Lab Session 09 System Response for Ramp and Arbitrary Input Signals

Exercise 01 – Ramp Input Response

Obtain the ramp response of the following transfer function.

$$G(s) = \frac{30}{s^2 + 5s + 30}$$

Solution

Code 01 is a MATLAB program that visualizes the response of transfer function $G(s)$ as defined above to a ramp input signal.

Code 01: Ramp Response

```
%%  FCS Lab Session 09 Task 01 - Ramp Response
%   Saad Mashkoor Siddiqui, EE-16163, Section D, TE-EE 16-17

%%  Prepare workspace
clear all; close all; clc;

%%  Define original transfer function
num_0 = [0 0 30];
denom_0 = [1 5 30];
tf_0 = tf(num_0, denom_0);

%%  Get and plot step response for the system
step_response_tf_0 = step(tf_0);
figure(); plot(step_response_tf_0, 'k'); xlabel('Time (\it{t/s})');
ylabel('Amplitude (\it{c(t)/arbitrary units})'); grid;
title('Step Response of G(s)');

%%  Modify numerator and denominator for ramp response
num_1 = [0 num_0];
denom_1 = [denom_0 0];
tf_1 = tf(num_1, denom_1);          % same as multiplying by 1/s
```

```
%%  Find the step response for the new tf - this is actually the ramp response
[y1, x1, t1] = step(num_1, denom_1);

%%  Plotting the output and ramp function against time
figure();
axes_limits = [0 10 0 10];      %   [x_lim_low, x_lim_high, y_lim_low, y_lim_high]
plot(t1, y1, 'k');              %   plot ramp response
axis(axes_limits); hold on;     %   also add axes limits to shrink response
plot(t1, t1, 'k--');            %   plot ramp function for comparison

%   Annotate the plot
grid;   title('Plot of unit ramp response of G(s) = 30/(s^2 + 5s + 30)');
xlabel('Time (\it{t/s})'); ylabel('Amplitude (\it{r(t)/arbitrary units})');
legend([{'Ramp Response'}, {'Ramp Input'}]);
```

Program Explanation

The program first initializes the system transfer function by defining its numerator and denominator as arrays of coefficients in descending powers of s and passing both to the built-in `tf` function. It then plots the step response for this transfer function in a separate figure to better visualize any similarities or differences between the system's step and ramp response (plotted later).

The program then modifies the transfer function by zero padding the beginning and end of the numerator and denominator respectively. This increases the order of the transfer function's denominator by 1, and is the same as dividing the transfer function by s. By passing this modified transfer function to the `step` command, the step response generated by MATLAB will actually be the ramp response for the original system. This response is plotted for the interval $[0, 10]$s.

The proof for this relation is as follows

$$G(s) = \frac{C(s)}{R(s)} \Rightarrow C(s) = G(s) \times R(s)$$
$$\Rightarrow C(s) = \frac{30}{s^2 + 5s + 30} \times R(s)$$

For the system's step response, the MATLAB `step` command evaluates this expression by using $R(s) = \frac{1}{s}$ and plotting the time domain equivalent for the resulting expression.

As the ramp signal $tu(t)$ is equivalent to $\frac{1}{s^2}$, the ramp response for the signal is therefore

$$C_{ramp}(s) = \frac{30}{s^2 + 5s + 30} \times \frac{1}{s^2} = \boldsymbol{\frac{30}{s^4 + 5s^3 + 30s}}$$

Converting this expression to the time domain will yield the ramp response for the system.

By dividing the original transfer function by s

$$G_{modified}(s) = \frac{G(s)}{s} = \frac{30}{s^3 + 5s^2 + 30s}$$

And passing it to the step command

$$C_{modified}(s) = G_{modified}(s) \times \frac{1}{s}$$

$$\Rightarrow C_{modified}(s) = \frac{30}{s^3 + 5s^2 + 30s} \times \frac{1}{s}$$

$$\Rightarrow \boldsymbol{C_{modified(s)} = \frac{30}{s^4 + 5s^3 + 30s^2}}$$

Thus,

$$\boldsymbol{C_{modified}(s) = C_{Ramp}(s)}$$

Which proves that converting the actual s domain expression for a ramp response is equivalent to dividing the original transfer function by s and evaluating its step response using MATLAB. The benefit of the latter approach is that the built-in step function can be used to

- perform the necessary adjustments in the orders of the denominator's polynomial.
- convert the resulting expression into the time domain, and
- plot the resulting response

without having to write custom code for them.

Program Output

Figures 1 and 2 show the plots generated by Code 01. Figure 1 shows the system's step response while Figure 2 shows the ramp response obtained by passing a modified version of the transfer function to the `step` command. Figure 2 also plots the ramp input signal along with the response to better illustrate how the system's response follows its input.

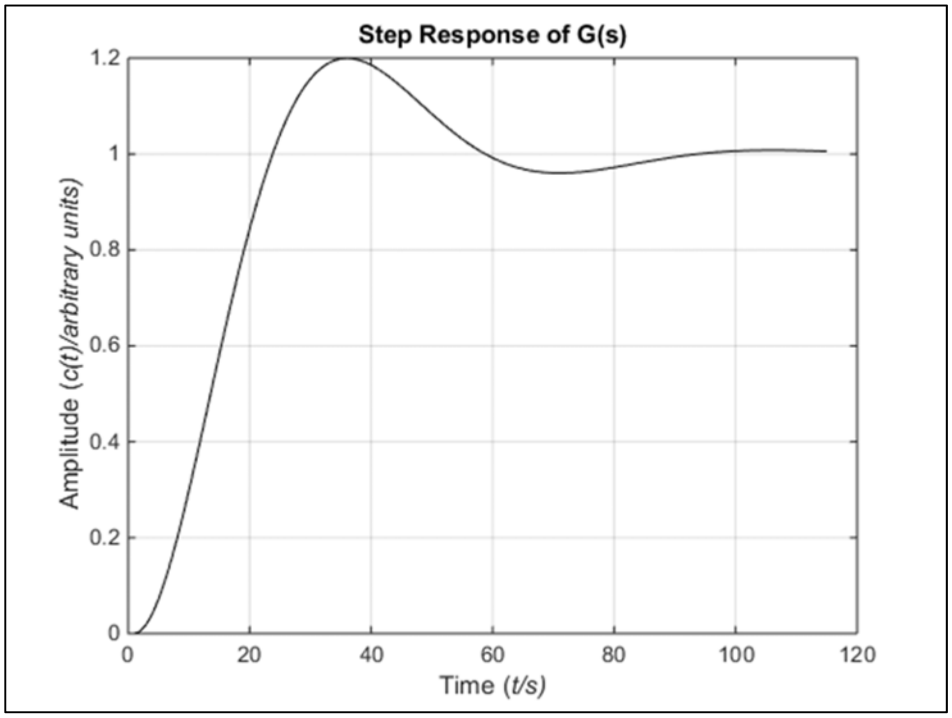

Figure 1: Step Response of System

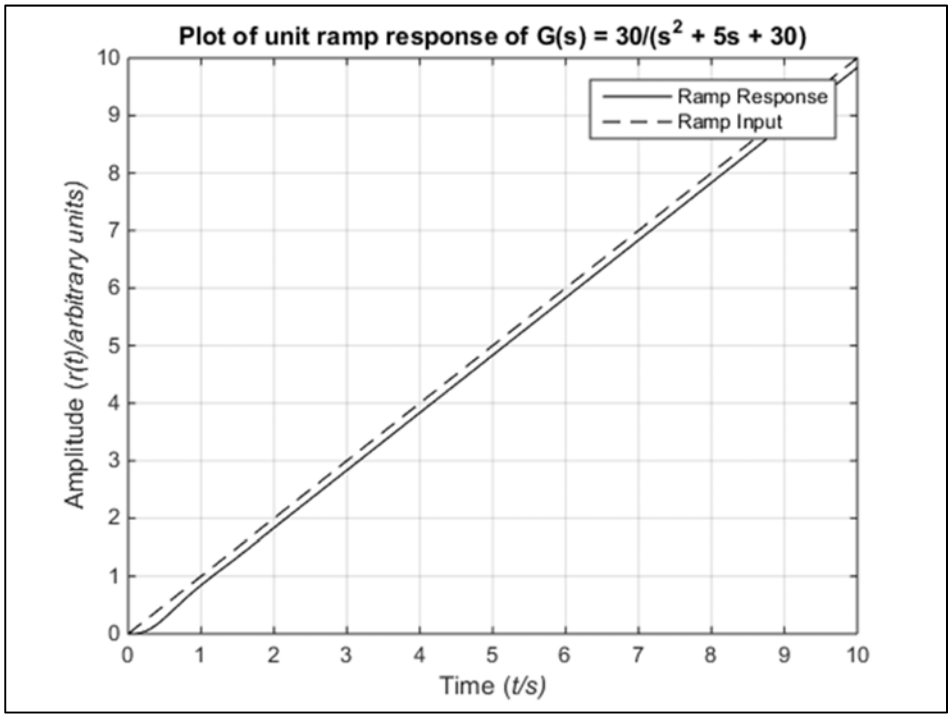

Figure 2: Ramp Response of System compared with Ramp Input

Analysis of Results

Step Response

The step response is characteristic of an underdamped system - the response signal rises gradually to a peak value before dropping below its mean/steady state value. At around 100s, the system's response has attained its steady state value. The unit ramp signal would have instantaneously risen to 1 at t = 0s, which shows that while the system's response is accurate, it is not fast and has a significant rise and delay time.

Ramp Response

Unlike the step response, the ramp response does not stabilize at a constant value. This is because as $t \to \infty$, the ramp input signal also $\to \infty$, and so the system has both an unbounded input and, consequently, unbounded output.

There is a slight delay between the two signals: the ramp input signal begins increasing at t = 0s but the response signal only begins to increase at ~0.3s. This is the ramp response's equivalent of the step response rise time - the system takes some time to produce an output signal in response to an input signal. Also, even though the system's input is linear, there is a non-linear response region that lasts for approximately 0.2s near the beginning of the response. This is likely a transient response that eventually dissipates, as for the majority of the 10s observation period, the system response is a straight line whose value increases with time.

The slight offset/difference between the ramp input and output signal values is likely a consequence of the delay between the two signals due to the system's rise time.

Exercise 02 – Arbitrary Input Response

Obtain the response of the system defined in Example 2 of the manual for an input $\boldsymbol{r(t) = sin(t) + e^{-0.2t}}$ from t = 0s to 15s in steps of 0.001s. Comment on the results.

Solution

Code 02 is a MATLAB program that is uses to investigate the response of a system with the transfer function

$$\boldsymbol{G(s) = \frac{s+5}{s^3 + 2s^2 + 3s + 5}}$$

for an arbitrary input signal $r(t) = \sin(t) + e^{-0.2t}$ for $t \in [0, 15]$s.

Code 02: Step Response of a Second Order System

```
%%  FCS Lab Session 09 Task 02 - Arbitrary Input Signal Response
%   Saad Mashkoor Siddiqui, EE-16163, Section D, TE-EE 16-17

%%  Prepare workspace
clear all; close all; clc;

%%  Defining the transfer function for the plant
num_plant = [1 5];
denom_plant = [1 2 3 5];
plant_tf = tf(num_plant, denom_plant)
t = 0 : 0.001 : 15;

%%  Defining input signal
input_signal = sin(t) + exp(-0.2 * t);

%%  Getting plant response for input signal
plant_response = lsim(num_plant, denom_plant, input_signal, t);

%%  Plotting Response with input signal
figure();
plot(t, input_signal, 'k--', t, plant_response, 'k');
grid; legend([{'Input r(t) = sin(t) + e^{-0.2t}'}, {'System Output'}]);
xlabel('Time (\it{t/s})'); ylabel('Signal (\it{f(t)/arbitrary units})');
title('System Response for r(t) = sin(t) + e^{-0.2t}');
```

Program Explanation

The program first defines the numerator and denominator for the transfer function and then creates a time domain array spanning $[0, 15]$s with values or observation points at every 0.001 s. The program then defines an arbitrary input signal $r(t) = \sin(t) + e^{-0.2t}$ for this time domain and it to the `lsim` command along with transfer function's numerator, denominator, and time domain. The `lsim` command returns the response of a system for an arbitrary input signal. The response data is then plotted on a figure along with the input signal.

Output and Analysis

Figure 03 shows the plot generated by Code 02. The input signal is a sinusoid of amplitude 1 with a DC offset that decays exponentially. By the end of the 15s observation period, the transient input signal has decayed almost entirely, which means the only input signal left is a sinusoid.

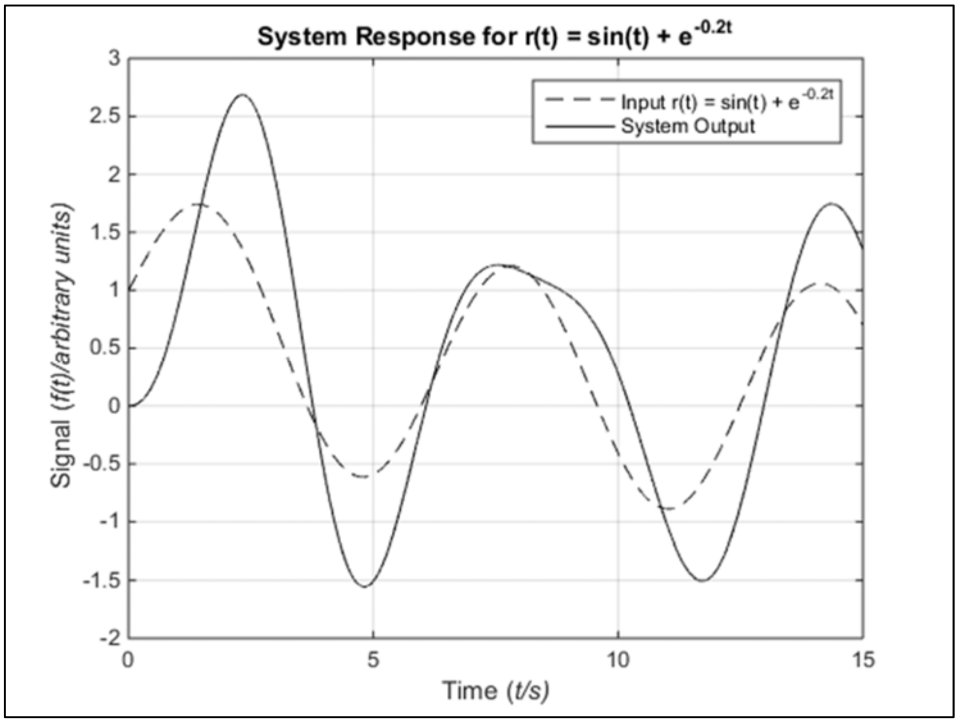

Figure 3: System Response over 15s

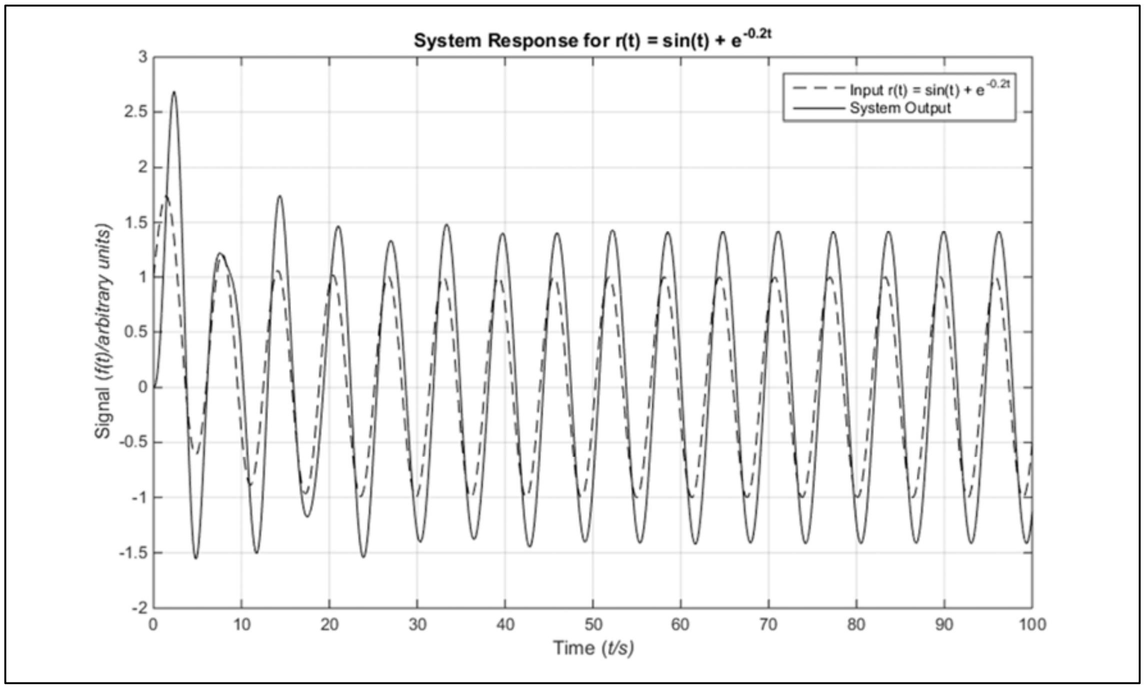

Figure 4: System Response over 30s

Delay between I/O

The system's output signal/response to this arbitrary input shows a more interesting trend.. The initial value of the input signal is 1 since $r(0) = \sin(0) + e^{-0.2(0)} = 0 + 1 = 1$. However, the system's response cannot change instantaneously from its resting value of 0 to a non-zero value, which is why there is a clear delay between the system's input and output signal peaks.

Overshoot

The output signal also seems to overshoot the input signal by at least one 1 arbitrary unit in terms of amplitude, although the overshoot tends to decrease as the exponential component of the input signal decays. At ~7.5s, the signal's output and input signals have roughly the same value but the output signal soon returns to overshooting the input in later cycles of the input signal.

Figure 04 visualizes the same response over 100s instead of just 15s to better gauge the system's steady state response. It is clear from the figure that the system's output amplitude is consistently higher than that of its input, indicating that the system has a gain > 1. Furthermore, the initial delay between the input and output signal persists in the steady state, although it becomes substantially lower.

Sinusoid I/O

Throughout the observation period, the system's input signal is essentially a sinusoid (with an exponentially decaying DC offset). This is why the system's output signal is also a sinusoid, indicating that the system itself represents a simple scaling transformation and not a more complicated non-linear transformation.