EE2-08C Numerical Analysis

Group 9

Contents

1 Exercise 1 - RL Circuit

1.1 Background

This simple RL circuit (1) forms a high pass filter circuit which takes an input signal Vin and only allows the high frequency components to pass. Despite the inductor being less convenient than a capacitor, this RL circuit can approach the model of a DC motor. In this specific case, we want to model a DC motor with inertia 250 μsNm/s^2 and $T_{max} = 50mNm/A$. To do so, according to KCL, we know that

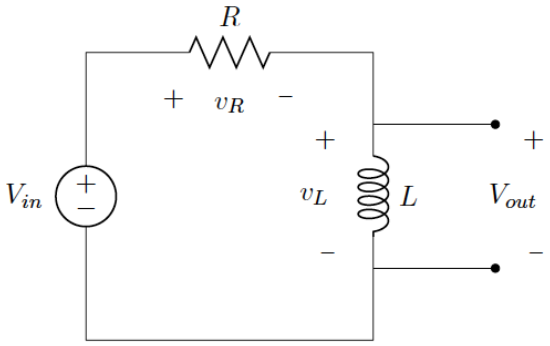


Figure 1: RL Circuit

$$Vin(t) = v_L(t) + v_R(t) \tag{1}$$

and therefore:

$$V_{in}(t) = L\frac{d}{dt}i_L(t) + Ri_L(t) \tag{2}$$

$i_L(t)$ is the state that we will solve following the three different second-order Runge Kutta methods.
All methods follow the same pattern:

$$x_{i+1} = x_i + h \tag{3}$$

h being the step size and

$$y_{i+1} = y_i + h(ak_1 + bk_2) \tag{4}$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + ph, y_i + qk_1h)$$

and the values of a, b, p and q depend on which method we are using:

Parameter	Heuns	Midpoint	Ralston
a	1/2	0	1/4
b	1/2	1	3/4
p	1	1/2	2/3
q	1	1/2	2/3

Each method will approximate the exact solution according to these parameters. Therefore, the smaller the step size h, the more exact the method will approximate the solution.
After doing so, the output value will be obtained as follows:

$$V_{out} = V_{in}(t) - Ri_L(t) \tag{5}$$

1.2 Second-Order Runge Kutta Scripts

The code for Heuns.m to solve $i_L(t)$ will be as follows:

```
function [t,y] = heun(func,t0,y0,tf,h) % y = i_L(t), func = equation (2),
% t0 = initial time, tf = final time, h = step size
N = round((tf-t0)/h);   % N = nr. of steps
arrt = zeros(1,N);      % arrt = array of time
arry = zeros(1,N);      % arry = array of y
arrt(1) = t0;           % initial conditions
arry(1) = y0;
for i = 1 : N-1         % evaluate arry and arrt for each step
    k1 = feval(func, arrt(i), arry(i));
    k2 = feval(func, arrt(i) + h, arry(i) + k1*h);
    arry(i+1) = arry(i) + h*0.5*(k1 + k2);
    arrt(i+1) = arrt(i) + h;
end
t= arrt;                % return values
y = arry;
end
```

Similarly, the code for Midpoint.m will be:

```
function [t,y] = midpoint(func, t0, y0, tf, h)     % y = i_L(t), func = equation (2),
% t0 = initial time, tf = final time, h = step size
N = round((tf - t0)/h); % N = nr. of steps
arrt = zeros(1,N);      % arrt = array of time
arry = zeros(1,N);      % arry = array of y
arrt(1) = t0;           % initial conditions
arry(1) = y0;
for i = 1:N-1           % evaluate arry and arrt for each step
    k1 = feval(func,arrt(i),arry(i));
    k2 = feval(func, arrt(i) + 0.5*h, arry(i) + 0.5*h*k1);
    arry(i+1) = arry(i) + h*((1/4)*k1 + (3/4)*k2);
    arrt(i+1) = arrt(i) + h;
end
t = arrt; y = arry;     % return values
end
```

Lastly, Ralston.m will be as similar:

```
function [t, y] = ralston(func,t0,y0,tf,h)  % y = i_L(t), func = equation (2),
% t0 = initial time, tf = final time, h = step size
N = round((tf-t0)/h);   % N = nr. of steps
arrt = zeros(1,N);      % arrt = array of time
arry = zeros(1,N);      % arry = array of y
arrt(1) = t0;           % initial conditions
arry(1) = y0;
for i = 1 : N-1         % evaluate arry and arrt for each step
    k1 = feval(func, arrt(i), arry(i));
    k2 = feval(func, arrt(i)+(2/3)*h, arry(i)+(2/3)*h*k1);
    arry(i+1) = arry(i) + (h/4)*(k1 + 3*k2);
    arrt(i+1) = arrt(i) + h;
end
y = arry; t = arrt;     % return values
end
```

As mentioned above, we can appreciate that the three second-order Runge Kutta methods are almost identical with the parameters inside the *for loop* the only difference between them.

1.3 Script to solve the ODE

Given that the values of the components of the RL circuit are:

$$R = 0.5\Omega \quad L = 1.5mH$$

We will find the value of V_{out} according to a script which always follows the same pattern, depending on the values of V_{in} and adjusting the values of h and t_f. A standard script would be as follows:

```
y0 = 0;
t0 = 0; tf = ***;   % to decide
h = ***;            % to decide

Vin =@(t) ***;      % different inputs

R = 0.5; L = 1.5*10^(-3);           % R, L values for this project
func =@(t, y) (Vin(t) - R*y)/L;     % equation 2
[t, y] = ****(func,t0,y0,tf,h);     % heuns, midpoint or ralston
vout = Vin(t)-R*y;                  % equation 5
figure;
plot(t, vout);      % plot V_out against time
title '****'
xlabel 't(s)'
ylabel 'Vout(t)'
```

1.4 Exercise 1 - Graphs

1.4.1 Step signal with amplitude

$\tilde{V}_{in} = 3.5V$
$t0 = 0$, $tf = 0.02$;
$h = 0.001$;
$Vin = @(t)3.5$;

In the step signal case, all methods will lead to equal approximation of the solutions. Figure 2 is applying Heuns method, but if we apply Midpoint (Figure 3b) or Ralston (Figure 3a) with the same step size and time conditions, we get similar plots.

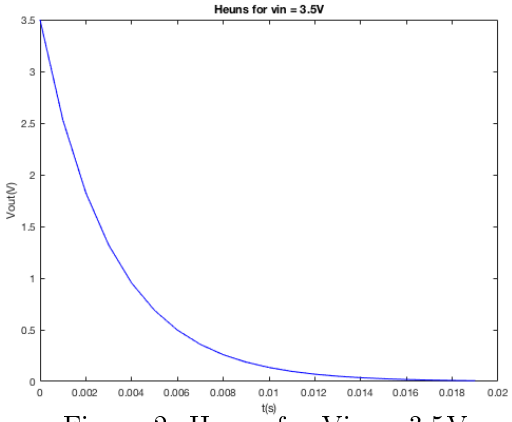
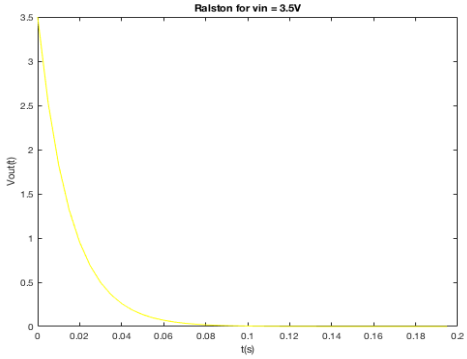
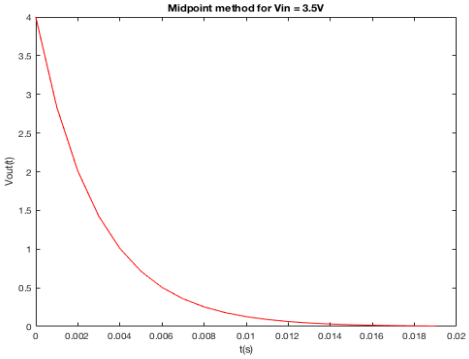


Figure 2: Heuns for Vin = 3.5V



(a) Ralston for Vin = 3.5V



(b) Midpoint for Vin = 3.5V

Figure 3: Ralston and Midpoint for Vin = 3.5V

The MATLAB code below is used to plot teh
result from all three methods on one graph (Figure 4).

```
[t,y] = heun(func,t0,y0,tf,h);
Vout = Vin(t) - R*y;
figure;
plot(t, Vout, 'b'); % plot heuns method in blue
hold on;
[t,y] = ralston(func,t0,y0,tf,h);
Vout = Vin(t) - R*y;
plot(t, Vout, 'r')  % plot midpoint method in red
hold on;
[t,y] = ralston(func,t0,y0,tf,h);
Vout = Vin(t) - R*y;
plot(t, Vout, 'g')
title 'All for step signal Vin = 3.5V'
xlabel 't(s)'
ylabel 'Vout(V)'
legend ('Heuns', 'Midpoint', 'Ralston')
```

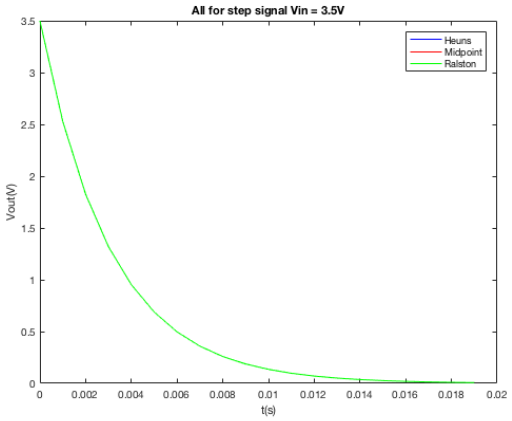


Figure 4: All for Vin = 3.5V

The code above was used to plot all three methods on the same graph. The plot (Figure 4) proves that the three methods are equal for the same step size and time conditions for the given input (Vin = 3.5V).

1.4.2 Impulsive signal and decay

$Vin = Vin * exp(-t^2/\tau)$
In this section, we will examine the 2nd order Runge Kutta Methods using an impulsive signal input given in the equation below:

$$V_{in} = \tilde{V}_{in} exp - t^2/\tau \tag{6}$$

```
t0 = 0, tf = 0.05;
h = 0.0005;
vin = 3.5;
z = 150*10^(-6);
t = t0: T : tf;
Vin = @(t) vin*exp(-(t.^2)/z)
```

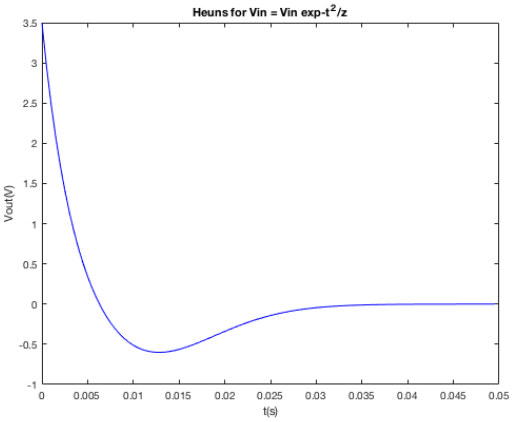
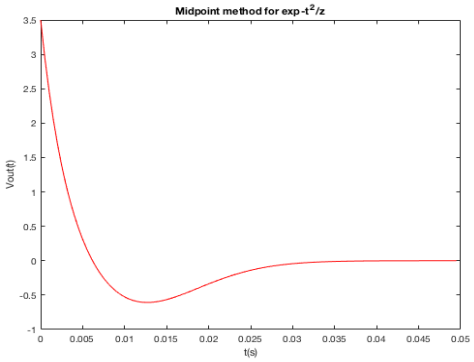
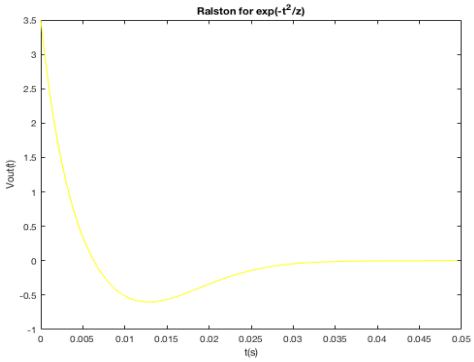


Figure 5: Heuns for $Vin = Vin * exp(-t^2/\tau)$



(a) Midpoint for $Vin = Vin * exp(-t^2/\tau)$



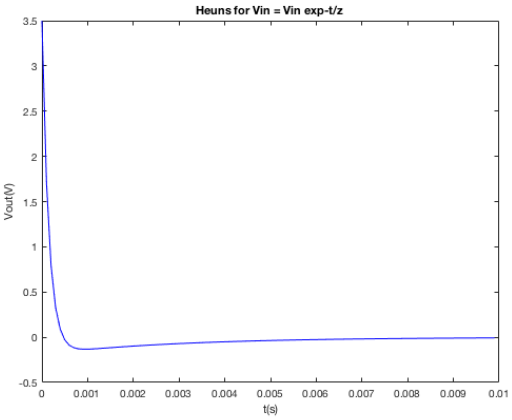
(b) Ralston for $Vin = Vin * exp(-t^2/\tau)$

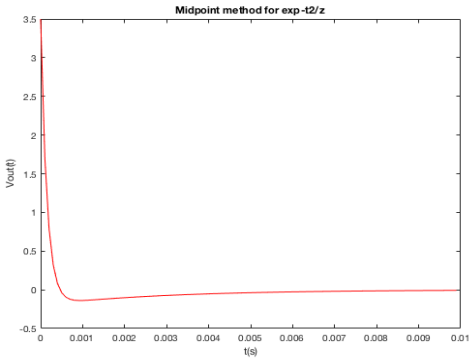
Figure 6: Ralston and Midpoint for $Vin = Vin * exp(-t^2/\tau)$

$Vin = Vin * exp(-t/\tau)$
In order to examine the behaviour of the three 2nd order Runge Kutta methods, we will demonstrate their outputs with another impulsive signal given below:

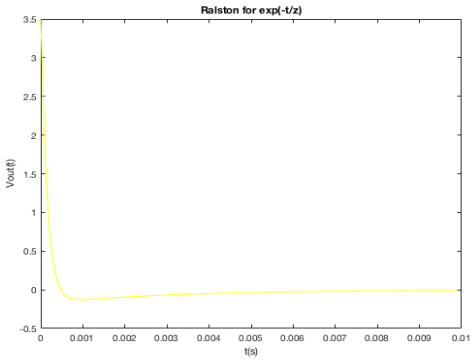
$$V_{in} = \tilde{V}_{in} exp{-t/\tau} \tag{7}$$

```
t0 = 0; tf = 0.1;
h = 0.001;
vin = 3.5;
z = 150*10^(-6);
t = t0: T : tf;
Vin = @(t) vin*exp(-(t)/z);
```





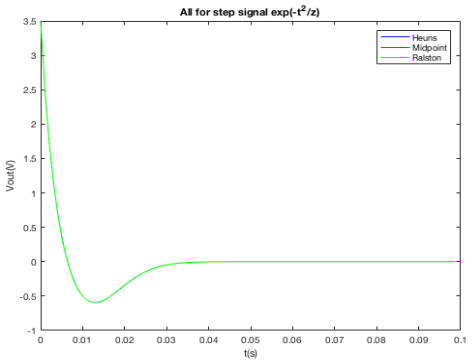
(a) Midpoint for $Vin = Vin * exp(-t/\tau)$



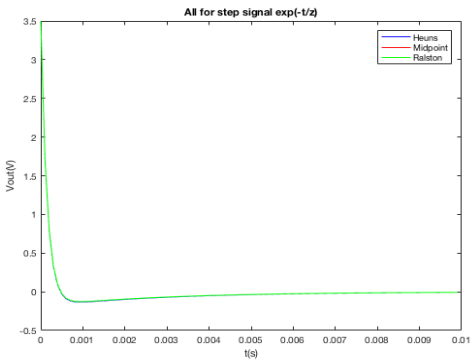
(b) Ralston for $Vin = Vin * exp(-t/\tau)$

Figure 8: Ralston and Midpoint for $Vin = Vin * exp(-t/\tau)$

For both impulsive signals, the difference in the outputs obtained from the three different methods between is minimal. This can be seen from the graphs in Figure:9a and Figure: 9b which plot all three curves obtained for the two impulsive signal inputs $V_{in} = \tilde{V}_{in} exp{-t^2/\tau}$ and $Vin = Vin * exp(-t/\tau)$ respectively.



(a) All for $Vin = Vin * exp(-t^2/\tau)$



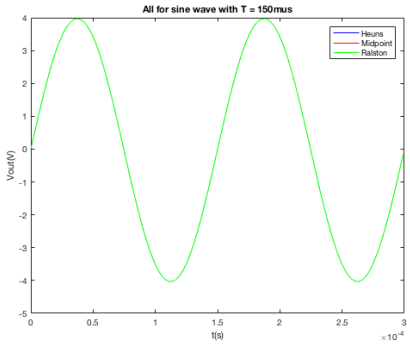
(b) All for $Vin = Vin * exp(-t/\tau)$

Figure 9: All for $Vin = Vin * exp(-t/\tau)$ and $Vin = Vin * exp(-t^2/\tau)$

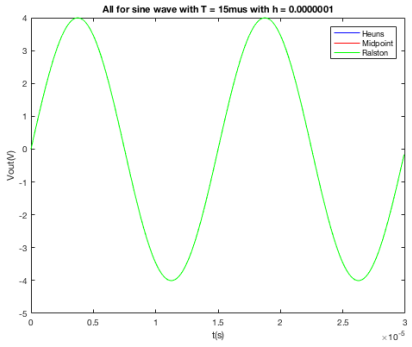
1.4.3 Sine wave amplitude $\tilde{V}_{in} = 4V$

For the waves cases, three methods will be plot in the same graph for comparison purposes and avoid overloading of graphs.

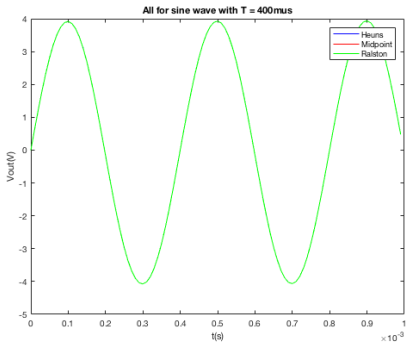
```
vin = 4;
T = ***10^(-6);
t = t0: T : tf;
f = 1/T;
Vin =@(t) vin*sin(2*pi*f*t);
```



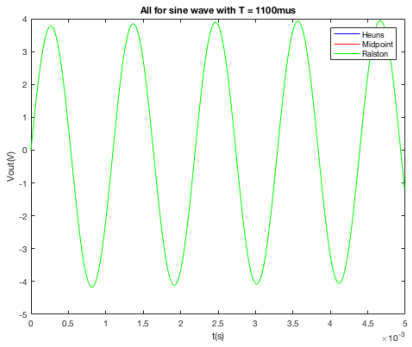
(a) $T = 150\mu s$, $h = 0.000001$ and $t_f = 0.003$



(b) $T = 15\mu s$, h = 0.0000001 and tf = 0.00003



(c) $T = 400\mu s$, $h = 0.00001$ and $t_f = 0.001$

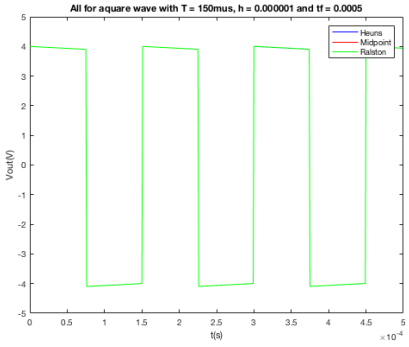


(d) $T = 1100\mu s$, h = 0.0000001 and tf = 0.00003

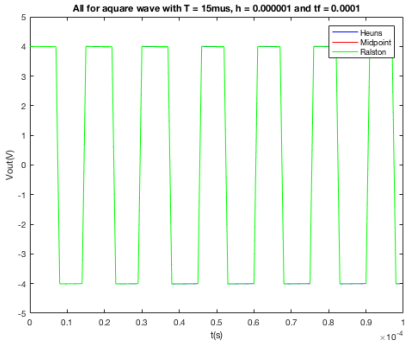
Figure 10: Plots of sine waves for different periods

1.4.4 Square wave with $\tilde{V}_{in} = 4V$

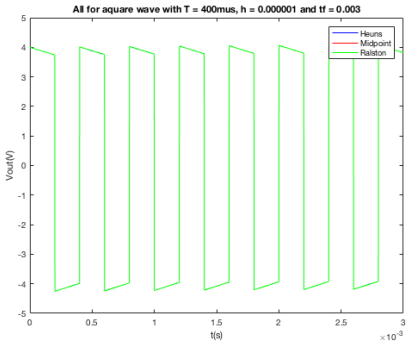
```
vin = 4;
T = ***10^(-6);
t = t0: T : tf;
f = 1/T;
Vin =@(t) vin*square(2*pi*f*t);
```



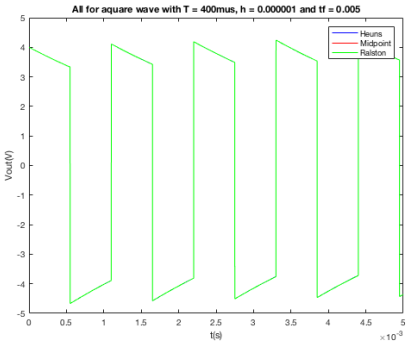
(a) $T = 150\mu s$



(b) $T = 15\mu s$



(c) $T = 400\mu s$

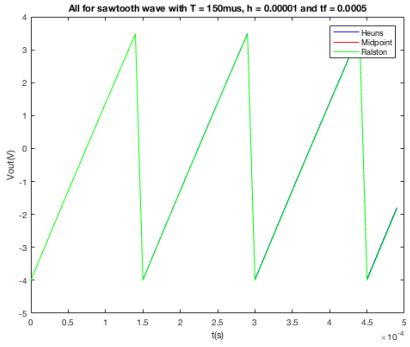


(d) $T = 1100\mu s$

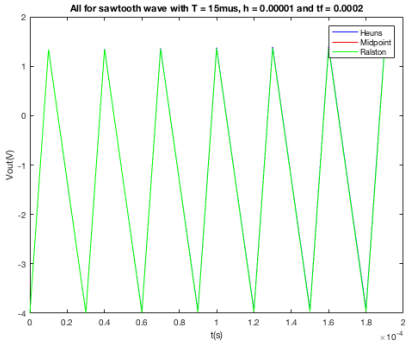
Figure 11: Plots of square waves for different periods

1.4.5 Sawtooth wave with $\tilde{V}_{in} = 4V$

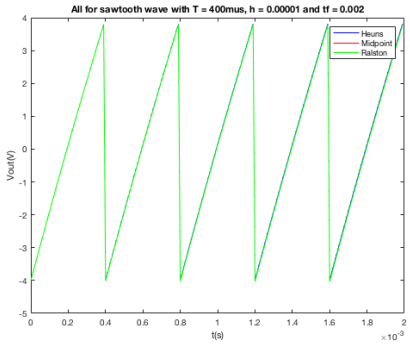
```
vin = 4;
T = ***10^(-6);
t = t0: T : tf;
f = 1/T;
Vin =@(t) vin*sawtooth(2*pi*f*t);
```



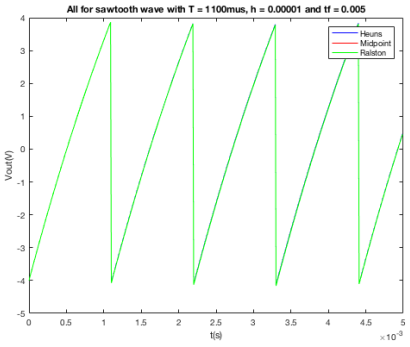
(a) $T = 150\mu s$



(b) $T = 15\mu s$



(c) $T = 400\mu s$



(d) $T = 1100\mu s$

Figure 12: Plots of sawtooth waves for different periods

2 Exercise 3 - RLC Circuit

2.1 Background

From Kirchoff's Voltage Law (KVL) the sum of all the voltages in a closed loop must sum to 0 V. Therefore from this we can derive the equation $V_{in}(t) = v_L(t) + v_R(t) + v_C(t)$. All of the voltages in the closed loop current which flows through them, all being equal to the inductor current. The voltage across a conductor is the inductance multiplied by the derivative of it's current, the voltage across the resistor is equal to it's resistance multiplied by the current travelling through it and the voltage across the capacitor is the charge held within the capacitor divided by the capacitance of the capacitor. Substituting these characteristics into the equation above gives us the equation:

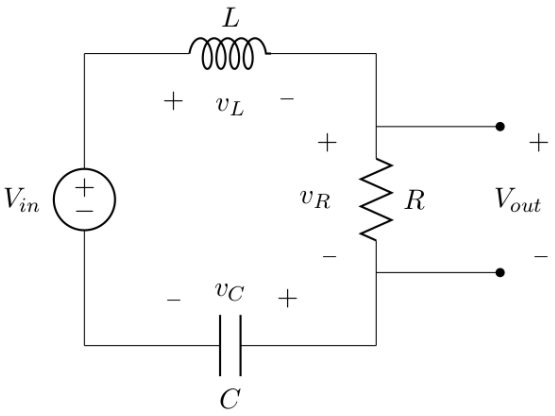


Figure 13: RLC Circuit

$$L\frac{d}{dt}i_L(t) + Ri_L(t) + \frac{1}{C}\int_0^t i_L(t)\,dt = V_{in}(t)$$

Substituting $i_L(t)$ for $\frac{d}{dt}q_C(t)$ you make the equation:

$$L\frac{d^2}{dt^2}q_C(t) + R\frac{d}{dt}q_C(t) + \frac{1}{C}q_C(t) = V_{in}(t)$$

We were then able to turn this second order ODE into a set of 2 coupled first order ODEs through the use of equating $q'_C(t) = y(t)$ and substituting $q_C(t)$ for $x(t)$ we have the coupled equations:

$$y(t) = x'(t) \quad \text{and} \quad y'(t) = \frac{1}{L}[V_{in}(t) - Ry(t) - \frac{1}{C}x(t)]$$

For our project we have values:

$$R = 250\Omega, C = 3\mu F, L = 650mH, i_L(0) = 0\ \ (y_0)\ \text{and}\ \ q_C(0) = 500nC\ \ (x_0).$$

2.2 Script Coupled Equations and Constants

Now that we have these two equations we are able to implement them into MATLAB scripf called 'RLC_script.m' by defining them as:

```
R = 250; L = 650*10^-3; C = 3*10^-6; %Impedance values for the components
w1 = 2*pi*500; %frequency for the 500 Hz sinusoid
w2 = 2*pi*100; %frequency for the sinusoid
w3 = 2*pi*5; %frequency for the sinusoid
tc = 3*10^-6; %tau for the exponential decay input
y0 = 0; x0 = 500*10^-9; t0 = 0; %Initial conditions y = iL, x =qC and t = time
h = 0.00001; %step size
tf = 0.03; %final condition
Vin = @(t) 5;
func1 = @(x, y, t) y; %y = q'
func2 = @(x, y, t) (Vin(t) - R*y - x/C)/L; %the second coupled equation
```

2.3 4th Order Runge Kutta 3/8 Algorithm

2.3.1 Theory

To now evaluate these coupled equations, we use the Runge Kutta 4th order 3/8 algorithm to use these two coupled equations to estimate the next point of the current i_L and the charge q_C. The algorithm is:

$$f_1(x,y,t) = y \quad f_2(x,y,z) = \frac{1}{L}[V_{in}(t) - Ry - \frac{x}{C}]$$

$$k1_x = hf_1(x_i, y_i, t_i) \quad k1_y = hf_2(x_i, y_i, t_i)$$

$$k2_x = hf_1(x_i + \frac{k1_x}{3}, y_i + \frac{k1_y}{3}, t_i + \frac{h}{3}) \quad k2_y = hf_2(x_i + \frac{k1_x}{3}, y_i + \frac{k1_y}{3}, t_i + \frac{h}{3})$$

$$k3_x = hf_1(x_i - \frac{k1_x}{3} + k2_x, y_i - \frac{k1_y}{3} + k2_y, t_i + \frac{2h}{3})$$

$$k3_y = hf_2(x_i - \frac{k1_x}{3} + k2_x, y_i - \frac{k1_y}{3} + k2_y, t_i + \frac{2h}{3})$$

$$k4_x = hf_1(x_i + k1_x - k2_x + k3_x, y_i k1_y - k2_y + k3_y, t_i + h)$$

$$k4_y = hf_2(x_i + k1_x - k2_x + k3_x, y_i k1_y - k2_y + k3_y, t_i + h)$$

$$x_{i+1} = x_i + \frac{k1_x + 3k2_x + 3k3_x + k4_x}{8} \quad y_{i+1} = y_i + \frac{k1_y + 3k2_y + 3k3_y + k4_y}{8}$$

$$t_{i+1} = t_i + h \quad \text{where h is time step}$$

The way that this works is it starts from a single point and calculates a gradient to the next point, using this it then creates another slope, this is then repeated twice more using the equations above and a weighting is given to each gradient, summed together and divided by the total weighting to scale it back to uniform size, from this we are able to estimate more accurately where the next point if going to go for this time step.

2.3.2 Implementation

To implement this into MATLAB we created a function called 'RK4second.m' in which would take in the two functions and work out the k efficients for both x and y and from there, work out the next position of the x (charge) and y (current) coordinates.

```
function [xout, yout] = RK4second (xin, yin, h, tin, func1, func2)

k1x = h*feval(func1, xin ,yin, tin); %approx consts for RK4th ord 3/8 Method
k1y = h*feval(func2, xin ,yin, tin);
k2x = h*feval(func1, xin + k1x/3, yin + k1y/3, tin+h/3);
k2y = h*feval(func2, xin + k1x/3, yin + k1y/3, tin+h/3);
k3x = h*feval(func1, xin  - k1x/3 + k2x, yin - k1y/3 + k2y, tin+2*h/3);
k3y = h*feval(func2, xin  - k1x/3 + k2x, yin - k1y/3 + k2y, tin+2*h/3);
k4x = h*feval(func1, xin + k1x - k2x + k3x, yin + k1y - k2y + k3y, tin+h);
k4y = h*feval(func2, xin + k1x - k2x + k3x, yin + k1y - k2y + k3y, tin+h);

xout = xin + (k1x + 3*k2x + 3*k3x + k4x)/8;
yout = yin + (k1y + 3*k2y + 3*k3y + k4y)/8;
end
```

This takes in the present value of current and charge a certain time, the time position, the step time and the two functions to evaluate and approximate the next point, which is then outputted at the end.

2.4 RLC_Script.m Implementation

```
N = round((tf-t0)/h); %calculates number of time steps
ya = zeros(1,N); xa = zeros(1,N);
ta = zeros(1,N); in = zeros(1,N); %set up arrays

xa(1) = x0; ya(1) = y0; ta(1) = t0; in(1) = Vin(t0); %input initial conditions
for i = 1:N-1 %works out next timestep value
   [xa(i+1), ya(i+1)] = RK4second (xa(i), ya(i), h, ta(i),func1, func2);
    ta(i+1) = ta(i) + h; %next time step
    in(i+1) = Vin(ta(i+1)); %next value of the input
end

figure; %sets up the figures
Vout = R*ya; %Vout = R*iL
subplot(2,1,1);
plot(ta, Vout); %plot output Voltage
grid on;
xlabel('Time/s'); ylabel('Voltage Out/V');
title('R*dq/dt with a Step Response')
subplot(2,1,2);
plot(ta, in); %plot input Voltage
grid on;
xlabel('Time/s'); ylabel('Voltage In/V');
title('Step Response')
```

2.5 Exercise 3 Graphs

2.5.1 Step Response

The step response causes a sudden rise of voltage up to 5 V, due to the inductor not being able to change current instantaneously the voltage out starts from 0 V and as you can see on Figure 2. It increases up to around 1.8 V at this point the capacitor starts to reduce the rate of it charging up due to saturating in charge, as the voltage out starts to tend to 0, the capacitor emits charge causing a negative current through the inductor. As this goes on, the voltage across the capacitor tends to 5 V causing no current to flow through the inductor and the resistor, causing the output voltage to tend to 0 V.

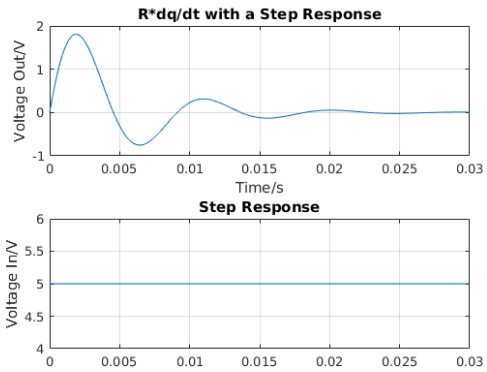


Figure 14: $V_{in}(t) = 5$

2.5.2 Sine Wave Input

In Figure 3 the input is a sine wave of amplitude 5 V and a frequency of 5 Hz, as the sine wave moves away from zero it causes a sharp increase in the current going through the inductor and therefore causes a sharp rise on the voltage out, this then attenuates to form the shape of a sine wave of amplitude is ≈ 0.1 V and leads the input voltage by a phase difference of $\frac{\pi}{2}$ this is relative to the frequency of the input and the complex impedance

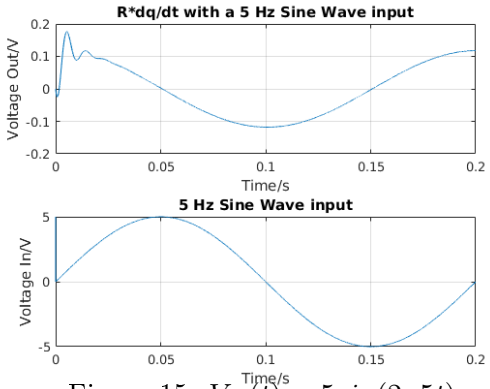


Figure 15: $V_{in}(t) = 5sin(2\pi 5t)$

of both the inductor and the capacitor being $j\omega L$ and $\frac{1}{j\omega C}$ respectively. We chose the final time as 0.2 s to show how initially the output changes but then follows the shape of the input eventually.

In Figure 4 the input is a sine wave of amplitude 5 V and a frequency of 100 Hz. Upon the start of the wave, the output voltage rises slower than the voltage of the input, due to the fact that the rate of change of the inductor current cannot shange instantaneously and will eventually equal the frequency of the input. This causes a slight phase difference and the amplitude is far greater than the 5 Hz input, resulting in an amplitude of ≈ 4 V. This greater output voltage is due to being a band pass filter, low frequencies as we can see, 5 Hz for example attenuate to a value close to 0 V whereas at this frequency the output voltage is close to the input voltage and at some frequency greater than this, the same result as the 5 Hz wave will be apparent.

In Figure 5 the input is a sine wave of amplitude 5 V and a frequency of 500 Hz. The response is very similar to Figure 3's output signal, rising initially to a value that is greater than its steady state amplitude in which is ≈ 0.5 V which is far 10 times less than the input amplitude, and less than the the amplitude of the Figure 4. This suggests that the center of the band pass filter is based around 100 Hz as is seen in Figure 4 and therefore the outer bounds falling in amplitude at frequencies greater and less than this.

2.5.3 Square Wave Inputs

In Figure 6 the input is a square wave of frequency 5 Hz and amplitude 5 V. At this frequency, the response is identical to the response of a step response due to there being enough time for the wave to return to 0 V after the change in amplitude, the only difference is when the voltage drops to -5 V the step response is equal to the positive step response but negative. The time scale was chosen to show 1 and a quarter wavelengths so that it is easy to see that each time the sign changes, the response is identical.

In Figure 7 the input is a square wave of frequency 100 Hz and amplitude 5 V. At this frequency, the output follows the characteristic of a sine wave due to the step response giving a sinusoidal peak, this is repeated but switched in direction as the sign of the input changes, causing a periodic signal

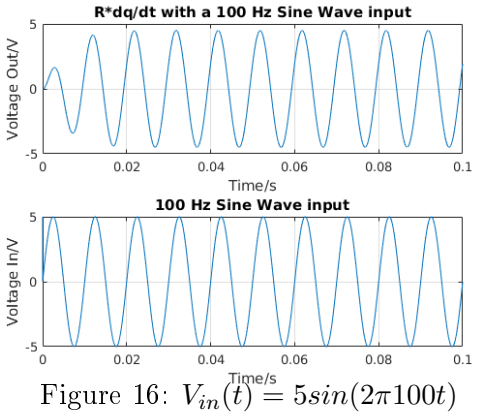


Figure 16: $V_{in}(t) = 5sin(2\pi 100t)$

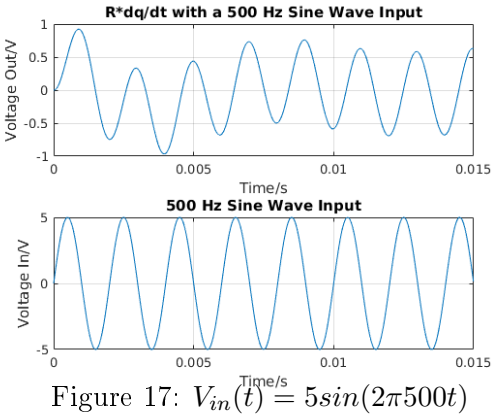


Figure 17: $V_{in}(t) = 5sin(2\pi 500t)$

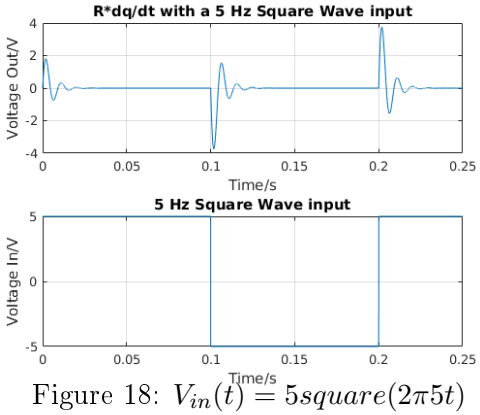


Figure 18: $V_{in}(t) = 5square(2\pi 5t)$

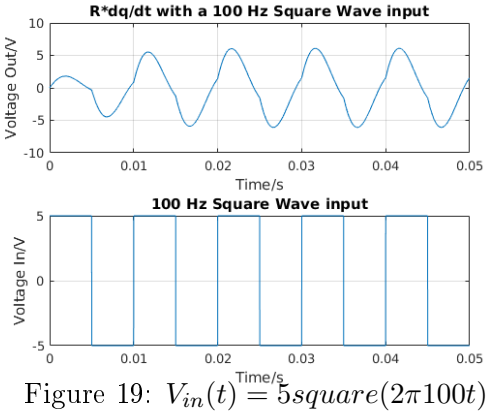


Figure 19: $V_{in}(t) = 5square(2\pi 100t)$

on the output respresenting something somewhat similar to a sine wave.

In Figure 8, the input is a square wave of frequency 500 Hz and amplitude 5 V. At this frequency, the step responses cause an output signal which is similar to a periodic tent function of exponential charge and discharge, this is due to the fact that the square wave changes sign too rapidly for the signal to smooth off or return to zero before the next half wavelength. As it gets to the 4th period of the input, the output settles to an amplitude of ≈ 1 V. In which looking back over Figures 6, 7 and 8. The bandpass filter seems to give similar changes in output amplitude relative to the input amplitude, where at 100 Hz, in Figure 7 the magnitude is greater than both magnitudes. Confirming that the center frequency if based around 100 Hz and the bandwidth is based between 5 Hz and 500 Hz.

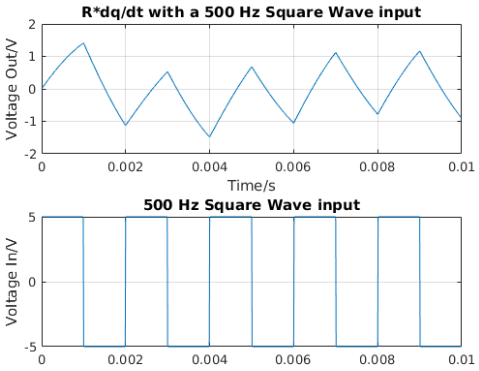


Figure 20: $V_{in}(t) = 5square(2\pi 500t)$

2.5.4 Exponential Decay Input

In Figure 9, the input is a decaying exponential in which it's time constant $\tau_C = 3(ms)^2$ causing a sharp drop in amplitude towards 0 V in a time of less than 5 ms, causing an output in which is incredibly similar to the output of the step responss (Figure 2). The only main difference being that the max amplitude of the output of Figure 9 is ≈ 1 V, ≈ 0.8 V less than the max amplitude of Figure 2 in which the amplitude was ≈ 1.8 V.

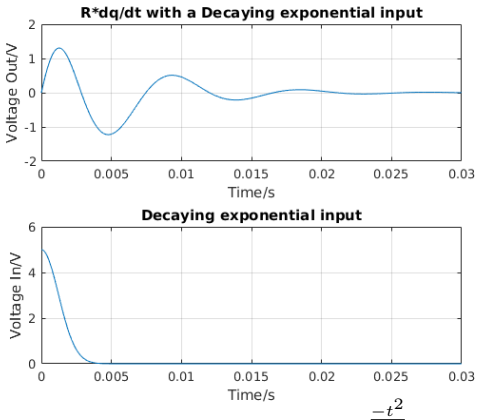


Figure 21: $V_{in}(t) = 5e^{\frac{-t^2}{\tau_C}}$

3 Exercise 4 - Finite Differences for PDE

3.1 Background

For this exercise we were asked to implement the finite difference method for the 1D heat equation which is:

$$\frac{\partial y}{\partial t} = \frac{\partial^2 x}{\partial t^2} \quad \text{in which} \quad 0 < x < 1, \quad t > 0 \quad \text{and} \quad y(0,t) = y(1,t) = 0 \ \forall \ t$$

3.2 Script Equations and Constants

```
x0 = 0;
h = 0.005; %Step size in the X direction
Tin = @(x) sin(2*pi*x);
Bc = @(t) 0;
xf = 1 + h; %Final value of X plus one more step to complete the graph
t0 = 0; %Initial time value
k = 0.45*h^2; %v max of 1/2 we use 0.45 work out k relatively, steps in time
tf = 200*k; %the end time is 250 steps in the k direction
N = round((xf-x0)/h); %number of steps in the x direction
M = round((tf - t0)/k); %number of steps in the t direction
v = k/(h^2); %recalculating the value of v
b = 1 - 2*v; %create the value of beta
ya = zeros(1,N); xa = zeros(1,N); %create empty arrays
xa(1) = x0; xa(N) = xf-h;  ya(1) = Bc(0); ya(N) = Bc(0); %initial conditions
```

In the script above, we define the step size in the x direction, from this we are then able to calculate k, the step in time. Due to the formula to calculate finite difference:

$$U_j^{m+1} = vU_{j-1}^m + (1-2v)U_j^m + vU_{j+1}^m \quad \text{where for stability} \quad v < \frac{1}{2}$$

Therefore by making v = 0.45, thus never becoming unstable, if the x direction step size is changes, we can calculate k by multiplying it by h^2 due to $v = \frac{k}{h^2}$ we also replaced $1 - 2v$ with β.

3.3 Implementation of the Finite Difference Method

```
for i = 1:N-2
    xa(i+1) = xa(i) + h; %next step in x direction
    ya(i+1) = feval(Tin, xa(i+1)); %calculate heat inital points
end
t = t0; %initial time value
plot(xa, ya, '.', 'markersize', 4); %plot graph at t = 0
hold on; %plots them all on the same graph
yn = ya; %to take the next time values
for a = 1:15 %plots the graph 15 times
    for i = 1:M %cycle through every time step
        ya(1) = Bc(t); ya(N) = Bc(t); %set boundary conditions
       for j = 2:N-1 %work out temp for next time step
           yn(j) = v*ya(j-1) + b*ya(j) + v*ya(j+1);
       end
        t = t+k; %change the time one step forward
        ya = yn; %replace ya with the next time array
    end
    plot(xa, ya, '.', 'markersize', 4); %plot heat
```

```
    title 'Finite Difference of PDE 1D Heat Equation - '
end
xlabel 'Distance/cm'
ylabel 'Temperature/C'
```

The way that it works is that we use a for loop to calculate the initial temperature value at each spot on the x axis and place them into the array ya, this then is plotted on a graph to show the initial graph. After that we then use a for loop which runs 15 times so that we are able to plot each graph whilst being able to visualise how it changes as time increases. There is then another for loop in which cycles through every step, in which recalculates the boundary conditions (for the sake of having a time variable boundary conditions). Finally within another for loop it cycles through all the points between the first and last values of temperature and using the finite difference equation stated above. After this ya is replaced with the new calculated array and then the values are plotted on the graph.

3.4 Finite Difference Graphs

As is shown in Figure 10, for the tent function the temperature decreases as it tends towards 0 and 1 from the centre of 0.5. As it steps through time, the heat tries to diffuse evenly within the object, this is shown by as time goes on, the heat decreases within the body and if we were to go further into time, the heat would altogether tend to zero across the body.

In Figure 11 you can see the sine function in which meets the boundary conditions, having 'positive' and 'negative' temperatures, they have identical characterisics of how the temperature is dissipated. The central point at x = 0.5 remains equal to 0 due to the equation taking both points either side of it and multiplying with v, in which they both have the same magnitude, but different signs so when added together they cancel and cause the temperature to remain at 0 for the time being.

In Figure 12, the absolute value of the sine wave was taken in which the reason it differs with the standard sine curve is that due to now across the centre, on either side the point they are no longer opposite in magnitude, therefore the point at the centre increases in magnitude as the heat disperses across the body, trying to uniformally distribute it.

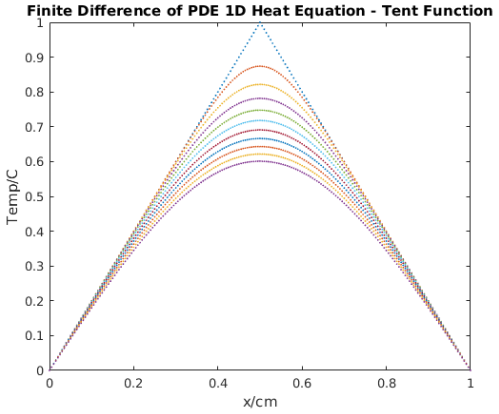


Figure 22: $T_{in}(x) = tent(x)$

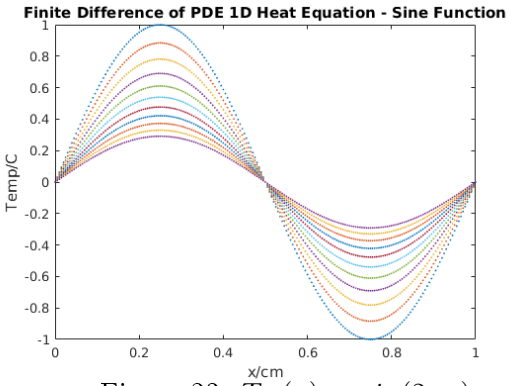


Figure 23: $T_{in}(x) = sin(2\pi x)$

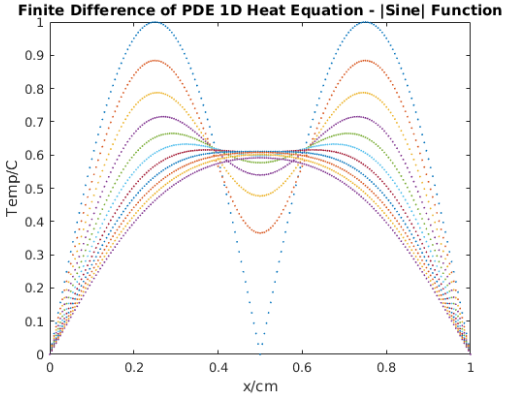


Figure 24: $T_{in}(x) = |sin(2\pi x)|$

In Figure 13, the input is half of a sine wave in which you can simply see as time goes on, throughout the body, the heat dissipates, tending towards 0.

In Figure 14, the input is a polynomial with the roots $x = 0, 0.3, 0.5$ and 1 showing a very random distribution of the heat throughout the body, at the start of time, the heat from $0 < x < 0.5$ smooths off a lot quicker than the second half of the graph due to the fact that there is a maximum at the point of $x \approx 0.83$ this takes a while for the heat to dissipate. As with all the rest, as time travels towards infinity, the heat will tend to 0, unifomally across the body.

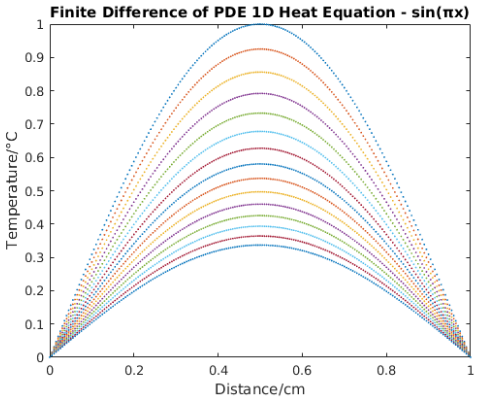


Figure 25: $T_{in}(x) = sin(\pi x)$

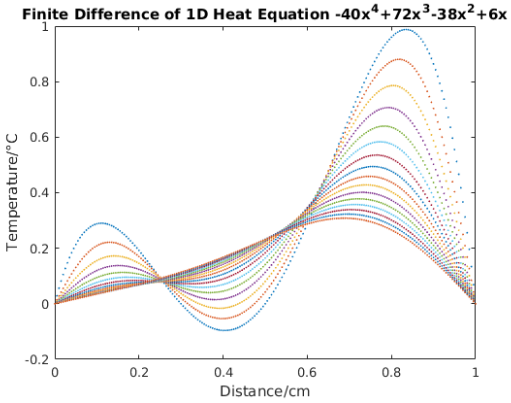


Figure 26: $T_{in}(x) = -40x^4 + 72x^3 - 38x^2 + 6x$

3.5 Bonus Exercises

3.5.1 Boundary Conditions Not Met

In Figure 15, the input was the straigt line of $T_in(x) = 1 - x$ where the left hand portion of the graph doesn't meet the left hand condition. With the boundary condition both being 0, it causes the temperature from just after x = 0 to fall away considerably quicker than any other portion of the graph due to the greater difference the boundary condition and the points to the side of it, as you can see on Figure 15.

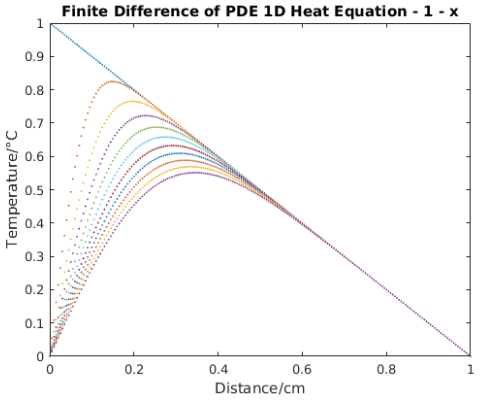


Figure 27: $T_{in}(x) = 1 - x$

3.5.2 Non Zero Boundary Conditions

In Figure 16, the input is a tent function, but rather than having boundary conditions in which were equal to 0, the boundary condition at x = 0, the temperature is equal to 0.75 and at x = 1, the value is equal to 0.25. As you can see the points to the left of the graph all pull up towards the value of 0.75 and on the other side, all the points are pulled towards the 0.25. As time goes to infinity, you would get the heat distributed as a straight line going from 0.75 to 0.25.

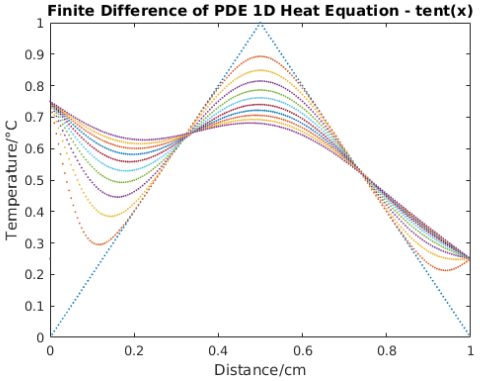


Figure 28:
$T_{in}(x) = tent(x)$ $y_0(t) = 0.75, y_1(t) = 0.25$

3.5.3 Time Varying Boundary Conditions

In Figure 17, the input is a tent function, initially with zero boundary conditions, but as time moves forward, the temperature at the end points increases with the equation of $y_0(t) = y_1(t) = 20t$. So the heat in the body will end up heating up from the edges inwards.

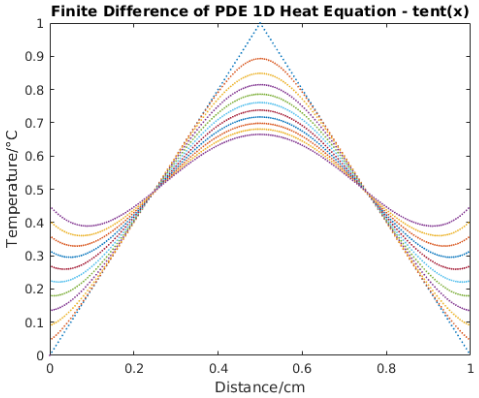


Figure 29: $T_{in}(x) = tent(x)$ $y_0(t) = y_1(t) = 20t$