Math 4540/MSSC 5540 - Homework #2

Henri Medeiros Dos Reis

November 12, 2023

Directions. All work is to be done in complete sentences. Assignments must be stapled. Each problem (1., 2., ... not i), ii) ...) must begin on a separate sheet of paper. Please USE THE BACK. Please begin each problem with the full statement of the problem. While you are encouraged to work through confusion with your classmates, your work must be written in your own words. The assignment is due by 5:00 pm on Wednesday, November 15, 2023 in Dr. Clough's mailbox outside CU 340 in the narrow dark hallway.

1. Consider the bungee jumper using a strong cord.

   i-) Write down the equation of motion for the bungee jumper.

   $$\frac{dv}{dt} = g - \frac{C_{drag}}{m}v^2, v(0) = 0$$

   ii-) Use the gravitational constant g = 9.81 m/s. Assume a maximum weight of 350 lb, a rope of length 30 m, an initial height of 500 m, and a coefficient of drag of 0.25 kg/m. Show that the units of the equation are balanced.

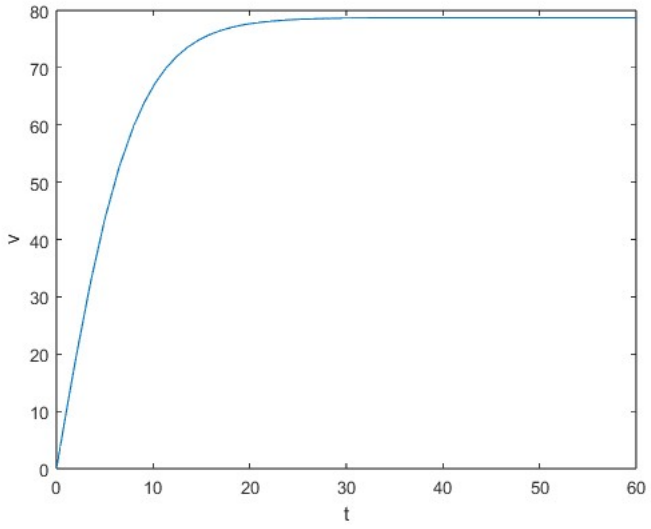
   Since the gravitational constant is in meters/seconds^2, $\frac{C_{drag}}{m}v^2$ is Kilograms/meters*(meters/second)^2/Mass, which when simplifying gives meters/seconds^2. Which is the same units we have in the left hand side of the equation, as long as we convert the weight from pounds to kilograms.

   iii-) Determine the terminal velocity of the jumper.

   $$\begin{aligned} 0 &= g - \frac{C_{drag}}{m}v^2 \\ v_{final} &= \sqrt{\frac{gm}{C_{drag}}} \\ v_{final} &= \sqrt{\frac{9.81 * 350/2.2}{0.25}} \\ v_{final} &= 79.01 \end{aligned}$$

iv-) Solve the ODE in Matlab using ode45. Plot the velocity as a function of time. Include the graph here.

```
m = 158;
c_drag = 0.25;
g = 9.81;

f = @(t,v) g-(c_drag./m).*v.^2;

tspan = [0 60]; %time interval
v0 = 0; % Initial velocity

%%RK
[t,v] = ode45(f,tspan,v0);

%%
plot(t,v)
xlabel('t')
ylabel('v')
```



v-) Is the terminal velocity correct?
Yes, it is not exact, but very close. At the last v, we get 78.9279.

vi-) How long does it take the jumper to reach a speed of 80 m/s?
The jumper never reaches this speed if the jumper weights $350 pounds = 350/2.20462 kg = 158.757 kg$

2. Consider the bungee jumper using a stretchable bungee cord.

i-) Write down the equation of motion for the bungee jumper.

$$\frac{d^2y}{dt^2} = g - sign(v)a_{drag}v^2/mass - F_{cord}/mass$$

Where

$$F_{cord} = \begin{cases} a_{spring}(y-L) + a_{damp}v & \text{if } y > L \\ 0 & \text{else} \end{cases}$$

ii-) Assume a spring constant of k = 40 N/m and a coefficient of damping of 8 N s/m. Show that the units of the equation are balanced.

Writing only the units for the equation gives

$$\begin{aligned} \frac{d^2y}{dt^2} &= \frac{m}{s^2} - \frac{kg/m(m/s)^2}{kg} - \frac{N/m(m-m) + Ns/m(m/s)}{Kg} \\ &= \frac{m}{s^2} - \frac{m}{s^2} - \frac{N+N}{Kg} \\ &= \frac{m}{s^2} - \frac{m}{s^2} - \frac{Kgm/s^2 + Kgm/s^2}{Kg} \end{aligned}$$

iii-) Write the equation as a system of ODEs.

Let

$$\begin{aligned} y_1 &= y \\ y_2 &= v = \frac{dy}{dt} = y_1' \end{aligned}$$

Then the system is

$$\begin{cases} y_1' &= y_2 \\ y_2' &= y'' \end{cases}$$

$$\begin{cases} y_1' &= y_2 \\ y_2' &= g - sign(v)a_{drag}y_2^2/mass - F_{cord}/mass \end{cases}$$

iv-) Solve the ODEs in Matlab using ode45. Plot the height above the ground as a function of time. How close to the ground does the jumper get? Include your code.

```
%% RK45 ODE solver
clear all; close all;
global a_damp a_spring a_drag g L mass
tspan = [0 100];
%parameters
```

```
a_damp = 8; a_spring=40; a_drag = 0.25; g = 9.81; L =
   30;
mass = 158.757;init_height = 500;
xinit =0; yinit = 0;
ICs = [xinit yinit];
%% RK
[t,y] = ode45(@(t,y) bungee(t,y),tspan,ICs);

%% Graph
plot(t,-y(:,1)+init_height)
hold on
%%
function dydt = bungee(t,y)
global a_damp a_spring a_drag g L mass
q = y(2);
if y(1) > L       %Cord exerts force
    F_cord = a_spring*(y(1) - L) + a_damp*y(2);
else
    F_cord = 0;
end

TotalForce = g - sign(y(2))*a_drag*y(2)^2/mass - F_cord
   /mass;
r = TotalForce;
dydt = [q;r];
end
```

The jumper stays 382.5746 meters of the ground

3. Now assume that we want to position the platform in the tree as close as possible to the ground while always maintaining at least an 8 m height above the ground for the jumper.

   i-) How high does the platform need to be? Explain or show how you obtained your solution.

   $$500 - 382.5746 + 8 = 125.4254$$

   So we should place at 126m.

   ii-) Now consider that we will also build a bungee jump for kids with maximum weight 75 pounds. How high does the platform need to be? Explain or show how you obtained your solution.

Change the 158.757 kg to 34.0194. We obtain that if the a kid jumps from 126m, the lowest it reaches is 72.8136. So $126 - 72.8136 + 8 = 61.1864$, then place at 62m.

4. Consider the Lorenz system of differential equations. Use s = 10, r =28, and b = 8/3.

   i-) Determine whether the system of equations has any equilibrium points.

   In order to find the equilibrium points, we need to set all left hand sides of the equations to 0 and solve for what the variables should be.

$$\frac{dx}{dt} = -sx + sy, \frac{dy}{dt} = -xz + rx, \frac{dz}{dt} = xy - bz$$

   Need $x = y$

$$\frac{dz}{dt} = 0 = x^2 - bz \Rightarrow z = x^2/b$$

$$\frac{dy}{dt} = 0 = -\frac{x^3}{b} + rx - x$$

$$x = +- \sqrt{b(r-1)} = y$$

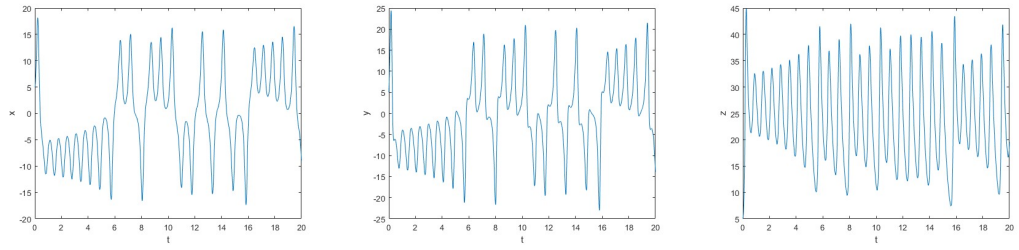
$$z = \frac{+- b(r-1)}{b}$$

   ii-) Construct matlab code to solve the system on t = [0,20] using ode45. Check that your code is working correctly by using initial conditions = equilibrium points.

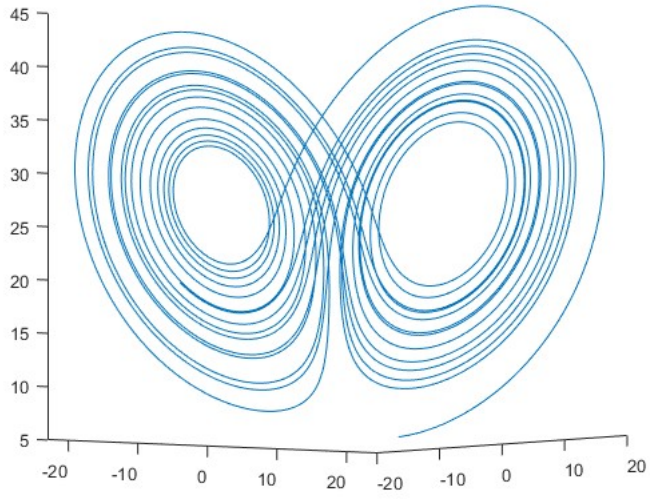
```
%% RK45 ODE solver
clear all; close all;
global s r b
tspan = [0 20];
%parameters
s = 10;r=28;b=8/3;
x_init = -sqrt(b*(r-1));
y_init = x_init;
z_init = r-1;
vars = [x_init,y_init,z_init];

options = odeset('RelTol',1e-8,'AbsTol',1e-8);
[t,Y] = ode45(@(t,y) lorenz(t,y),tspan,vars,options);

function ddt = lorenz(t,Y)
global s r b
    x = Y(1);
    y = Y(2);
    z = Y(3);
```

```
    dxdt = -s*x+s*y;
    dydt = -x*z+r*x-y;
    dzdy = x*y-b*z;
    ddt = [dxdt,;dydt;dzdy];
end
```

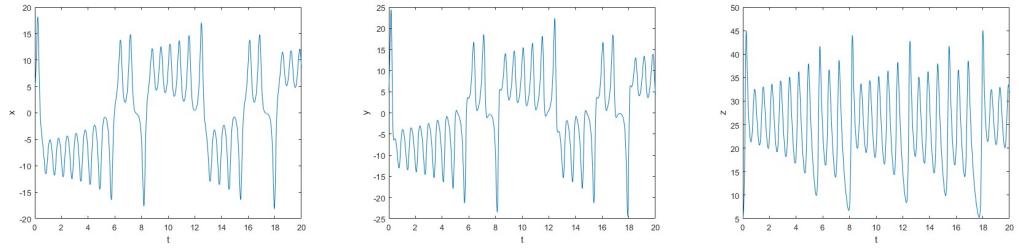
iii-) Now use x(0) = y(0) = z(0) = 5. Generate plots of (t,x), (t,y), and (t,z). Label the axes. Submit graphs and code

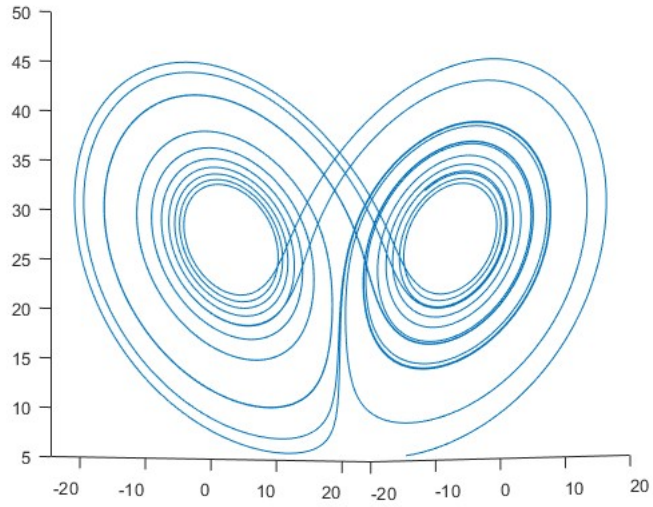


iv-) Plot the phase portrait for the system. Suppose [t,Y] is the output from ode45. Use plot3(Y(:,1),Y(:,2),Y(:,3)) to generate the plot. Submit plot. v.)

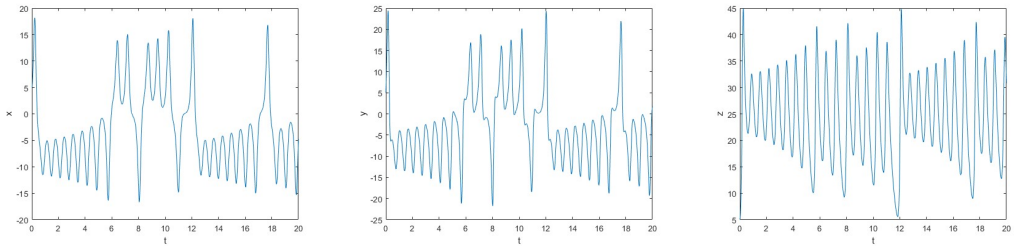


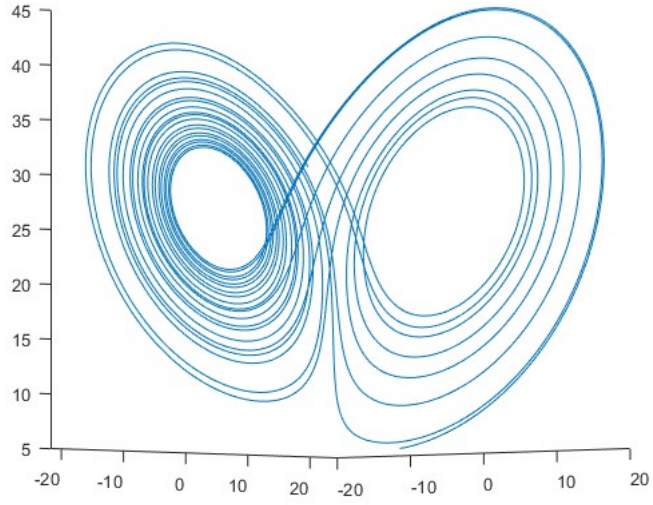
v-) Repeat 3. and 4. Using an initial value of 5.01, 5.001, 5.0001, and 5.00001.



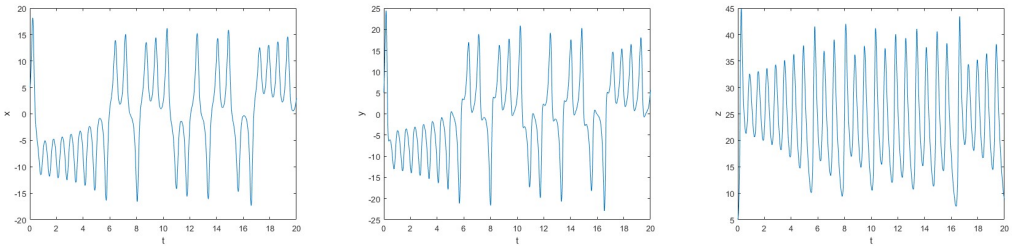

50
45
40
35
30
25
20
15
10
5
-20
-10
0
10
20
-20
-10
0
10
20

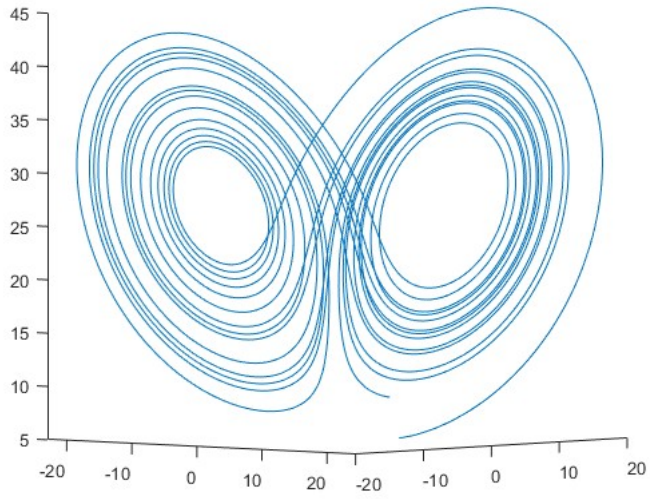
5.001:


x
t
y
t
z
t

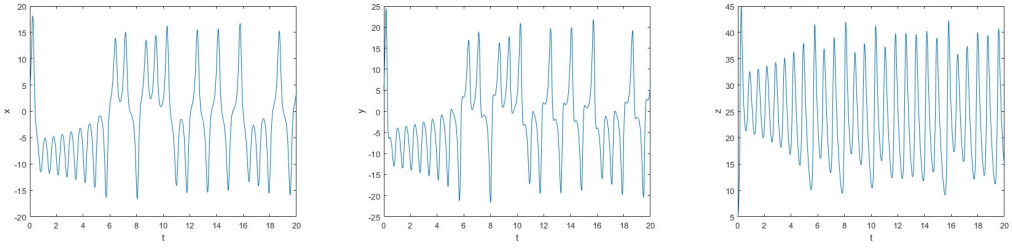

45
40
35
30
25
20
15
10
5
-20
-10
0
10
20
-20
-10
0
10
20

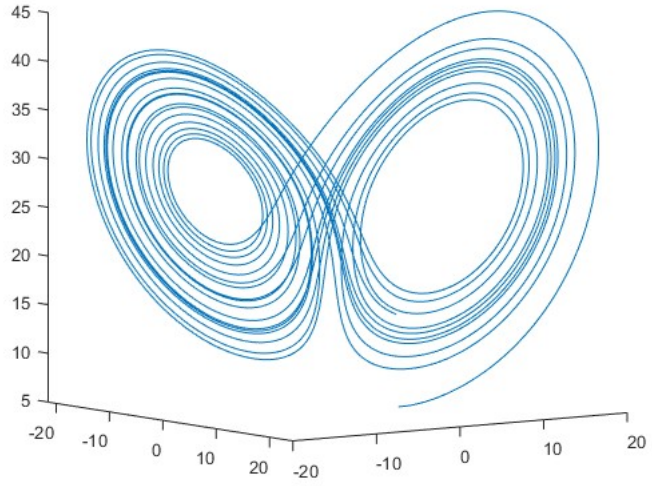
5.0001:





5.00001:





vi-) Describe the behavior you are seeing. Do you think the behavior is because of numerical error or the dynamics of the system? What conclusions can you draw about weather prediction?

Any small change to the initial values changes the output by a lot, solutions are unstable. There is not a smooth transition between input and output. I believe this is due to the system, and it implies that weather prediction is only reliable for small values of t. Since we can see the system getting chaotic after t around 12

5. ACTIVITY 9: Stiff differential equations

i-) Consider the differential equation $\frac{dy}{dt} = 10\,(1-y)\,,\ y\,(0) = \frac{1}{2}$. This is a linear first order equation. **Solve the problem analytically using an integrating factor. Sketch in the solution to the problem.**

$$\frac{dy}{dt} = 10 - 10y, y(0) = \frac{1}{2}$$
$$\frac{dy}{dt} + 10y = 10$$
$$IF = e^{10t}$$
$$\int \frac{d}{dt} e^{10t} y = 10e^{10t} y + \frac{dy}{dt} e^{10t}$$
$$\int \frac{d}{dt} e^{10t} y = \int 10e^{10t}$$
$$e^{10t} y = \frac{10e^{10t}}{10} + c$$
$$y = 1 + ce^{-10t} \quad \text{plugging initial}$$
$$1/2 = 1 + c$$
$$c = \frac{-1}{2} \Rightarrow y = 1 - 1/2e^{-10t}$$

ii-) Consider solving this problem using Euler's method. Then we have

$$y_{i+1} = y_i + h\ 10\ (1 - \ y_i) = y_i\,(1 - 10h) + 10h$$

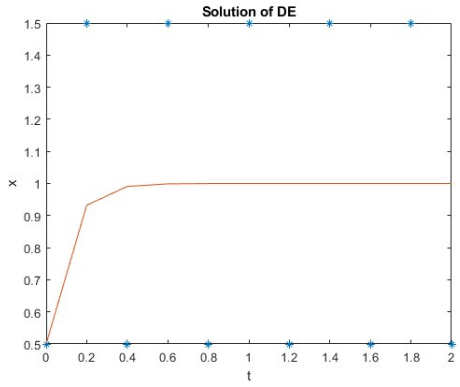
Note that if the method were converging, we would have $y_{i+1} \approx y_i = w$ say.
So the problem looks like $w = w\,(1 - 10h) + 10h$, which is a fixed point problem, $w = g\,(w)$. Provide a check that the fixed point is w = 1

$$1 = 1(1 - 10h) + 10h = 1$$

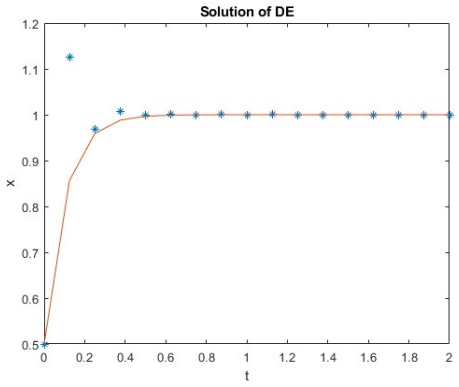
iii-) We know that fixed point iteration will converge for $|g'\,(1)| < 1$. Determine the values of h for which this inequality is true.

$$\begin{aligned} g(w) &= w(1 - 10h) + 10h \\ &= w - 10hw + 10h \\ g'(w) &= 1 - 10h \\ g'(1) &= 1 - 10h \\ 11 - 10h\ | < 1 &\Rightarrow -1 < 1 - 10h < 1 \\ -2 < -10h &< 0 \\ 2 > 10h &> 0 \\ \frac{1}{5} &> h \end{aligned}$$

iv-) Solve this differential equation using Euler's method on $t\ \in [0, 2]$ using first an h that is too large, and then an appropriately small h. Include a plot of both "solutions" here.



h=1/5



h=1/8

v-) As we discussed earlier, this problem is numerically unstable, when h is "large". We have here an example of a "stiff" differential equation. There is not a precise mathematical definition of stiff, but here are some statements that "characterize" stiffness:

- The stability requirement of the problem constrains h, rather than the accuracy (truncation error requirement) constraining h. (For example, in our differential equation we have not set any constraint on the size of the truncation error, but rather you have determined that you need h to be sufficiently small just to obtain a "stable" solution.)

- A problem is stiff if its solution is composed of some terms that are decaying very rapidly compared to others. (For example, if the solution had an e^{-t} and an e^{-10t}term in it.)

- From the above statement, we can see that stiffness also suggests that there are regions of t where h must be excessively small relative to the smoothness of the exact solution. (In the example where the solution has an $e^{-t}andane^{-10t}$term in it, we could use a large h for large t, but would need a very small h for very small t, in order accurately capture the solution.) Neither Euler nor RK, including ode45, are suitable for stiff equations. Instead we introduce the Backward Euler Method, which uses the slope of the solution at the right-hand endpoint (instead of the left as in regular Euler) in Euler's formula. This gives $y_{i+1} = y_i + h\ f\ (t_{i+1},\ y_{i+1})$. The only problem is that we don'ty_{i+1} so how can we possibly plug it in on the RHS? This is an example of an implicit method, rather than the explicit methods we have considered previously.

Consider our example. In this case Backward Euler gives $y_{i+1} = y_i + h\ 10(1 - y_{i+1})$. Solve this for y_{i+1}.

$$y_i + 10h - 10hy_{i+1} = y_{i+1}$$
$$y_{i+1} + 10hy_{i+1} = y_i + 10h$$
$$y_{i+1} = \frac{y_i + 10h}{1 + 10h}$$

vi-) Again, by letting $y_{i+1} \approx y_i = w$, this can be written as a fixed point problem. What is g in this case?

$$w = \frac{w + 10h}{1 + 10h}$$

vii-) Show that this fixed point iteration converges for all h ¿ 0.
ow that this fixed paint iteration converges for all $h > 0$.

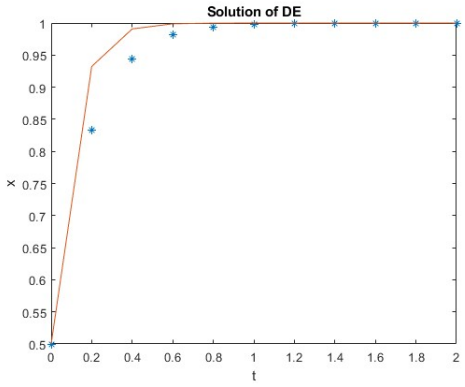
$$g(w) = \frac{w + 10h}{1 + 10h}$$
$$g'(w) = \frac{(1)(1 + 10h) - (w + 10h)(0)}{(1 + 10h)^2}$$
$$g'(w) = \frac{1 + 10h}{(1 + 10h)^2} = \frac{1}{1 + 10h}$$

which is always less than 1, given h is always > 0

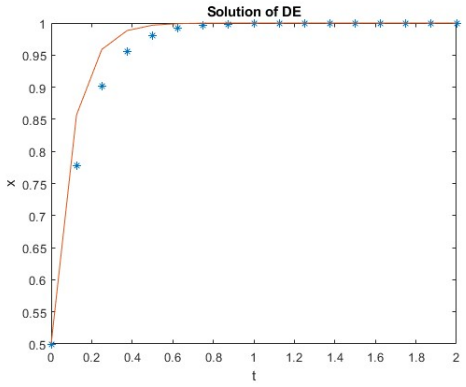
viii-) Write matlab code to solve our example problem using Backward Euler. Run it using the same two values of h that you used previously for regular Euler. Include your code and the graphs of the solution here.

```
%% Backwards Euler DE Solver
clear all;
close all;
clc;
% input
f = @(t,y) 10-10*y;
deltat = 1/5;
tfinal = 2;
N = round(tfinal/deltat);
t(1) = 0; x(1) = 1/2;

% Backward Euler steps
for i=1:N
    t(i+1) = t(i)+deltat;
    x(i+1) = (x(i)+10*deltat)/(1+10*deltat);
end
```

```
% Output

plot(t,x,*)
xlabel(t)
ylabel(x)
title(Solution of DE)

xexact = 1-1/2*exp(-10*t);
hold on
plot(t,xexact)

% More output
t=t';
x=x';
xexact=xexact';
table(t,x,xexact)

% Error
error = abs(x-xexact);
```



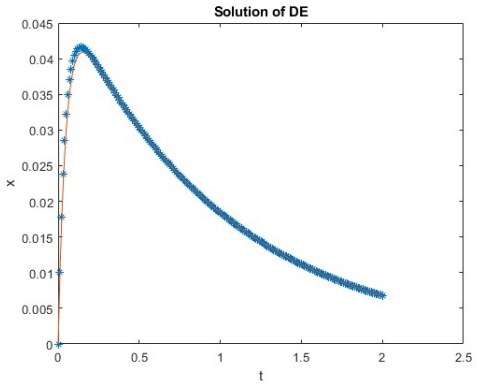
h=1/5



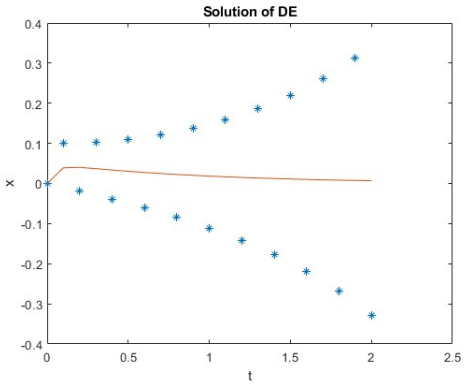
h=1/8

ix-) Of course, generally it will not be so easy to rewrite the problem so that we can use Backward Euler. So some other strategy will be necessary to solve the problem, for example fixed point iteration or Newton's method. Backward Euler is a first order method, just like Euler. We can extend Backward Euler to a 2nd order method (just as we did for Euler to get Heun's method for example.) We could then develop the corresponding 2nd order implicit method, which would be suitable for stiff equations. In matlab, ode23s can be used for solving stiff equations (note the s). It's like ode45 in that it compares RK2 and RK3 (rather than RK4 and RK5) to determine the appropriate h. Another example: $\frac{dy}{dt} = -21y + e^{-t},\ y(0) = 0.$

Solve this problem using Euler's method on $t \in [0, 2]$ with h = 0.01 and again with h = 0.1. You should see that the problem is stiff. Paste the graphs of the solutions here.



h=0.01



h=0.1

x-) Solve the problem analytically using an integrating factor to obtain the exact solution.

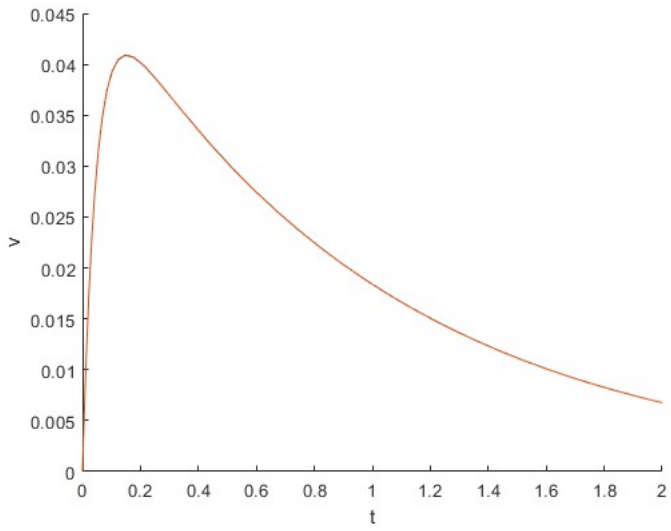
$$\frac{dy}{dt} = -21y + e^{-t}, y(0) = 0$$
$$\frac{dy}{dt} + 21y = e^{-t}$$
$$e^{21t}\frac{dy}{dt} + 21ye^{21t} = e^{21t}e^{-t}$$
$$\frac{d}{dt}e^{21t}y = e^{20t}$$
$$e^{21t}y = \frac{e^{20t}}{20} + c \Rightarrow y = \frac{1}{20}e^{-t} + ce^{-21t}$$
$$0 = \frac{1}{20} + c \Rightarrow c = -1/20$$
$$y = \frac{1}{20}e^{-t} - \frac{1}{20}e^{-21t}$$

xi-) Discuss what you observe about the analytic solution and the numerical solution that suggest the problem is stiff.

In the analytic solution, we have both e^{-t} and e^{-21t}, which definitely characterize the problem as stiff. Looking at the numerical solution, it is possible to see that when h is small enough it converges to the solution, but that is not the case for other h values, without a smooth change, due to the imbalance between terms.

xii-) Finally, modify your matlab code that uses ode45 to instead use ode23s for this problem. The syntax for ode23s is exactly the same as that for ode45. Include your code and a graph of the solution here.

```
clear all;
close all;
%%
f = @(t,y) -21*y+exp(-t);

tspan = [0 2]; %time interval
v0 = 0; % Initial velocity

%%RK
[t,v] = ode23s(f,tspan,v0);

%%
xexact = 1/20*exp(-t)-1/20*exp(-21*t);
hold on
plot(t,xexact)
plot(t,v)
xlabel('t')
ylabel('v')
```



Both xexact and v line up perfectly

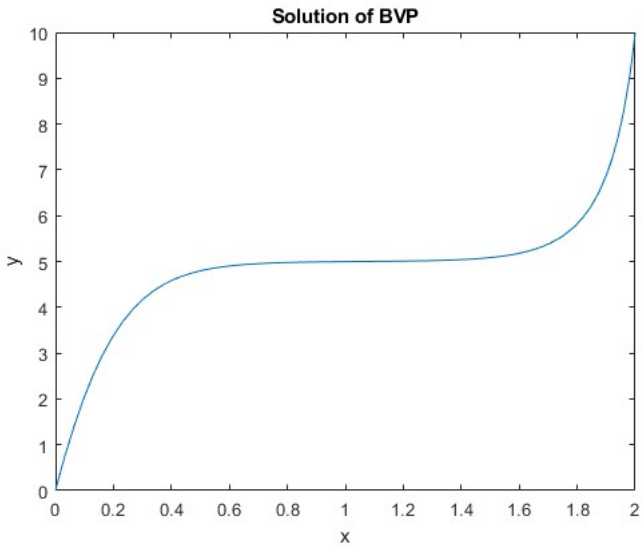
6. Given points (a,s) and (b,t) falling on a straight line, y = mx + b, we have the equation $x = \frac{b-a}{t-s}(y-s)+a$. Write matlab code to solve the heat equation given in class using the shooting method. Use L = 2, w =4, u(0) = 0, u(L) = 10 and Tair = 5. Graph the solution and verify that it looks like the solution obtained in class. You will need to write the equation as a system and implement ode45 three different times. Submit your code and graph.

```
%%Shooting Method for Nonlinear BVP ODEs
clear all; close all
global y0 yL L Tair w
y0 = 0; yL = 10; L = 2; Tair = 5; w = 4;
%%
a = 1;
s = F(a);

b = 20;
line_t = F(b);


ydotopt = ((b-a)/(line_t-s))*(y0-s)+a;
%%
%Solve the IVP using ydotopt
ydot = @(t,y) [y(2);-w*(Tair - y(1))];      %Set up the
   system of ODEs
tspan = [0 L]; ICs = [y0 ydotopt];          %Set the BCs
[t,y] = ode45(ydot, tspan, ICs);            %Solve the IVP
plot(t,y(:,1))
xlabel('x'); ylabel('y');title('Solution of BVP')

%% Residual function. fzero finds root of fn R
function R = F(s)
global y0 yL L Tair w%%
%Define the system of ODEs
ydot = @(t,y) [y(2);-w*(Tair - y(1))];
tspan = [0 L]; ICs = [y0 s];            %Set the BCs
[t,y] = ode45(ydot, tspan, ICs);        %Solve the system
R = y(end,1) - yL;
% Compute and return the difference between
% the predicted and actual BC
end
```

7. A more realistic ODE describing heat transfer in the rod, that also accounts for radiative transfer, is $u'' + w\left(Tair - u\right) + \sigma\left(Tair^4 - u^4\right) = 0$. Note that this is a nonlinear ODE. Modify the matlab code provided to solve this BVP with the same parameters and BCs. Use σ= 0.01. Plot the resulting solution.

```
%%Shooting Method for Nonlinear BVP ODEs
clear all; close all
global y0 yL L Tair w sigma
```

```
y0 = 0; yL = 10; L = 2; Tair = 5; w = 4; sigma = 0.1;

%Before running, determine values of ydot that yield
%one + and one - residual (R= pred y(L) - y(L)).
%These values are the endpoints of the search interval.
%fzero will determine the root on that search interval
% u'_opt
ydotopt = fzero(@F, [1 25]) ;
%Value of y' that will hit the BC

%Solve the IVP using ydotopt
% u'
% Set up the system of ODEs
ydot = @(t,y) [y(2);
-w*(Tair - y(1))-sigma*(Tair.^4-y(1).^4)];
 %Set the BCs
tspan = [0 L];
ICs = [y0 ydotopt];
[t,y] = ode45(ydot, tspan, ICs);            %Solve the IVP
plot(t,y(:,1))
xlabel('x');
ylabel('y');title('Solution of BVP')

%% Residual function. fzero finds root of fn R
function R = F(s)
global y0 yL L Tair w sigma
% THIS GIVES RESIDUAL WITH DIFFERENT S VALUES
    ydot = @(t,y) [y(2);
    -w*(Tair - y(1))-sigma*(Tair.^4-y(1).^4)];
    tspan = [0 L]; ICs = [y0 s];
    [t,y] = ode45(ydot, tspan, ICs);
    R = y(end,1) - yL;
end
```



8. Consider constructing the analytic solution to the BVP

$$u'' = -wu,\ u(0) = 0,\ u(\pi/2) = 10$$

. We'll use the guessing method as done in class.

i-) Start with w = 1. What function(s) u satisfies the DE? How many solutions must there be?

$$w = 1 \Rightarrow u'' = -u \Rightarrow u = sin(x), cos(x)$$

So general solution is

$$u = c_1 sin(x) + c_2 cos(x)$$

ii-) Now solve the ODE with w = 4. Recall that the general solution is the linear combination of two linearly independent solutions.

$$w = 1 \Rightarrow u'' = -4u \Rightarrow u = sin(2x), cos(2x)$$

So general solution is

$$u = c_1 sin(2x) + c_2 cos(2x)$$

iii-) Impose the boundary conditions to determine the solutions to the BVP.
for part i:

$$u(0) = 0 \Rightarrow c_2 = 0$$

$$u(\pi/4) = \sqrt{2}/2 c_1 = 10 \Rightarrow c_1 = 20/\sqrt{2}$$

Therefore, the solution is

$$u = 20/\sqrt{2} sin(x)$$

for part ii:

$$u(0) = 0 \Rightarrow c_2 = 0$$

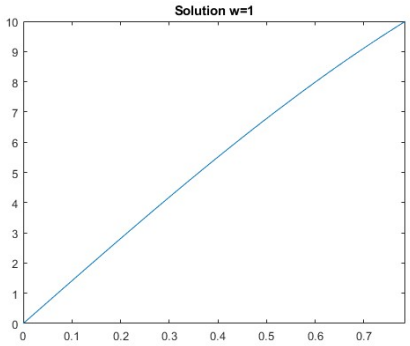
$$u(\pi/4) = c_1 = 10$$

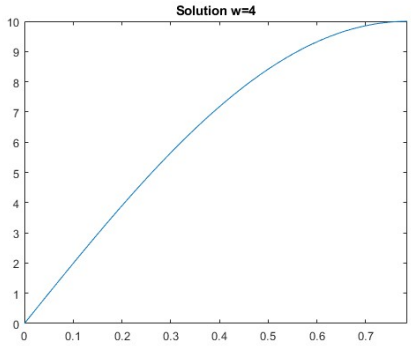
Therefore, the solution is

$$u = 10sin(2x)$$

iv-) Use matlab to plot the solutions.

```
figure
fplot(@(x) 10*sin(2*x), [0 pi/4])
figure
fplot(@(x) 20/sqrt(2)*sin(x),[0,pi/4])
```

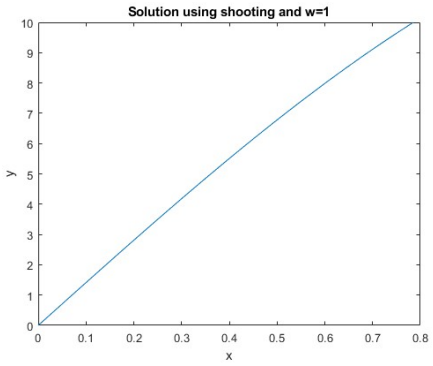


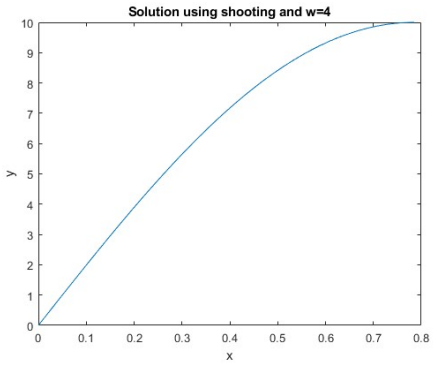


v-) Now solve the problem numerically using the shooting method. Plot the solutions and compare to iv).

```
%% Shooting Method for Nonlinear BVP ODEs
clear all; close all
global y0 yL L Tair w
y0 = 0; yL = 10; L = pi/4; Tair = 0; w = 1;

ydotopt = fzero(@F, [10 21]);
ydot = @(t, y) [y(2); -w *y(1)];
tspan = [0 L]; ICs = [y0 ydotopt];
[t, y] = ode45(ydot, tspan, ICs);
plot(t, y(:, 1))
xlabel('x'); ylabel('y');
title('Solution using shooting and w=1')

%% Residual function. fzero finds the root of fn R
function R = F(s)
global y0 yL L Tair w
```

```
ydot = @(t, y) [y(2); -w *y(1)];
tspan = [0 L]; ICs = [y0 s];
[t, y] = ode45(ydot, tspan, ICs);
R = y(end, 1) - yL;
end
```





Solutions look the same graphic wise, analysing the differences between those two numerically shows a little bit of a difference, but basically the same.

9. Students taking the course for graduate credit only:

i-) Consider the original heat equation (σ=0) with fixed temperature at x = L (Dirichlet condition) but with a "free" end at x = 0. This means that since we are considering steady-state (long time equilibrium), convection must equal conduction at the x = 0 end of the rod. Set up the heat balance there to determine the boundary condition at x = 0.

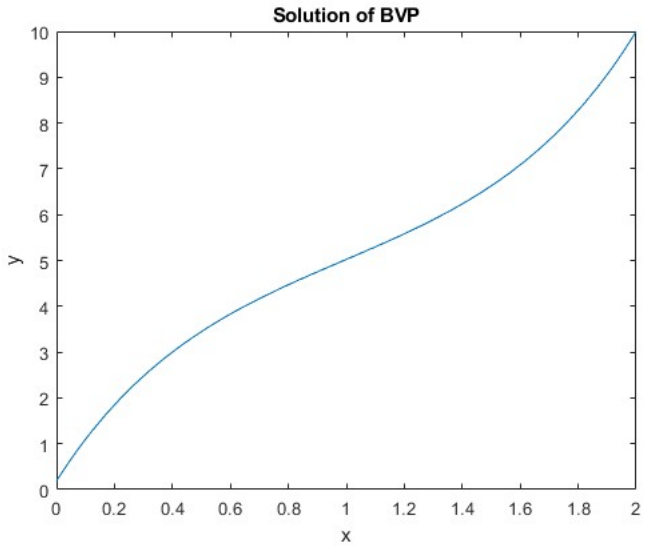
$$u'' + w(t_{air} - u) = 0, u''(L) = 0$$

This steady state means a linear function of the form $u'(0) = w(t_{air} - u(0))$ at the end.

ii-) Solve this new BVP using the shooting method. Plot the solution.

```
%%Shooting Method for Nonlinear BVP ODEs
clear all; close all
global y0 yL L Tair w sigma
y0 = 0; yL = 10; L = 2; Tair = 5; w = 4; sigma = 0;

ydotopt = fzero(@F, [-1 1]) ;

%Solve the IVP using ydotopt
% u'
ydot = @(t,y) [y(2);
```

```
    -w*(Tair - y(1))-sigma*(Tair.^4-y(1).^4)];
tspan = [0 L]; ICs = [ydotopt yL];
[t,y] = ode45(ydot, tspan, ICs);
plot(t,y(:,1))
xlabel('x'); ylabel('y');title('Solution of BVP')
%% Residual function. fzero finds root of fn R
function R = F(s)
global y0 yL L Tair w sigma%%
%y0 = 0; yL = 10; L = 2; Tair = 5; w = 4;
% THIS GIVES RESIDUAL WITH DIFFERENT S VALUES
ydot = @(t,y) [y(2);
    -w*(Tair - y(1))-sigma*(Tair.^4-y(1).^4)];
tspan = [0 L]; ICs = [s yL];
[t,y] = ode45(ydot, tspan, ICs);
R = y(end,1) - yL;
end
```



iii-) Interpret the solution. Does it make sense?

It makes sense, it is not starting exactly at 0, like it was previously. And it is ending exactly at 10.