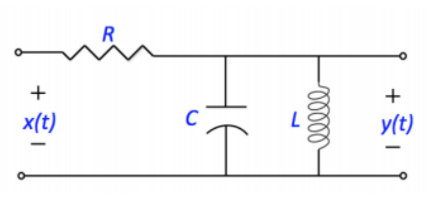
Frank Isidore Gomez
4/23/21
ID: 1650550
figomez
ECE 103L

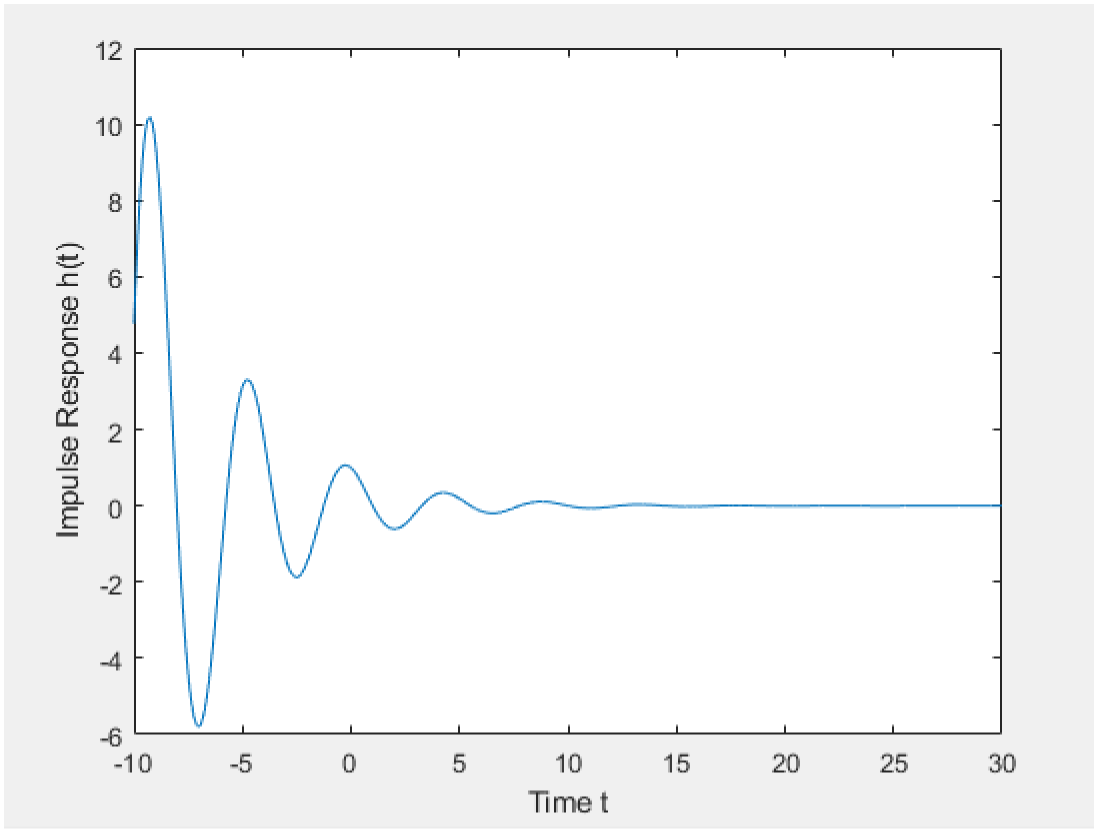
Lab 3 - Convolutions in Matlab

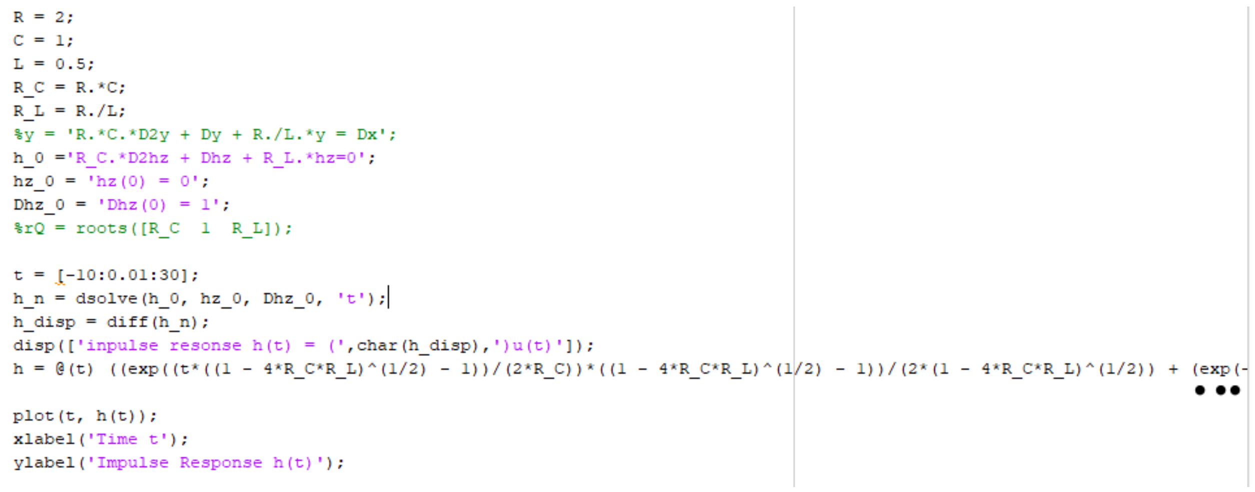
1. Following *RLC* circuit is described by the differential equation (1). Use Matlab built-in differential equation solver `dsolve()` to derive the impulse response *h(t)* for this circuit when $R=2\ \Omega$, $C=1F$, $L=0.5\ H$. Plot the impulse response $h(t)$ from a range $-10 \le t \le 30$.



$$RC\frac{d^2y(t)}{dt^2} + \frac{dy(t)}{dt} + \frac{R}{L}y(t) = \frac{dx(t)}{dt} \quad \ldots(1)$$

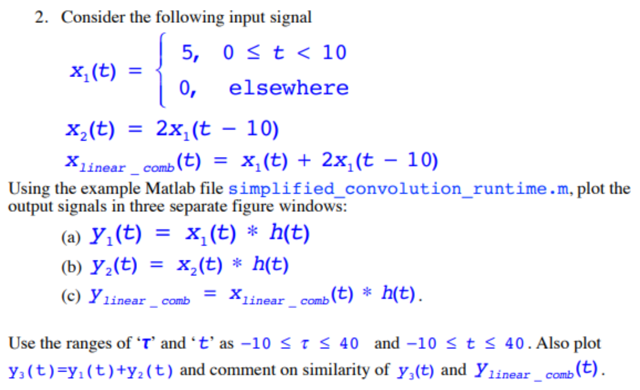
1.





```
R = 2;
C = 1;
L = 0.5;
R_C = R.*C;
R_L = R./L;
%y = 'R.*C.*D2y + Dy + R./L.*y = Dx';
h_0 ='R_C.*D2hz + Dhz + R_L.*hz=0';
hz_0 = 'hz(0) = 0';
Dhz_0 = 'Dhz(0) = 1';
%rQ = roots([R_C  1  R_L]);

t = [-10:0.01:30];
h_n = dsolve(h_0, hz_0, Dhz_0, 't');
h_disp = diff(h_n);
disp(['inpulse resonse h(t) = (',char(h_disp),')u(t)']);
h = @(t) ((exp((t*((1 - 4*R_C*R_L)^(1/2) - 1))/(2*R_C))*((1 - 4*R_C*R_L)^(1/2) - 1))/(2*(1 - 4*R_C*R_L)^(1/2)) + (exp(-
                                                                                                                    •••
plot(t, h(t));
xlabel('Time t');
ylabel('Impulse Response h(t)');
```

This time around, the first segment actually took time to do, because I couldn’t figure out the second initial condition, only to then read that its N-1’s initial condition is always 1 if x(t) isn’t anything special. Otherwise, it’s just the visualization of a circuit’s impulse response.



2. Consider the following input signal

$$x_1(t) = \begin{cases} 5, & 0 \le t < 10 \\ 0, & \text{elsewhere} \end{cases}$$

$$x_2(t) = 2x_1(t-10)$$

$$x_{linear\_comb}(t) = x_1(t) + 2x_1(t-10)$$

Using the example Matlab file `simplified_convolution_runtime.m`, plot the output signals in three separate figure windows:

(a) $y_1(t) = x_1(t) * h(t)$

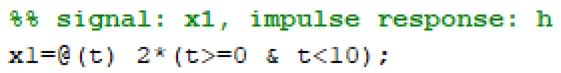
(b) $y_2(t) = x_2(t) * h(t)$

(c) $y_{linear\_comb} = x_{linear\_comb}(t) * h(t)$.

Use the ranges of 'τ' and 't' as $-10 \le \tau \le 40$ and $-10 \le t \le 40$. Also plot $y_3(t)=y_1(t)+y_2(t)$ and comment on similarity of $y_3(t)$ and $y_{linear\_comb}(t)$.

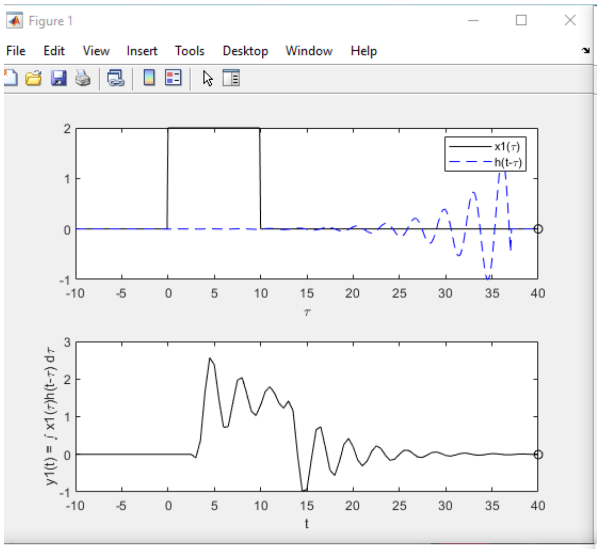
2.

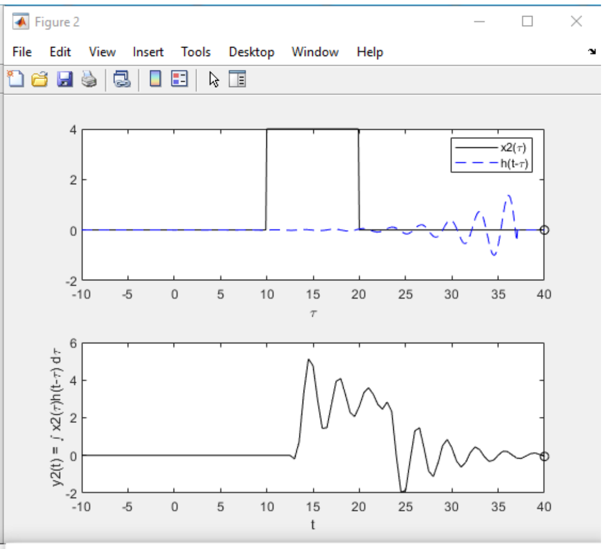
I'm not going to post the entire code for this section, otherwise it'd last for a few pages given how long it is, so I'll just post the new thing I learned.

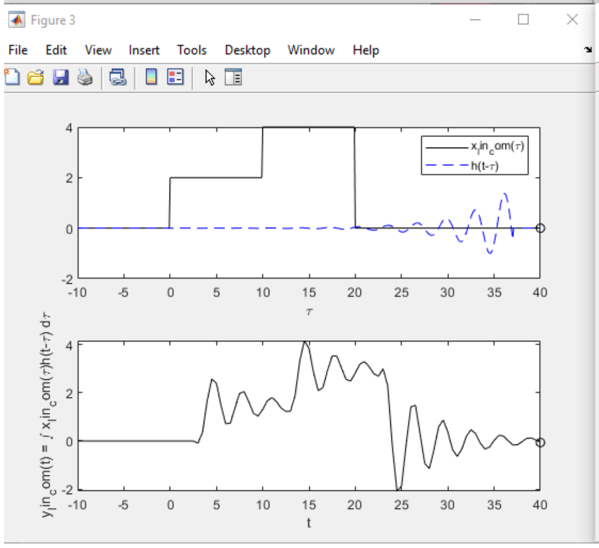


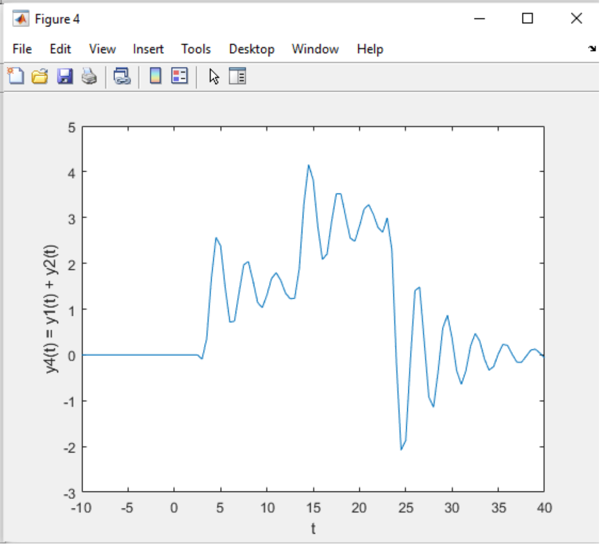
```
%% signal: x1, impulse response: h
x1=@(t) 2*(t>=0 & t<10);
```

Didn't know you could declare piecewise functions like that. It's good to know.





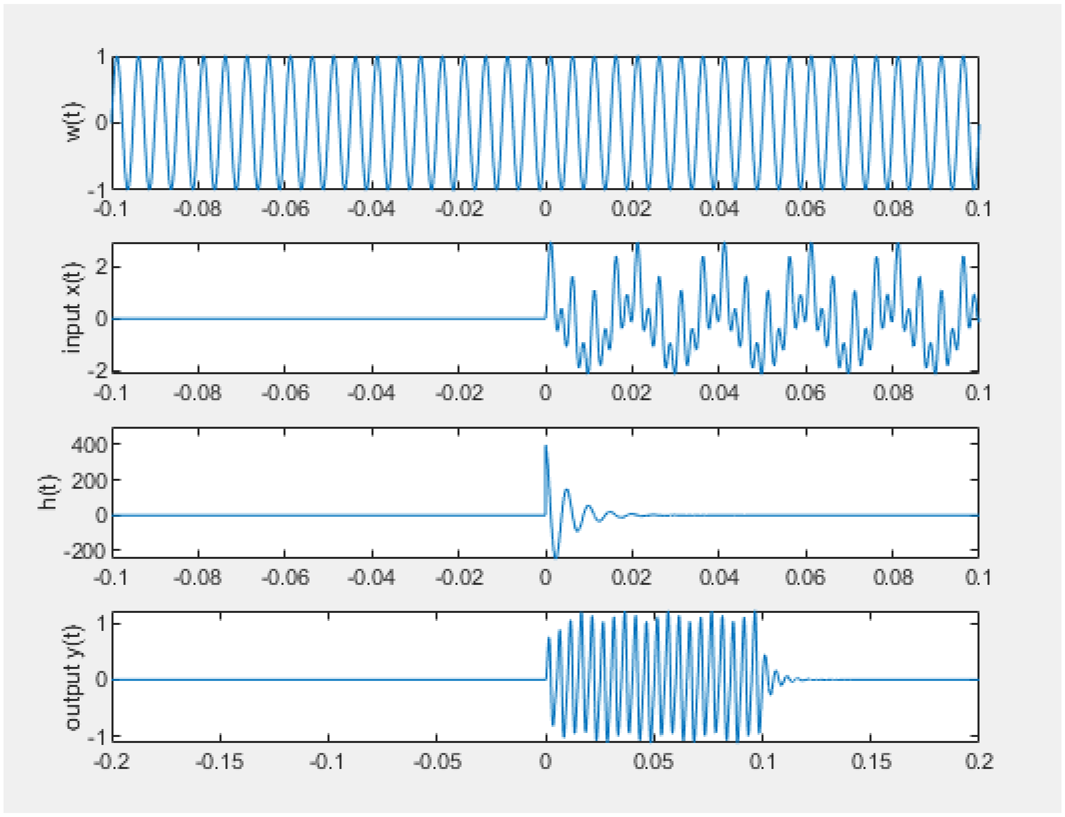




Anyways, for this part, we plotted three convolutions onto graphs, a piecewise function, a scaled, shifted version of that same function, and the linear of the two functions together. As can be seen, the fourth function is identical to the third function, as it's just the normal addition of the first and second functions together. Nothing much else to be said about them.

3. A single-tone signal $w(t) = \sin(400\pi t)$ is transmitted to an audio amplifier and speaker to produce a high-temperature warning for a silicon crystal-growing factory. A filter having impulse response $h(t) = 400e^{-200t}\cos(400\pi t)u(t)$ has been designed to reduce additive interference in the received signal. Using Matlab in-built convolution function: `conv()`, find the filter output signal $y(t)$, when the received signal is $x(t) = [\cos(100\pi t) + \sin(400\pi t) - \cos(800\pi t)]u(t)$ (signal $w(t)$ was corrupted by interference and resulted in an input signal $x(t)$). Plot the output signal, the input signal, and $w(t)$ for the range of $-0.1 \leq t \leq 0.1$. Comment on the effect of the filter on the signal. While solving this problem, pay attention to the time resolution/step (`dT`) you need to use.

3.



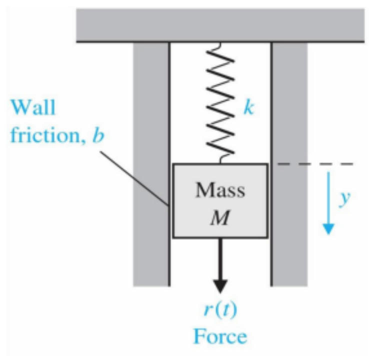
```
t = [-0.1:0.001:0.1];

dT = 1/20/400./pi;
tx = -0.1:dT:0.1;
th = -0.1:dT:0.1;
h = 400.*exp(-200.*th).*cos(400.*pi.*th).*(th >=0);
x = (cos(100.*pi.*tx) + sin(400.*pi.*tx) - cos(800.*pi.*tx)).*(tx >=0);
ty=min(tx)+min(th):dT:max(tx)+max(th);
y = conv(x, h)*dT;
figure(1)
subplot(414);
plot(ty, y);
ylabel('output y(t)');


subplot(412);
plot(tx,x);
ylabel('input x(t)');


t = [-0.1:0.0005:0.1];
w = sin(400.*pi.*tx);
subplot(411);
plot(tx,w);
ylabel('w(t)');


subplot(413);
plot(th,h);
ylabel('h(t)');
```

The story this time around, is there's a signal, w(t), that's supposed to be received, but it receives interference, and becomes the corrupted signal, x(t). As can be seen, the original signal, w(t), is relatively stable, it always returns to the same high points. The corrupted signal, x(t), on the other hand, is much more distorted and chaotic, oscillating before completing a period. Thankfully, the filter, h(t), handles the corrupted signal, blocking it gradually after time t = 0.1s. As for the dT I needed to use, to get it to scale correctly, I used 1/20/400/pi. This was needed to calculate the intervals for x, h, and w as well, and finally, the combination of x and h's created the final interval, ty, for the output y.

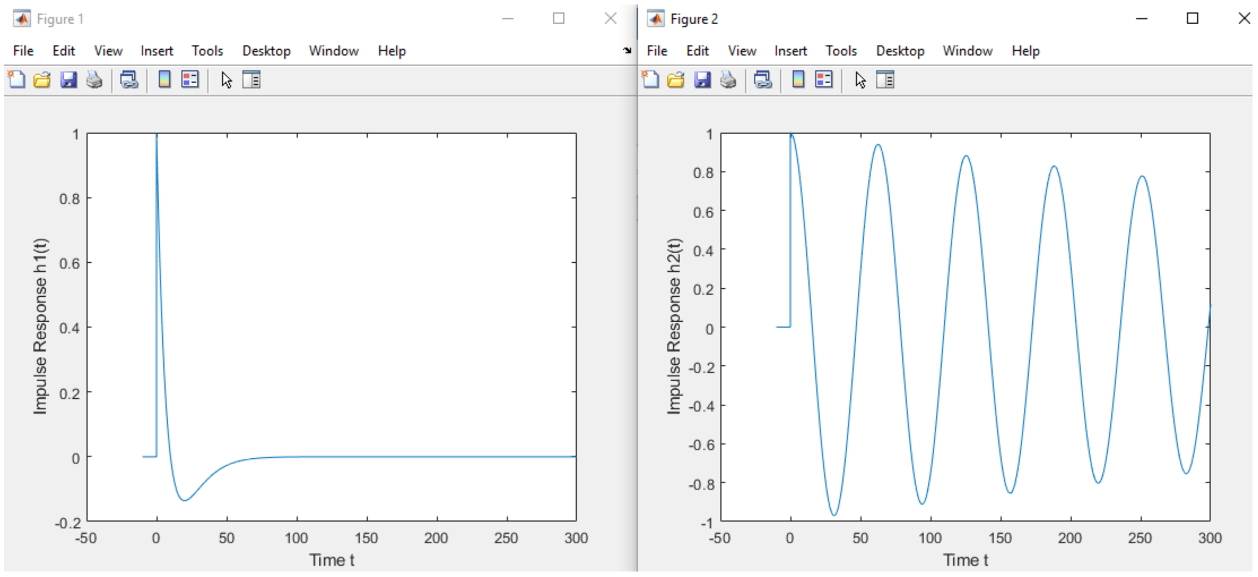
4. System response for an Industrial Shock Absorber (figure below) can be modeled with the following differential equation:

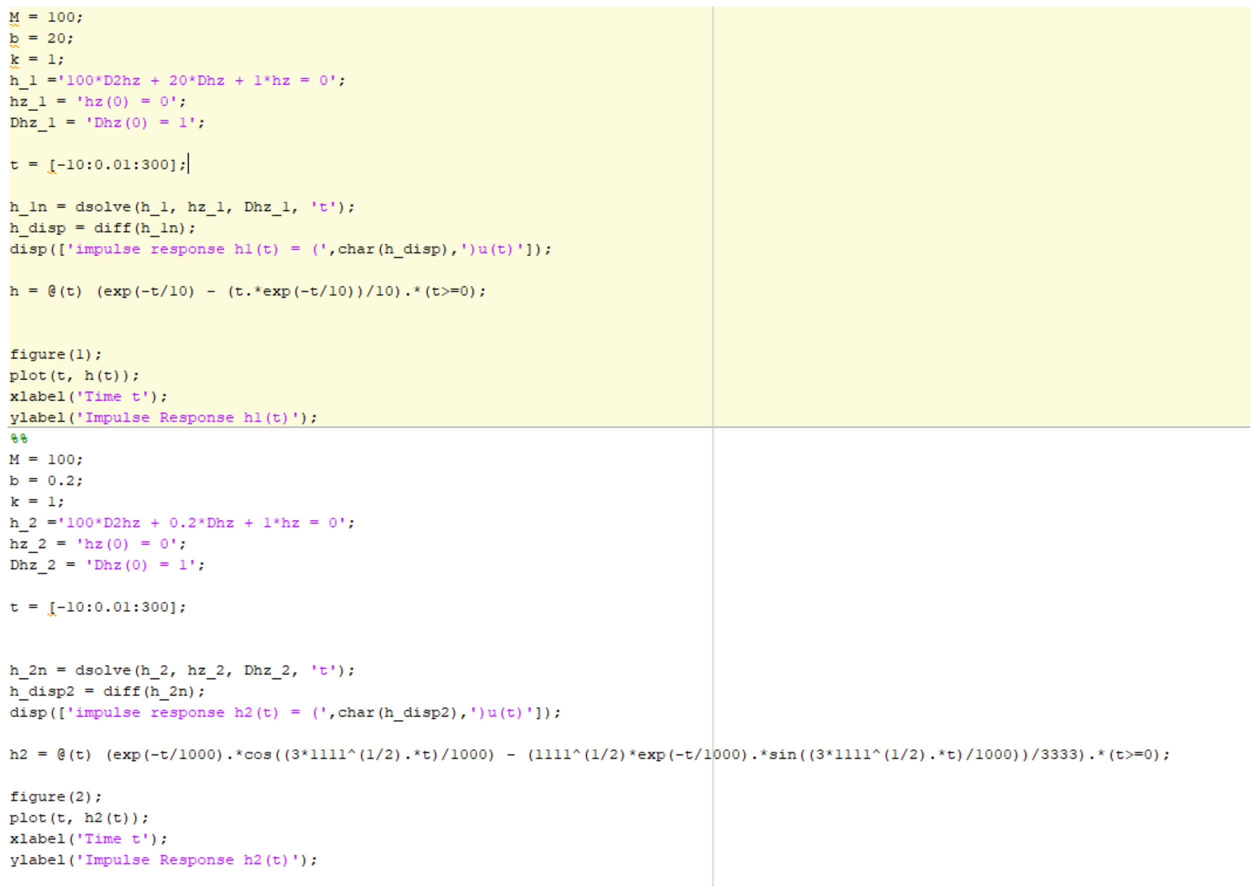
$$M\frac{d^2y(t)}{dt^2} + b\frac{dy(t)}{dt} + ky(t) = r(t) \qquad \ldots(2)$$

Let's assume the mass of the damper M is `100 kg`, the spring constant k is `1 kgs`$^{-2}$, and the friction coefficient b is `20 kgs`$^{-1}$. Using Matlab built-in differential equation solver `dsolve()` to derive the impulse response $h_1(t)$ for this Industrial Shock Absorber and the impulse response $h_1(t)$ from a range $-10s \le t \le 300s$. Overtime the oil inside the shock absorber degrades and the friction coefficient b becomes `0.2 kgs`$^{-1}$. Derive the new impulse response $h_2(t)$ for this Industrial Shock Absorber and plot $h_2(t)$ from a range $-10s \le t \le 300s$.



4.





```
M = 100;
b = 20;
k = 1;
h_1 ='100*D2hz + 20*Dhz + 1*hz = 0';
hz_1 = 'hz(0) = 0';
Dhz_1 = 'Dhz(0) = 1';

t = [-10:0.01:300];

h_1n = dsolve(h_1, hz_1, Dhz_1, 't');
h_disp = diff(h_1n);
disp(['impulse response h1(t) = (',char(h_disp),')u(t)']);

h = @(t) (exp(-t/10) - (t.*exp(-t/10))/10).*(t>=0);


figure(1);
plot(t, h(t));
xlabel('Time t');
ylabel('Impulse Response h1(t)');
%%
M = 100;
b = 0.2;
k = 1;
h_2 ='100*D2hz + 0.2*Dhz + 1*hz = 0';
hz_2 = 'hz(0) = 0';
Dhz_2 = 'Dhz(0) = 1';

t = [-10:0.01:300];


h_2n = dsolve(h_2, hz_2, Dhz_2, 't');
h_disp2 = diff(h_2n);
disp(['impulse response h2(t) = (',char(h_disp2),')u(t)']);

h2 = @(t) (exp(-t/1000).*cos((3*1111^(1/2).*t)/1000) - (1111^(1/2)*exp(-t/1000).*sin((3*1111^(1/2).*t)/1000))/3333).*(t>=0);

figure(2);
plot(t, h2(t));
xlabel('Time t');
ylabel('Impulse Response h2(t)');
```

It's a weird problem that was only fixed by removing the variables, and using the numbers 100, 20, and 0.2 instead. Anyways, it's a block, held by a spring, that oscillates after being released at time t = 0. B stands for the friction coefficient, which is 20 in the first one, pretty high. Since it's so high, the block barely oscillates before stopping. In contrast, for the second part, the friction is so low, that the block oscillates beyond the time it's being observed.