```
%{
Aidan Chin ECE 213 Spring 2024
5/6/2024
--- Exercise SS-C1: Understanding Convolution ---
this code will solve for output voltage of convolution
between 2 functions h(t) and x(t)
%}

% -------- Initialize -------

clc % clear terminal
clf   % clear all figures
clear   % remove all variables from the workspace

% --------  Given ------

R = 10; % Resistor value in Ohms
C = 2; % Capacitor value in mF
V0 = 12; % Voltage (V)
tau = R*C; % Time constant
n = 4000; % Number of intervals
t0 = 0; % Start time
tf = 8 * tau; % End time
t = linspace(t0, tf, n+1); % Create overall time array, Because C is in mF,
                           % tau = RC is in ms
t2 = 2*tau; % the point in time where x becomes 0
t2i = (t2/tf)*(n); % the index in t where x becomes 0

h = (1/tau)*exp(-t/tau); % create function to that we will convolute

% -------- Calculations -------

% hand calculation

c1 = V0*(1-exp(-t/tau)); % convolution answer for 0<t<2tau
c2 = V0*(1-exp(-2))*(exp(-(t-t2)/tau)); % convolution answer t<2tau

% matlab calculation

ym = zeros(1, n+1); % initialize the convolution bounds
dr = (tf - t0)/n; % define step size scaler
for in = 1:n % iterate over whole time of convolution
    for k = 1:length(h) % iterate over whole length of h
        if in - k + 1 > 0 && in - k + 1 <= length(h) % represent x(t) using if
                                                     % statement with time shift
            if in - k + 1 < t2i % only add to integration when v = V0
                ym(in) = ym(in) + V0 * h(k) * dr; % perform integration v = V0
            end
        end
    end
end
```

```
52
53 % -------- Plotting -------
54
55 plot(t, c1,':', t, c2, ':', t, ym, 'LineWidth', 2);
56 ylim([0 V0+1]); % adjust limits
57 legend('0<t<2\tau', 't>2\tau', 'y(t)','Location', 'east'); % create legend
58 title('Exercise SS-C1: Understanding Convolution Aidan Chin'); % create title
59 xlabel('time t (ms)');  % Custom x-axis label
60 ylabel('Voltage (V)');  % Custom y-axis label
61 xticks(0:tau:tf); % define x-axis steps
62 grid on;
63 % -------- Checks --------
64
65 % analytical check, should be near 0
66
67 analyticalCheck = c1(t2i) - c2(t2i)
68
69 % Numerical Checks
70
71 % Check if maximums are the same, if analytical check passes, this should
72 % also be near 0
73
74 [ymy, ymx] = max(ym); % find position of max of ym
75
76 yCheck1 = c1(t2i) - ymy
77 yCheck2 = c2(t2i) - ymy
78
79 % Check if maximums occur at the same time, should near 0
80 xCheck = t(ymx) - t2
```