Lab Session 08
Performance Characteristics of 1st & 2nd Order Systems

Exercise 01 – Time Constant of a 1st Order RC System

Given the values of R and C, obtain the unit step response of the first order system.

- **System 1 – R = 2 kΩ, C = 0.01 F**
- **System 2 – R = 2.5 kΩ, C = 0.003 F**

Verify in each case that the calculated time constant ($\tau = RC$) and the one measured from the figure as 63% of the final value are the same. Also obtain the steady state value of each system.

Solution

Code 01 is a MATLAB program that investigates the step response of a simple RC circuit by measuring the voltage drop across its capacitor in response to a unit step input voltage applied at time $t = 0$. Specifically, it explores how the value of the system's time constant $\tau = R \times C$ affects the speed of the response, and in doing so illustrates the effect of time constant on the response of a 1st order system.

Code 01: Step Response of a RC Circuit

```
%%  FCS Lab Session 08 - Exercise 01: Step Response of First Order System
%   Saad Mashkoor Siddiqui, EE-16163, Section D, TE-EE 16-17
%   Tests the step response of 1st order RC ckts with different time constants

%%  Preparing workspace
clear all; close all; clc;

%%  Initializibng data
R1 = 2e3; R2 = 2.5e3;           % Ohms
C1 = 0.01; C2 = 0.003;          % Farads
tau_1 = R1 * C1;                % Time constant for first circuit
tau_2 = R2 * C2;                % Time constant for second circuit
```

```
%   Defining different time domain for each system. Upper limit is 5 * tau
%   because all transient behaviour has ceased by approximately this time.
t_sys_1 = 0: 0.01 : tau_1 * 5;
t_sys_2 = 0: 0.01 : tau_2 * 5;

%% Initializing transfer functions for both circuits
sys_1 = tf([0 1], [tau_1, 1]);
sys_2 = tf([0 1], [tau_2, 1]);

%% Storing step response data for both inputs
step_data_sys_1 = stepinfo(sys_1);
step_data_sys_2 = stepinfo(sys_2);

%% Visualizing step responses
%   System 1
figure(); step(sys_1, t_sys_1, 'k'); grid on; xlabel('Time (\it{t/s})');
ylabel('Control Signal (\it{v_C(t)/V})');
title(sprintf('RC System 1 Step Response - R = %.2f, C = %.2f', R1, C1));

%   System 2
figure(); step(sys_2, t_sys_2, 'k'); grid on; xlabel('Time (\it{t/s})');
ylabel('Control Signal (\it{v_C(t)/V})');
title(sprintf('RC System 2 Step Response - R = %.2f, C = %.3f', R2, C2));

%   System 1 and 2 on the same graph
figure(); step(sys_1, t_sys_1, 'k'); hold on; step(sys_2, t_sys_1, 'k--');
legend = ['System 1', 'System 2']; grid on;
xlabel('Time (\it{t/s})'); ylabel('Control Signal (\it{v_C(t)/V})');
title('Comparing Responses for Both Systems');
```

Program Explanation

The code first initializes two different resistance and capacitance values for two different RC circuits. It then calculates the time constant for each of these systems as `tau_1` and `tau_2` respectively. The program defines a separate time domain for each system that spans a duration from 0s to 5 times the system's time constant. This is because all transient behaviour has virtually ceased to exist after 5 time constants, which means any remaining response will be the system's steady behaviour. The program then defines a transfer function for each system and passes it to the MATLAB step function along with the system's corresponding time domain. This generates plots of each system's step response, both in separate figures as well as on the same set of axes to allow for better comparison of the two system's and their respective responses.

Program Output

Figures 1, 2, and 3 show the plots generated by Code 01. Figures 1 and 2 show the step response of systems 1 and 2 as separate graphs respectively, while figure 3 juxtaposes them on the same axes to compare the responses of both systems.

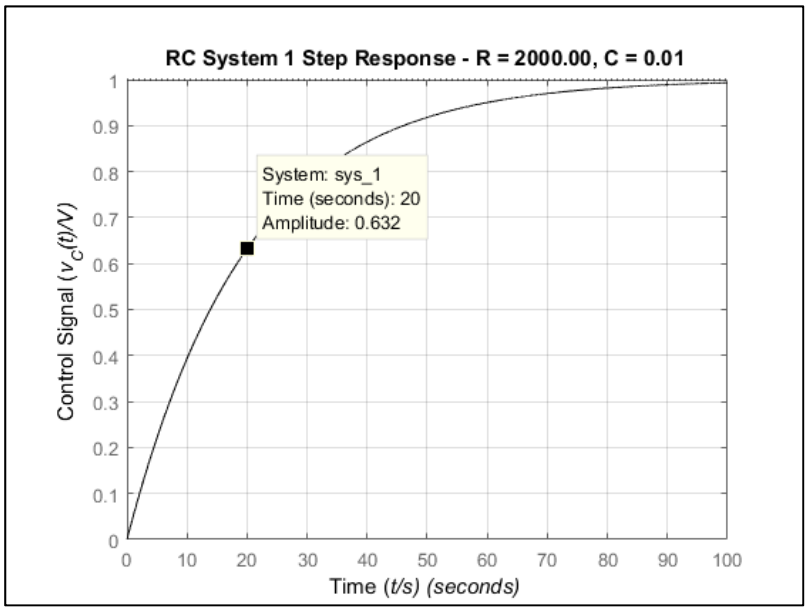

Figure 1: Unit Step Response for System 1

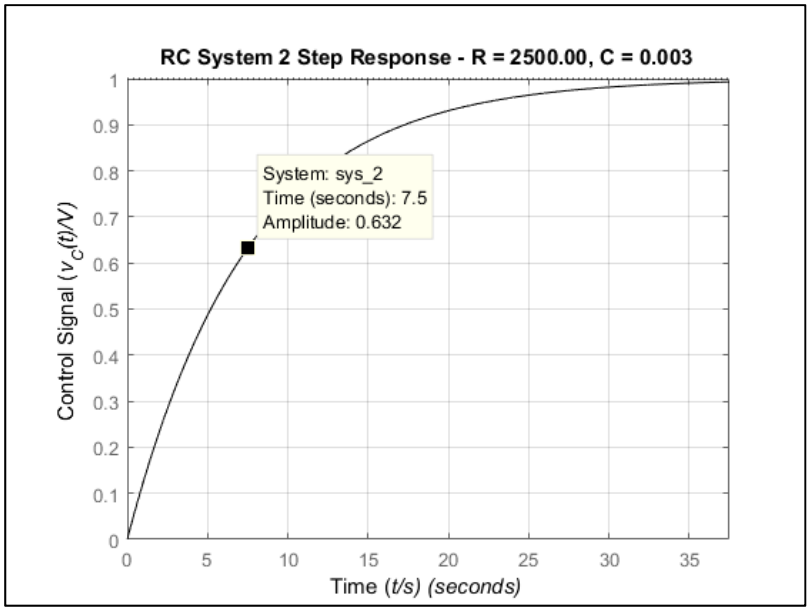

Figure 2: Unit Step Response of System 2

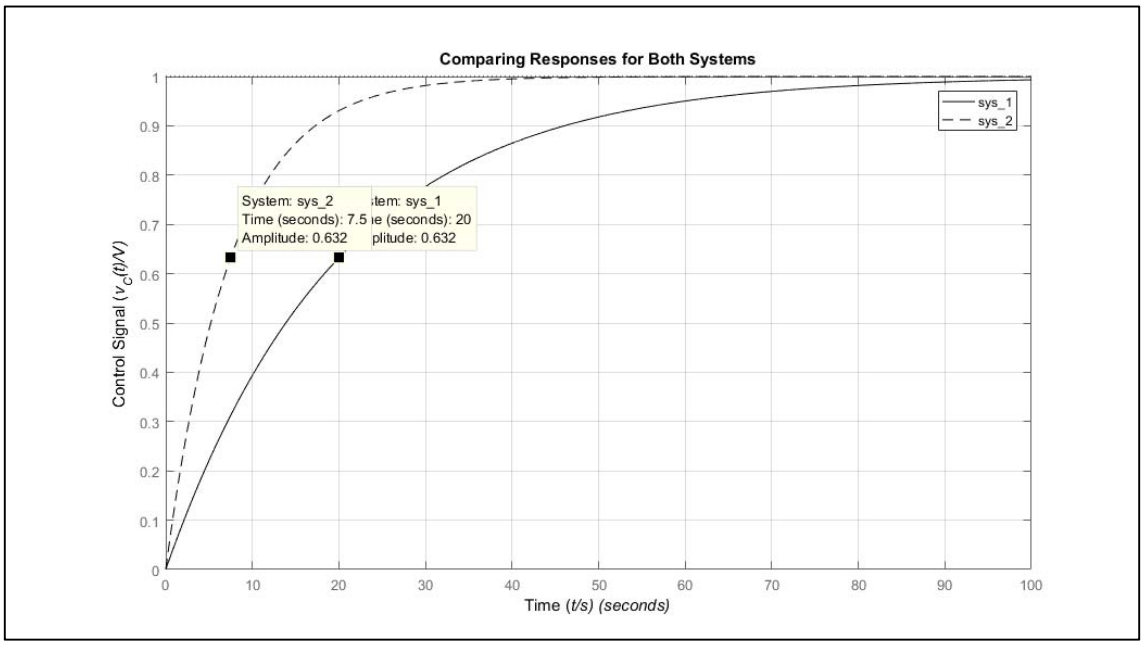

Figure 3: Comparing Responses - System 2 clearly has a faster response time

Theoretical Time Constant Calculation

The theoretical time constant for each system is calculated as follows

$$\tau_1 = R_1 \times C_1 = 2\ k\Omega \times 0.01\ F = 20s$$
$$\tau_2 = R_2 \times C_2 = 2.5\ k\Omega \times 0.003\ F = 7.5s$$

Analysis of Results

- The steady state value for both systems is 1 V - this is the voltage that appears across the capacitor when the transient response of the circuit has died down.
- The voltage across the capacitor is 1 V in steady state condition because the capacitor essentially becomes an open circuit to the step DC input voltage after its transient current (charging current) has dissipated.
- In both systems, the theoretical time constant is the same as the one derived from the figure. The data cursors were placed at points along the graph where the response value was 0.632 (~63.2% of steady state value) as this is the definition of the time constant.
- System 2 has a smaller time constant than system 1, which is why it reaches its steady state condition much more quickly than system 2, as is shown in Figure 3.
- This investigation thus proves that the time constant of a first order system affects only the speed of its response and not the magnitude of its steady state value - the only difference between the responses of `sys_1` and `sys_2` is their duration, not their shape.

Exercise 02 – Effect of Damping Factor ζ on 2nd Order System Performance

Find the step response of the system for the values of $\omega_n = 1$ and $\zeta = 0.1, 0.4, 0.7, 1.0, 2.0$. Plot all the results in the same figure and tabulate the rise time, peak time, percentage overshoot, settling time, and steady state value for each value of ζ.

Solution

Code 02 is a MATLAB program that issued to investigate the step response of a generic 2nd order control system for a fixed value of natural frequency $\omega_n = 1\frac{rad}{s}$ and different values damping factor ζ. In doing so, the program both visualizes and quantifies the step response of a 2nd order system while emphasizing the role of the damping factor.

Code 02: Step Response of a Second Order System

```
%%  FCS Lab Session 08 - Exercise 02: Step Response of Second Order System
%   Saad Mashkoor Siddiqui, EE-16163, Section D, TE-EE 16-17
%   Tests the step response of 1st order RC circuits with different time constants

%%  Preparing workspace
clear all; close all; clc;

%%  Initializing parameters
nat_freq = 1;                                   % radians/second
damping_factors = [0.1, 0.4, 0.7, 1.0, 2.0];  % several values of damping factor
step_info_array = [];                           % performance params for each zeta
step_response_array = [];                       % step response for each zeta
zeta_index = 1;                                 % counter variable
num = [nat_freq^2];                             % numerator for transfer function
t = 0:0.01:50;                                  % investigating response for 50s

%%  Getting Step Responses for all values of damping factor
%   Creating figure on which the for loop will add plots
figure(); grid on; xlabel('Time (\it{t/s})');
ylabel('Control Signal \it{c(t)/arbitrary units})');
title('Comparing 2^{nd} Order System Responses');

for zeta = damping_factors
    % Create a transfer function for this value of eta
    transfer_fcn = tf(num, [1, 2 * zeta * nat_freq, nat_freq^2]);
```

```
    % Get step response for this transfer function - store in separate column
    step_response_array(:, zeta_index) = step(transfer_fcn, t);

    % Store step info for this transfer function in a separate column
    step_info_array(:, zeta_index) = stepinfo(transfer_fcn);

    % start holding plots from first response
    if zeta_index == 1
        hold on;
    end

    % plot the step response
    plot(step_response_array(:, zeta_index));

    % Increment counter
    zeta_index = zeta_index + 1;
end

%   Once all plots have been added, create a legend
legend('0.1', '0.4', '0.7', '1.0', '2.0');
```

Program Output

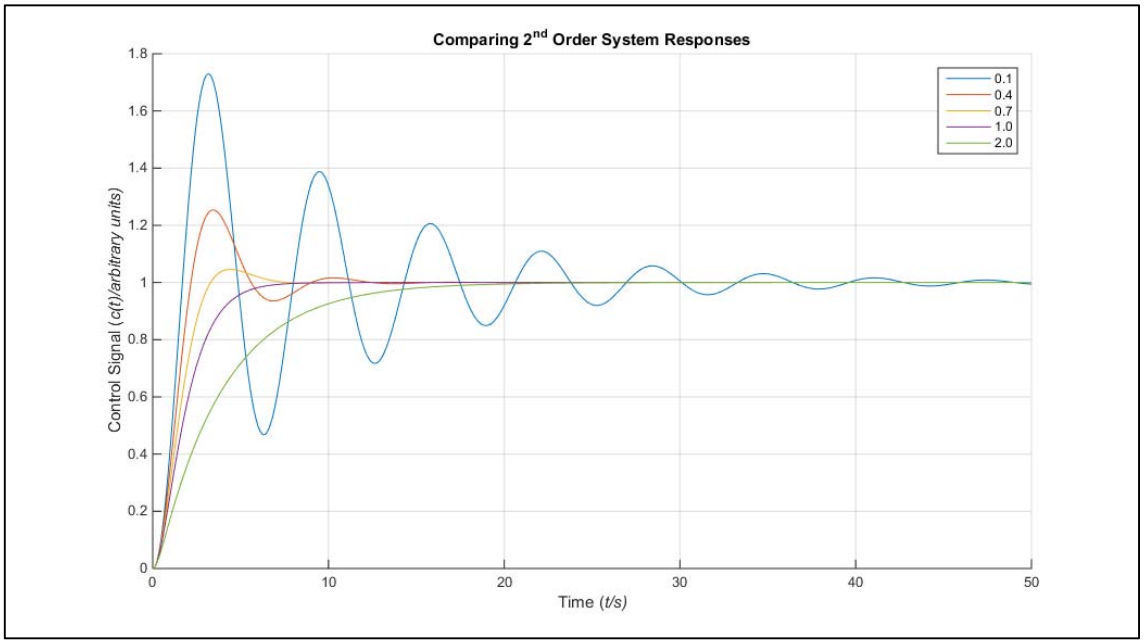

Figure 4: 2nd Order System Response for Different Values of ζ

Table of Performance Metrics

The performance parameters for each system were obtained using MATLAB's built-in `stepinfo()` function, and are tabulated below.

Table 1: Performance Characteristics of Different Second Order Systems

System Number	ζ	Rise Time	Peak Time	% OS	Settling Time	Steady State Value
1	0.1	1.12718	3.14159	72.9156	38.373	1
2	0.2	1.46521	3.45388	25.3741	8.4094	1
3	0.7	2.1268	4.40781	4.59862	5.97887	1
4	1	3.35792	9.79	0	5.83393	1
5	2	8.23075	27.3269	0	14.8789	1

Analysis of Results

- Increasing the damping factor minimizes the magnitude of oscillations in the system's response, albeit at the cost of response speed.
- This is evident from the response for system 5: with the highest damping factor (2.0), this system has virtually no oscillations and %age overshoot, but has the longest rise, peak, and settling times.
- In contrast, too low a value of damping factor ζ can increase response speed but also cause large percentage overshoot and many oscillations, as is evident from the response for system 1.
- In fact, even with the increase in response speed, the system may take much longer to stabilize if the damping factor is too low. This is evident from the step response for system 1, which has the lowest rise and peak times but still takes the longest time to reach its steady state value.

- From the visualization of the systems' responses, system 1 is clearly underdamped, as its oscillations are both large in amplitude and duration. Systems 2 and 3 are significantly less underdamped, while system 5 is overdamped.
- System 4 appears to be critically damped - it has virtually no overshoot and a relatively low settling time, although a pole-zero map is necessary to confirm this.
- The steady state value of all systems is 1, even though their waveforms are drastically different - this suggests that the damping factor does not affect the steady state value, but does affect the speed of the responses as well as other characteristics of the response.
- The damping factor in a 2^{nd} order system plays a similar role to the time constant of a 1^{st} order system. However, unlike the latter, the former affects both the duration and shape of the response.

Exercise 03 - Programmatically Determining Step Response Parameters for a 2^{nd} Order System

Understand the code given below for the time specification of a second order system.

Code 03 - Time Specification for a 2^{nd} Order System's Step Response

```
function steptimespec % find time specification for step response of a second
%order system and calculates the rise time,
%delay time, maximum overshoot, peak time and settling time of a system
%whose  damping ratio and natural frequency are known.
clc;
zeta=input('Enter the value of damping ratio ');
wn=input('Enter the value of Natural frequency ');
n=wn*wn;
d=[1 2*zeta*wn wn*wn];
disp('The transfer function is: ')
printsys(n,d);
t=0:0.02:6.0;
[y,x,t]=step(n,d,t);
plot(t,y);
grid on;
title('step response');
%to find rise time i.e. time taken for output to rise from 10% to 90%
k=1;
while y(k)<=0.1;
    k=k+1;
```

```
end
tenpercent=t(k);
while y(k)<=0.9;
    k=k+1;
end
nintypercent=t(k);
rtime=nintypercent-tenpercent;
fprintf('The rise time is: %f sec \n',rtime);
format short
% to find delay time i.e. time taken to rise to 50% of step
k=1;
while y(k)<=0.5;
    k=k+1;
end
dtime=t(k);
fprintf('The delay time is: %f sec\n', dtime);
% to find maximum overshoot
for k=1:1:300;
    if y(k+1)<=y(k);
        % to find value of k till response keeps rising
        break;
    end;
end;
Oshoot=y(k)-1;
fprintf('The overshoot is: %f sec\n', Oshoot);
% to find the peak time
tp=t(k);
fprintf('the peak time is :%f sec\n',tp)
% to find the settling time
%maximum tolerance for considering output to be in steady state taken as 2%
tol=0.02;
for k=300:-1:2;
    if(abs(y(k)-y(300))>tol)
        break;
    end;
end;
stime=t(k);
fprintf('the settling time is :%f sec\n',stime)
```

Program Explanation

This program is a manual implementation of some of the functionality of the MATLAB `stepinfo` function, which provides information about the time characteristics of a 2^{nd} order system's response.

The program first defines the system's transfer function and uses MATLAB's built-in `step` function to store the response as an array of values. It then parses these values to get each of the following time characteristics:

- Rise Time: Uses while loops to find the indexes of response values for which the response is 0.1 (10% of the steady state value) and 0.9 (90% of the steady state value) for this system. It then subtracts the 0.1 index from the 0.9 index to yield the rise time in seconds.

$$t_{Rise} = t_{90\%} - t_{10\%}$$

- Delay Time: Continues to implement a counter variable to keep track of the response array indexes parsed until it finds the index corresponding to a response value of 0.5 This index represents the delay time i.e. the time taken by the system to reach 50% of its steady state value.

$$t_{Delay} = t_{50\%}$$

- Maximum Overshoot: Continues to increment an index counter variable until the current value of the response becomes less than the previous recorded response value. This suggests that the previous value was the peak value of the response. The steady state value is then subtracted from the steady state value and the difference is expressed as a percentage of the latter.

  - $\%age\ OS = \frac{c(t_{peak}) - c(t_{steady\ state})}{c(t_{steady\ state})} \times 100\%$

  In this program, the steady state value is simply 1, so the difference between the steady state value and peak value is numerically equal to the percentage overshoot.

- Settling Time: The time taken for the system to reach its steady state value. Determining the settling time for a numerical response such as the one calculated by MATLAB in this program requires a tolerance level - a threshold that is used to assess whether the difference between two consecutive response values is sufficiently low for the system to have reached its steady state value. The program defines such a tolerance level and then uses it to compare the current value of the system's response with its last recorded value

$$|\, v(t) - v(t_n)| < \ tolerance$$

  However, instead of doing so from the first index (and thus the first instant of time), the program optimizes the process by comparing values towards the end of the response array. This is because the system is more likely to have reached its steady state value towards the end of the investigation period, which means the process will require fewer comparisons.

  Implicit in this implementation is the assumption that the system will indeed have reached its steady state value towards the end of the observation period, and that the last value in the system's response will be the same as its steady state value. This may not always be the case.