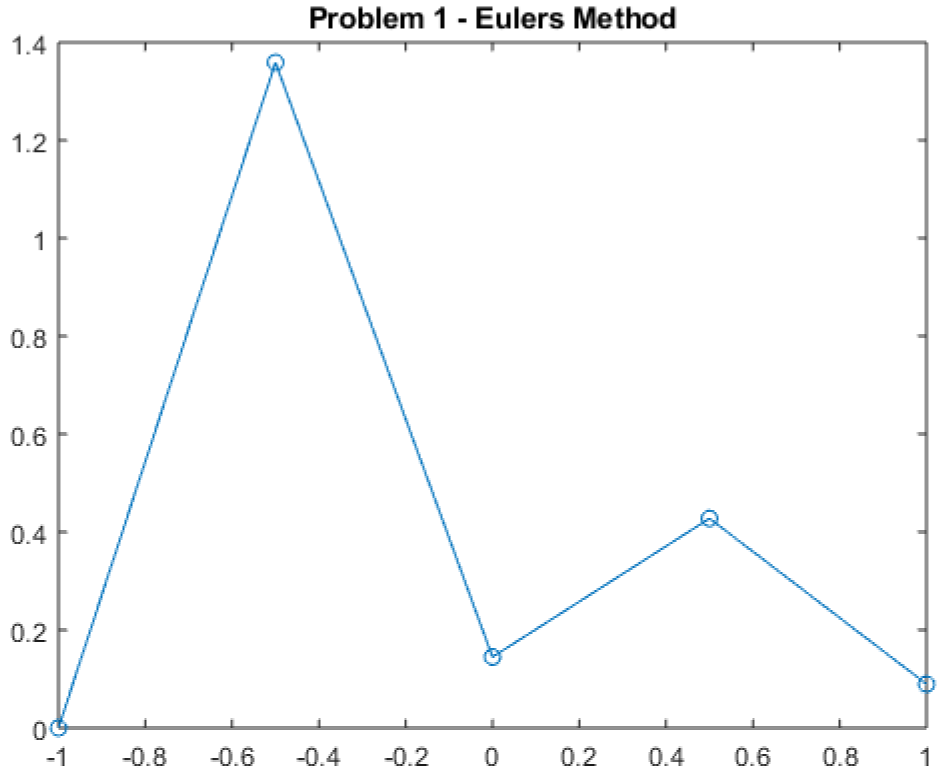


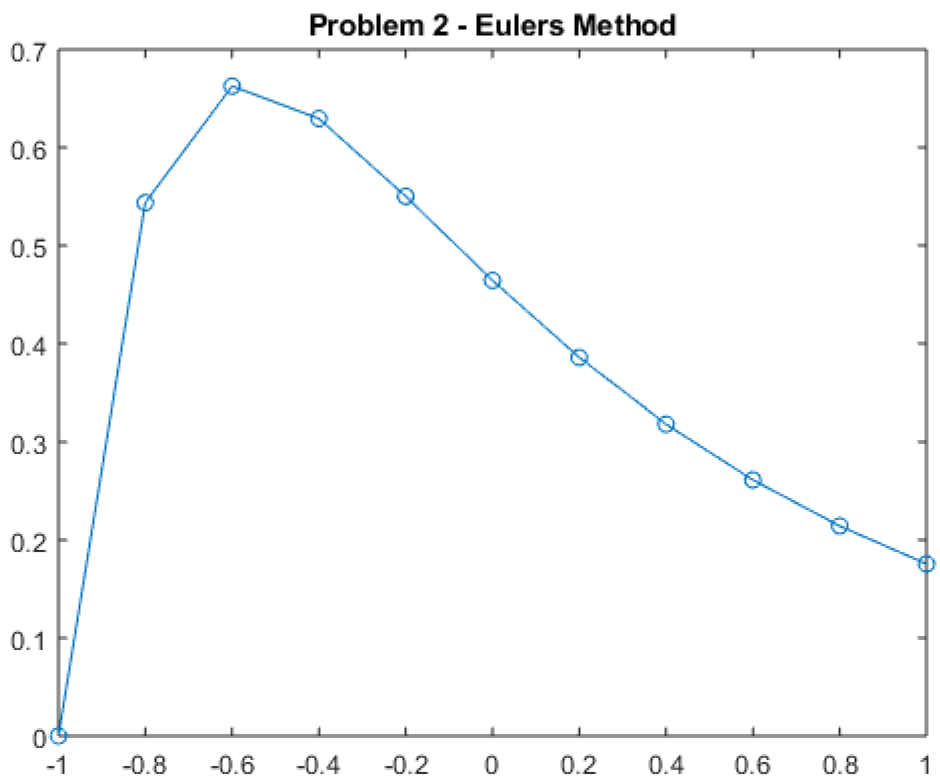
QUESTION

(a)

```
syms t y
f = @(t,y) exp(-t) - 3.*y
[t,y] = myeuler(f,-1,0,0.5*4-1,4);
figure
%plot([t,y])
plot(t,y,'-o')
title 'Problem 1 - Eulers Method'
[t,y] = myeuler(f,-1,0,0.2*10-1,10);
figure
%plot([t,y])
plot(t,y,'-o')
title 'Problem 2 - Eulers Method'
hold off
```

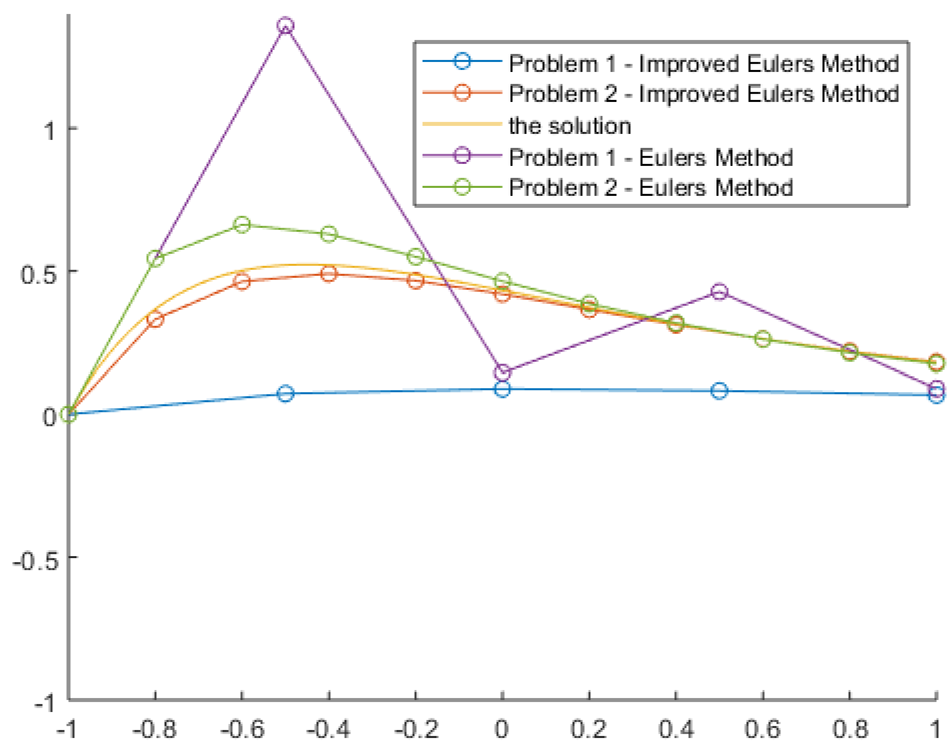
```
f =

  function_handle with value:

    @(t,y)exp(-t)-3.*y
```


Problem 1 - Eulers Method
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Problem 2 - Eulers Method
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

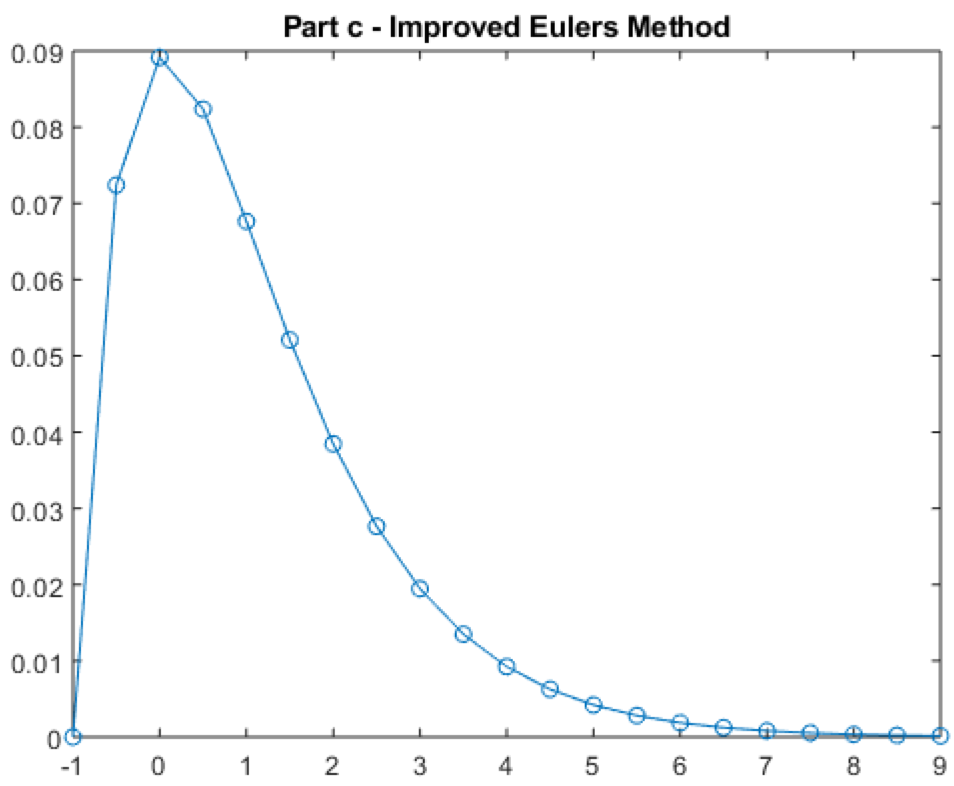
This is for part (b)

```
[t,y] = RungeTrap(f,-1,0,0.5*4-1,4);
figure
hold on
plot(t,y,'-o')
[t,y] = RungeTrap(f,-1,0,0.2*10-1,10);
plot(t,y,'-o')

fplot(dsolve('Dy = exp(-t)-3*y','y(-1)=0','t'),[-1 1]);
[t,y] = myeuler(f,-1,0,0.5*4-1,4);
plot(t,y,'-o')
[t,y] = myeuler(f,-1,0,0.2*10-1,10);
plot(t,y,'-o')

hold off
legend('Problem 1 - Improved Eulers Method','Problem 2 - Improved
 Eulers Method','the solution','Problem 1 - Eulers Method','Problem 2
 - Eulers Method')
axis([-1 1 -1 1.4])
```




```
%c
% The exact solutions are more accurate when compared to the numerical
% approximations. This is also because we had a small step size. We
 can
% make a more reliable prediction about the long-term
% behaviour if the step size is decreased and the number of steps are
```

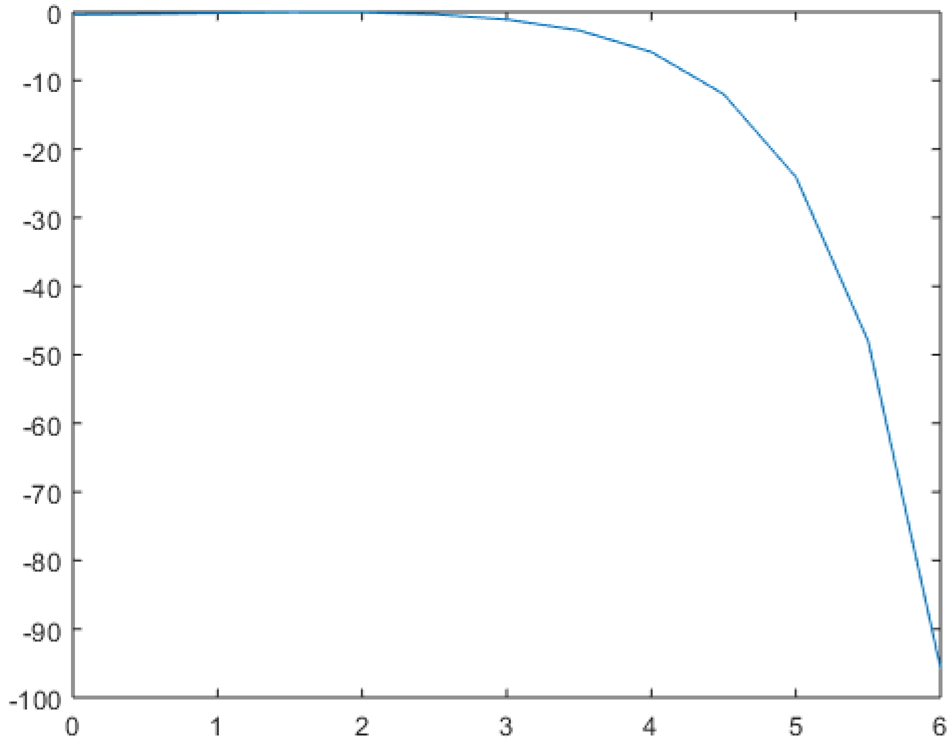
```
% increased. We can also observe that as t increases, the
 approximation
% gets more precise. Since the partial derivative f(t,y) with respect
 to y
% is negative, the equation is stable. When we approximate the
 solution of
% a stable function, the error decreases as t increses.

% Plotting the solution
ode45(f,[-1, 9], 0)
% Here n = 20 since n = (tF-tI)/h
[t,y] = myeuler(f,-1,0,9,20);
figure
plot(t,y,'-o')
%axis([0 12 -1 2])
title 'Part c - Eulers Method'

[t,y] = RungeTrap(f,-1,0,9,20);
figure
plot(t,y,'-o')
% axis([0 12 -1 2])
title 'Part c - Improved Eulers Method'
```



QUESTION

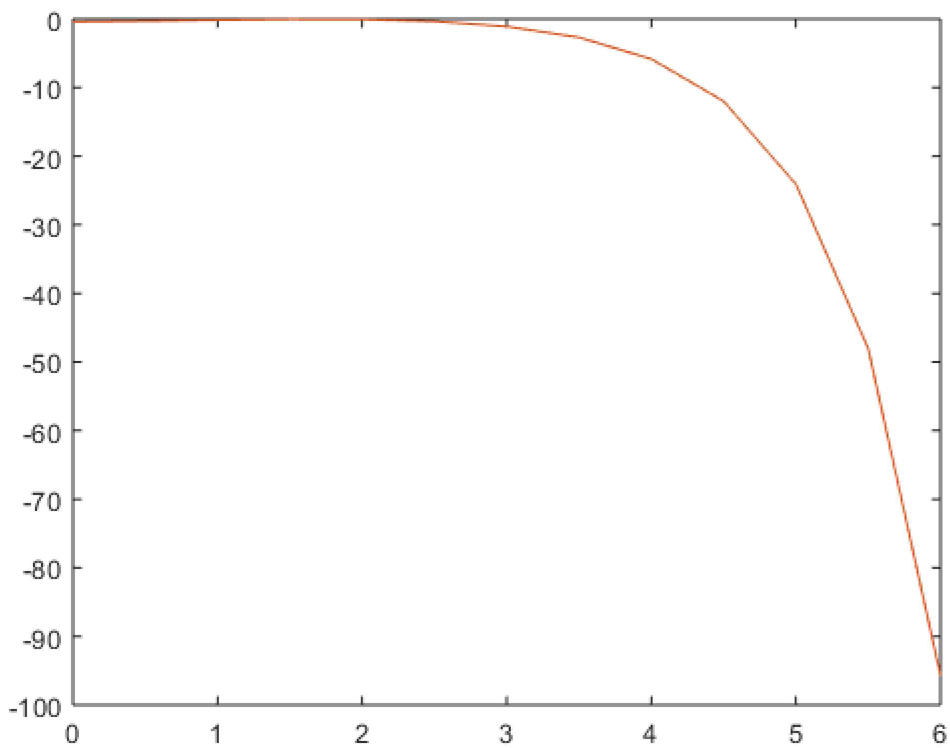
```
%15.a
```

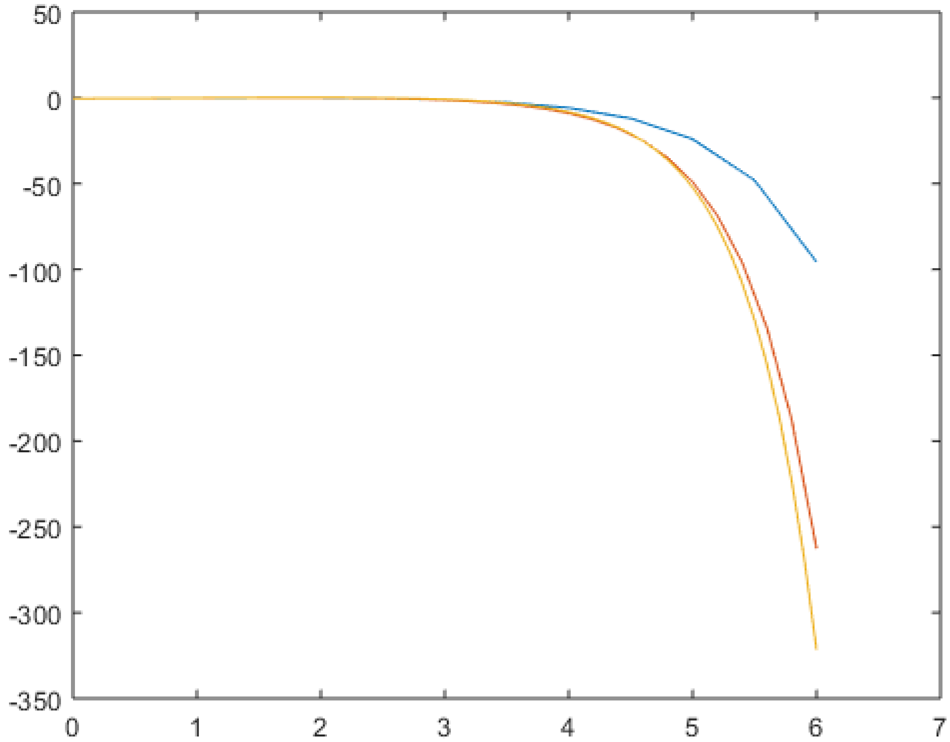
```
f=@(t,y) 2*y+cos(t);
[t,y] = myeuler(f,0,-2/5,6,12);
figure (10)
plot(t,y)
%y appears to be an exponentially decreasing function of t that
 approaches
%negative infinity as t approaches infinity.
```




```
hold on
f = @(t,y) 2*y+cos(t);
[t,y] = myeuler(f,0,-2/5,6,12);
figure (10)
plot(t,y)
%15.b
%n=30, h=0.2
f = @(t,y) 2.*y+cos(t);
[t1,y1] = myeuler(f,0,-2/5,0.2*30,30);

%15.b
%n=60, h=0.1
f = @(t,y) 2.*y+cos(t);
[t2,y2] = myeuler(f,0,-2/5,0.1*60,60);

%15.b
[t1,y1] = myeuler(f,0,-2/5,0.2*30,30);
[t2,y2] = myeuler(f,0,-2/5,0.1*60,60);
```

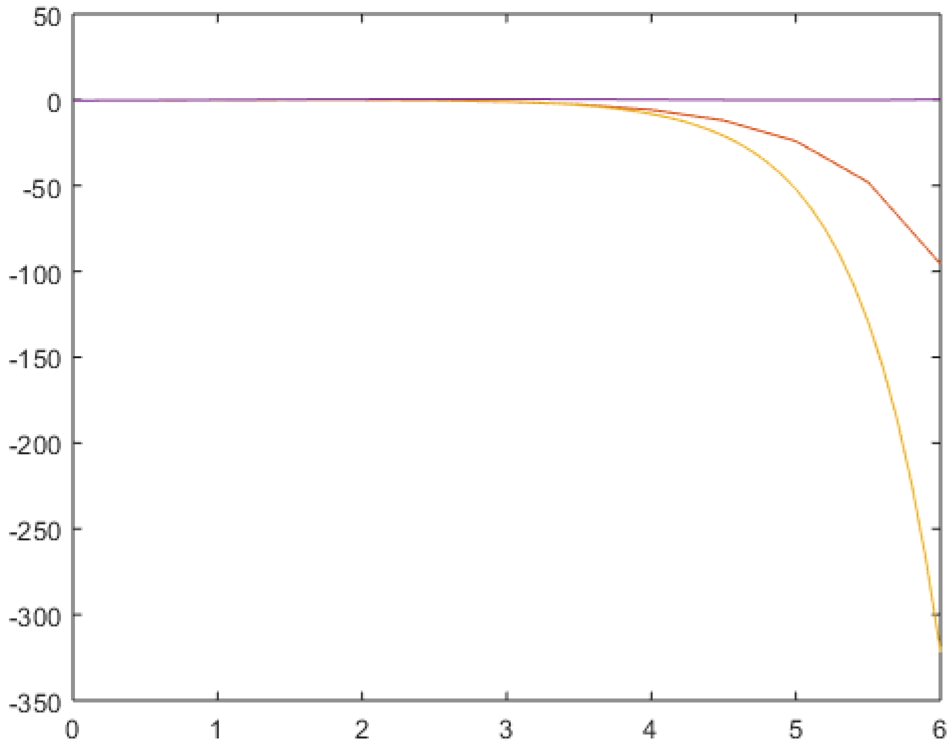
```
figure
plot(t,y)
hold on
plot(t1,y1)
hold off
hold on
plot(t2, y2)
hold off
%As the step size decreases, y exponentially decreases at a faster
 rate.
%y approaches negative infinity as t approaches infinity.
```

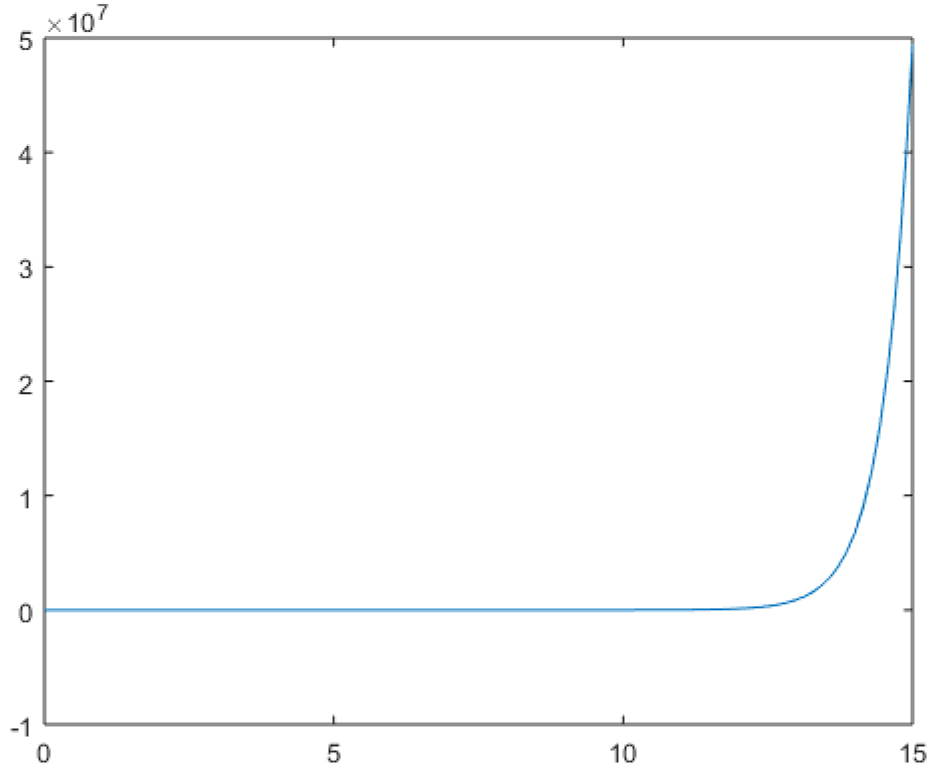




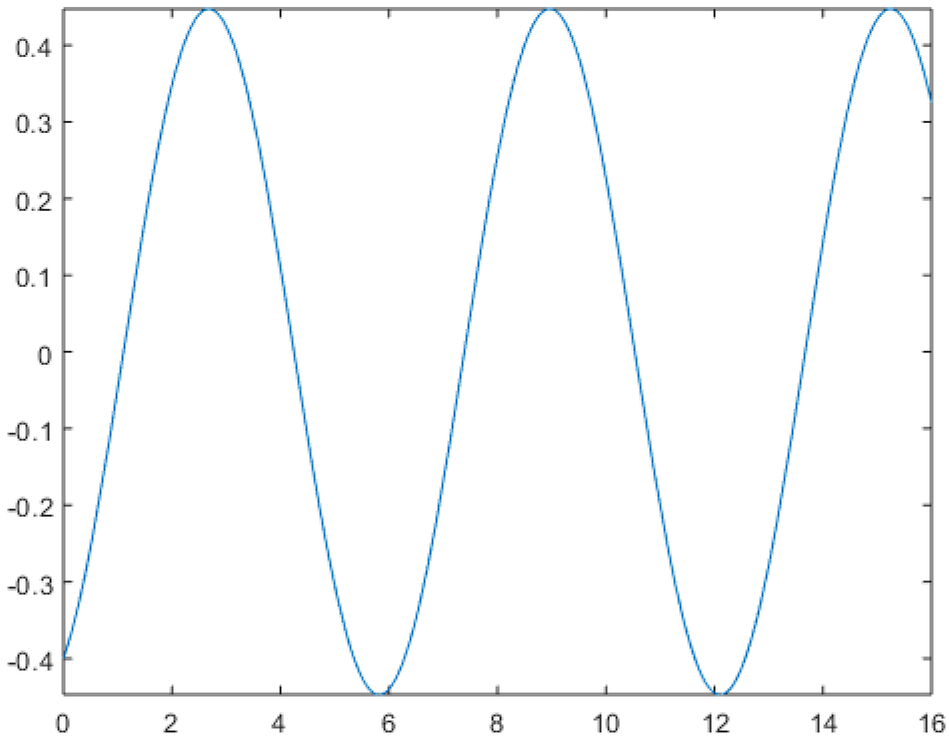

```
%15.c
syms t y
f=@(t, y) 2*y+cos(t);

[t,y]=myeuler(f,0,-2/5,6,12);
[t1,y1] = myeuler(f,0,-2/5,0.2*30,30);
[t2,y2] = myeuler(f,0,-2/5,0.1*60,60);

[t3, y3]=ode45(f, [0 6], -2/5);
figure
plot(t3, y3)
hold on
plot(t, y)
hold off
hold on
plot(t2, y2)
hold off
hold on
plot(t3, y3)
hold off

[t4,y4]=ode45(f,[0 15],-2/5);
figure
plot(t4, y4)
%Y looks as if it approaches infinty as t increases
%When it is plotted on the same set of axes, the ode45 solution
```

```
%looks constant because of the scale of the y axis from the Euler
%solutions
```



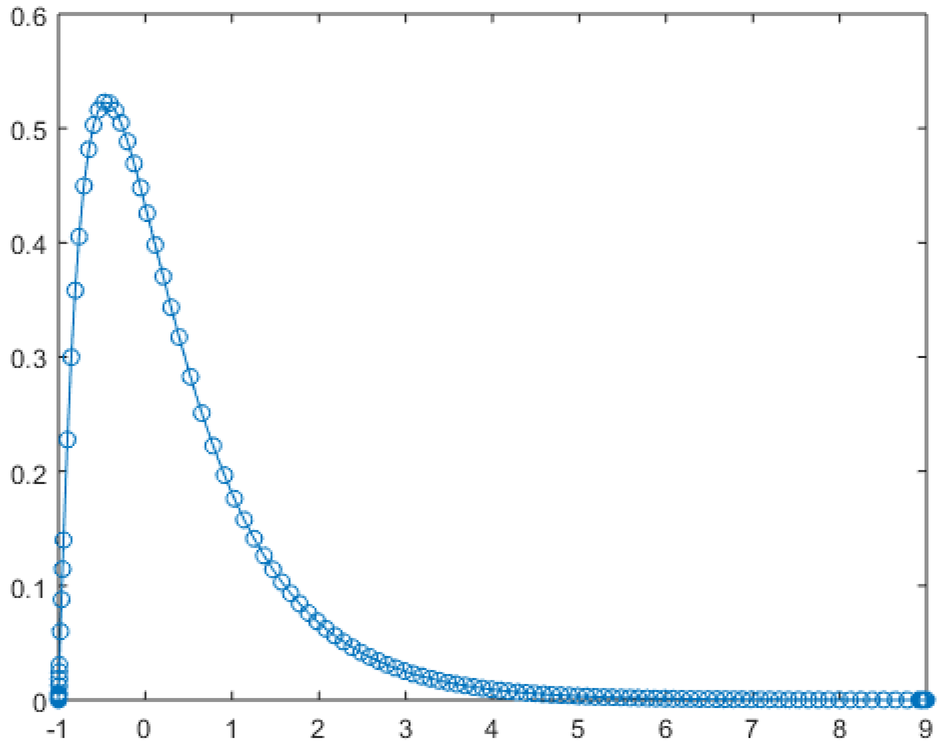


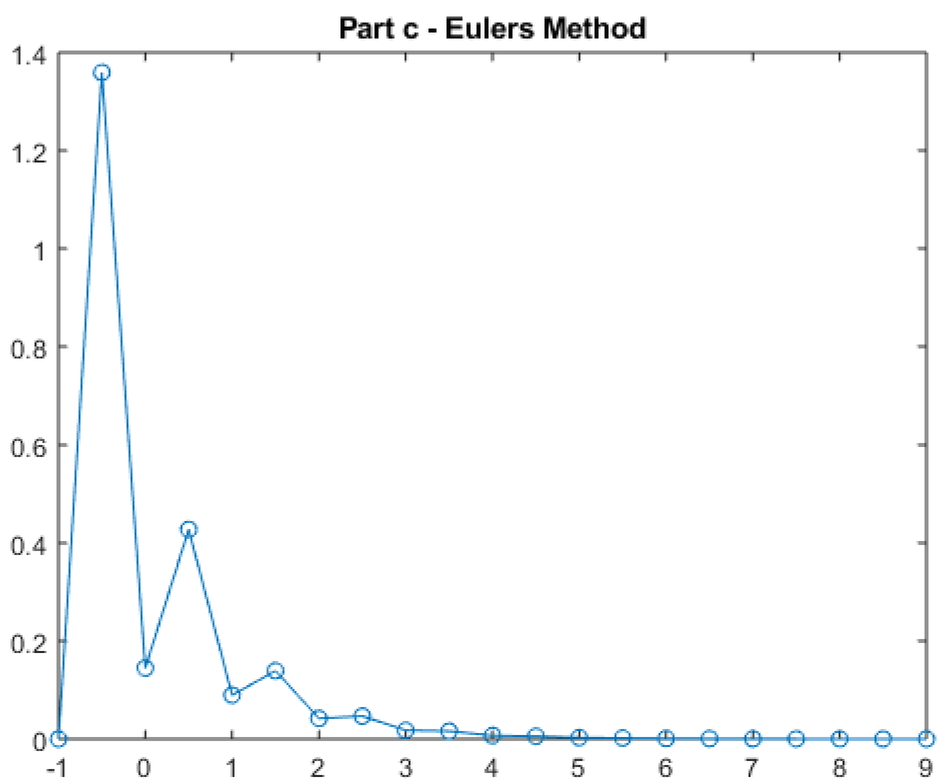

```
%15.d
syms t y
sol=dsolve('Dy=2*y+cos(t)','y(0)=-2/5','t')
figure
fplot(t,sol, [0,16])

%the exact solution and approximations found above are different.
%The exact solution is an oscillating function of t.
%if c=0, the exponential C*exp(2t) goes away from the general solution
%and the solution then becomes an oscillating solution.
%This solution is unstable, since C*exp(2t)is not zero when C is
%not zero, which is every case excluding
%y(0) = -2/5. This explains why every inexact numerical
%approximation will have significant error since C*exp(2t) predicts
%the large t behavior of the function whenC is not zero.


sol =

-(5^(1/2)*cos(t + atan(1/2)))/5
```



This is regular Euler's method

```
function [t,y] = myeuler(f, tinit, yinit, b, n)
h = (b - tinit)/n;
t = zeros(n+1,1);
y = zeros(n+1,1);
t(1) = tinit;
y(1) = yinit;
for i = 1:n
    ti = t(i);
    t(i + 1) = ti + h;
    yi = y(i);
    y(i + 1) = yi + h*f(ti, yi);
end
end
```


0.6
0.5
0.4
0.3
0.2
0.1
0
-1
0
1
2
3
4
5
6
7
8
9


Part c - Eulers Method
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-1
0
1
2
3
4
5
6
7
8
9

This is improved Euler's method. (Also known as Runge Trapezoidal)

```
function [t,y] = RungeTrap(f, tI, yI, tF, N)
t = zeros(N + 1, 1); y = zeros(N + 1, 1);
t(1) = tI; y(1) = yI; h = (tF - tI)/N; hhalf = h/2;
for j = 1:N
t(j + 1) = t(j) + h;
fnow = f(t(j), y(j));
yplus = y(j) + h*fnow; fplus = f(t(j + 1), yplus);
y(j + 1) = y(j) + hhalf*(fnow + fplus);
end
end
```

Published with MATLAB® R2017b