**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Ms. Neelma Naz  **Dated:** 04/02/2023

**Semester**: 6th **Section:** BEE 12C

**EE-371:** **Linear Control Systems**

Lab 7: Performance of Systems

Lab Instructor: Yasir Rizwan

Group Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. No** | **Lab Report Marks** | **Viva Marks** | **Total** |
|  |  | **10 Marks** | **5 Marks** | **15 Marks** |
| Danial Ahmad | 331388 |  |  |  |
| Muhammad Umer | 345834 |  |  |  |
| Tariq Umar | 334943 |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[2 Model Verification 3](#_Toc129990438)

[2.1 Objectives 3](#_Toc129990439)

[2.2 Introduction 3](#_Toc129990440)

[2.3 Software 3](#_Toc129990441)

[3 Lab Procedure 4](#_Toc129990442)

[3.1 Exercise 1 4](#_Toc129990443)

[3.2 Exercise 2 5](#_Toc129990444)

[3.3 Exercise 3 6](#_Toc129990445)

[3.4 Exercise 4 8](#_Toc129990446)

[3.5 Exercise 5 10](#_Toc129990447)

[3.6 Exercise 6 10](#_Toc129990448)

[3.7 Exercise 7 & 8 11](#_Toc129990449)

[3.8 Exercise 9 11](#_Toc129990450)

[3.9 Exercise 10 13](#_Toc129990451)

[3.10 Exercise 11 13](#_Toc129990452)

[3.11 Exercise 12 15](#_Toc129990453)

[3.12 Exercise 13 17](#_Toc129990454)

[4 Conclusion 19](#_Toc129990455)

# Model Verification

## Objectives

The objectives of this lab are:

* Learn how to compute the transient and steady state characteristics of a system in MATLAB.

## Introduction

The purpose of this lab report is to learn how to compute the transient and steady state characteristics of a system in MATLAB. Transient and steady state characteristics are important for analyzing the performance and stability of a system under different inputs and conditions. We will then calculate and plot the transient and steady state characteristics such as rise time, settling time, overshoot, peak time, steady state error, etc.

MATLAB is a powerful tool that can help us simulate and visualize the system's response to various inputs and parameters. In this report, we will use MATLAB to model a second-order system with different damping ratios and natural frequencies, and then apply step inputs to observe the transient and steady state behavior of the system.

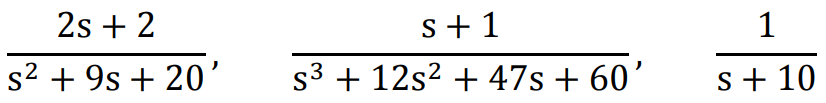
## Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data.

# Lab Procedure

## Exercise 1

Find the rise time, peak time, peak value, overshoot, settling time and the steady state error for step input of the following systems.



tf\_a = tf([2 2], [1 9 20]);

tf\_b = tf([1 1], [1 2 47 60]);

tf\_c = tf(1, [1 10]);

step\_a = stepinfo(tf\_a) %#ok<\*NOPTS>

step\_b = stepinfo(tf\_b)

step\_c = stepinfo(tf\_c)

**Output**

step\_a =

         RiseTime: 0.0510

    TransientTime: 1.5940

     SettlingTime: 1.5940

      SettlingMin: 0.0917

      SettlingMax: 0.1949

        Overshoot: 94.9219

       Undershoot: 0

             Peak: 0.1949

         PeakTime: 0.2878

step\_b =

         RiseTime: 0.1360

    TransientTime: 11.1598

     SettlingTime: 11.6485

      SettlingMin: 0.0025

      SettlingMax: 0.0378

        Overshoot: 126.5680

       Undershoot: 0

             Peak: 0.0378

         PeakTime: 0.4627

step\_c =

         RiseTime: 0.2197

    TransientTime: 0.3912

     SettlingTime: 0.3912

      SettlingMin: 0.0905

      SettlingMax: 0.1000

        Overshoot: 0

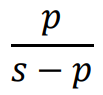
       Undershoot: 0

             Peak: 0.1000

         PeakTime: 1.0546

## Exercise 2

Consider the systems of the following form:



This system has a pole at , it has no zeros, and the gain is equal to the negative of the pole i.e., . Using MATLAB, plot the step response of systems of this form for . Plot all the step responses on a single figure. For each system also find the values of the various performance characteristics (rise time, overshoot, steady state error, etc.). Comment on how the pole of a first order system affects the step response of the system.

sys\_a = zpk([], -1, 1);

step(ss(sys\_a));

sys\_b = zpk([], -2, 2);

hold on

step(ss(sys\_b));

sys\_c = zpk([], -5, 5);

step(ss(sys\_c));

sys\_d = zpk([], -10, 10);

step(ss(sys\_d));

legend('p=-1', 'p=-2', 'p=-5', 'p=-10');

grid

**Output**

step\_a =

         RiseTime: 2.1970

    TransientTime: 3.9121

     SettlingTime: 3.9121

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

         PeakTime: 10.5458

step\_b =

         RiseTime: 1.0985

    TransientTime: 1.9560

     SettlingTime: 1.9560

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

         PeakTime: 5.2729

step\_c =

         RiseTime: 0.4394

    TransientTime: 0.7824

     SettlingTime: 0.7824

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

         PeakTime: 2.1092

step\_d =

         RiseTime: 0.2197

    TransientTime: 0.3912

     SettlingTime: 0.3912

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

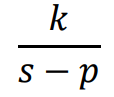
         PeakTime: 1.0546



The addition of an extra pole decreases the speed of the system’s response, with the decrease strength depending on how close the pole is to the axis.

## Exercise 3

Now fix the pole to a constant value and let’s see the effect of changing the gain ‘k’.



Using MATLAB, plot the step response of systems of this form for = and Plot all the step responses on a single figure. For each system also find the values of the various performance characteristics. Comment on the effects of changing the gain.

sys\_a = zpk([], -5, 1);

step(ss(sys\_a));

sys\_b = zpk([], -5, 2);

hold on

step(ss(sys\_b));

sys\_c = zpk([], -5, 5);

step(ss(sys\_c));

sys\_d = zpk([], -5, 10);

step(ss(sys\_d));

legend('k=1', 'k=2', 'k=5', 'k=10');

grid

**Output**

step\_a =

        RiseTime: 0.4394

    TransientTime: 0.7824

     SettlingTime: 0.7824

      SettlingMin: 0.1809

      SettlingMax: 0.2000

        Overshoot: 0

       Undershoot: 0

             Peak: 0.2000

         PeakTime: 2.1092

step\_b =

        RiseTime: 0.4394

    TransientTime: 0.7824

     SettlingTime: 0.7824

      SettlingMin: 0.3618

      SettlingMax: 0.4000

        Overshoot: 0

       Undershoot: 0

             Peak: 0.4000

         PeakTime: 2.1092

step\_c =

       RiseTime: 0.4394

    TransientTime: 0.7824

     SettlingTime: 0.7824

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

         PeakTime: 2.1092

step\_c =

         RiseTime: 0.4394

    TransientTime: 0.7824

     SettlingTime: 0.7824

      SettlingMin: 1.8090

      SettlingMax: 1.9999

        Overshoot: 0

       Undershoot: 0

             Peak: 1.9999

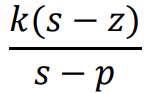
         PeakTime: 2.1092



The change of gain does not contribute to the overall speed of the system’s response, but rather, changes the final value at which the system settles.

## Exercise 4

Now we will introduce a zero and see the effect of changing it. Consider the system:



Using MATLAB, plot the step response of systems of this form for , and . Also have a plot for no zero. Plot all the step responses on a single figure. For each system also find the values of the various performance characteristic. Comment on the effects of changing the zeros.

sys\_a = zpk(-1, -5, 1);

step(ss(sys\_a));

sys\_b = zpk(-2, -5, 1);

hold on

step(ss(sys\_b));

sys\_c = zpk(-5, -5, 1);

step(ss(sys\_c));

sys\_d = zpk(-10, -5, 1);

step(ss(sys\_d));

legend('z=-1', 'z=-2', 'z=-5', 'z=-10');

grid

**Output**

step\_a =

        RiseTime: 0

    TransientTime: 0.7824

     SettlingTime: 1.0597

      SettlingMin: 0.2000

      SettlingMax: 1

        Overshoot: 400.0000

       Undershoot: 0

             Peak: 1

         PeakTime: 0

step\_b =

        RiseTime: 0

    TransientTime: 0.7824

     SettlingTime: 0.8635

      SettlingMin: 0.4000

      SettlingMax: 1

        Overshoot: 150

       Undershoot: 0

             Peak: 1

         PeakTime: 0

step\_c =

        RiseTime: 0

    TransientTime: 0

     SettlingTime: 0

      SettlingMin: 1

      SettlingMax: 1

        Overshoot: 0

       Undershoot: 0

             Peak: 1

         PeakTime: 0

step\_c =

        RiseTime: 0.3219

    TransientTime: 0.7824

     SettlingTime: 0.6438

      SettlingMin: 1.8005

      SettlingMax: 2.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 2.0000

         PeakTime: 2.1092



The addition of an extra zero increases the speed of the system’s response, with the increase strength depending on how close the pole is to the axis.

## Exercise 5

Use the formulas given above to find the values of the pole of a first order system that would give:

* rise times of 0.1, 0.5 and 1
* settling times of 1, 1.5 and 2

T\_r = [0.1, 0.5, 1];

T\_s = [1, 1.5 2];

p\_a =- (2.2) ./ T\_r;

p\_b =- (4) ./ T\_s;

**Output**

T\_r =

    0.1000    0.5000    1.0000

T\_s =

    1.0000    1.5000    2.0000

p\_a =

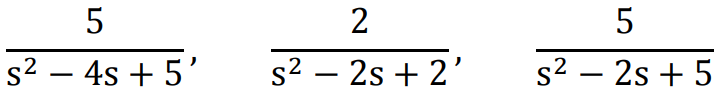
  -22.0000   -4.4000   -2.2000

p\_b =

   -4.0000   -2.6667   -2.0000

## Exercise 6

Find the damping ratio and the natural frequency of the following systems:



tf\_a = tf(5, [1 -4 5]);

tf\_b = tf(2, [1 -2 2]);

tf\_c = tf(5, [1 -2 5]);

disp("damp\_a: "); damp(tf\_a)

disp(newline + "damp\_b: "); damp(tf\_b)

disp(newline + "damp\_c: "); damp(tf\_c)

**Output**

damp\_a:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

  2.00e+00 + 1.00e+00i    -8.94e-01       2.24e+00        -5.00e-01

  2.00e+00 - 1.00e+00i    -8.94e-01       2.24e+00        -5.00e-01

damp\_b:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

  1.00e+00 + 1.00e+00i    -7.07e-01       1.41e+00        -1.00e+00

  1.00e+00 - 1.00e+00i    -7.07e-01       1.41e+00        -1.00e+00

damp\_c:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

  1.00e+00 + 2.00e+00i    -4.47e-01       2.24e+00        -1.00e+00

  1.00e+00 - 2.00e+00i    -4.47e-01       2.24e+00        -1.00e+00

## Exercise 7 & 8

Write a MATLAB function that takes the damping ratio and natural frequency as arguments and returns a transfer function of the form given in equation (2). Call this function my\_second\_order\_tf.

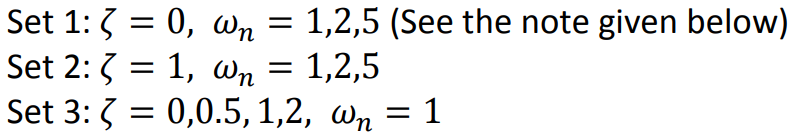
function *tf\_ret* = my\_second\_order\_tf(*zeta*, *w\_n*)

    tf\_ret = tf(w\_n ^ 2, [1, 2 \* zeta \* w\_n, w\_n ^ 2]);

end

## Exercise 9

Using the function that you have just created, my\_second\_order\_tf, make transfer functions for the following sets of damping ratios and natural frequencies:



For each set of values plot the step responses on a single figure. For each system also find the values of the various performance characteristic. Comment on the effects of changing the natural frequency and the damping ratio. Classify each of the above systems as undamped, underdamped, critically damped or overdamped.

% Set 1

tf\_a = my\_second\_order\_tf(0, 1);

tf\_b = my\_second\_order\_tf(0, 2);

tf\_c = my\_second\_order\_tf(0, 5);

% Set 2

tf\_d = my\_second\_order\_tf(1, 1);

tf\_e = my\_second\_order\_tf(1, 2);

tf\_f = my\_second\_order\_tf(1, 5);

% Set 3

tf\_g = my\_second\_order\_tf(0, 1);

tf\_h = my\_second\_order\_tf(0.5, 1)

tf\_i = my\_second\_order\_tf(1, 1);

tf\_j = my\_second\_order\_tf(2, 1);

figure

t = 0:0.01:5;

hold on

step(tf\_a, t); step(tf\_b, t); step(tf\_c, t);

legend('Set 1: 1', 'Set 1: 2', 'Set 1: 3')

grid

figure

hold on

step(tf\_d, t); step(tf\_e, t); step(tf\_f, t);

legend('Set 2: 1', 'Set 2: 2', 'Set 2: 3')

grid

figure

hold on

step(tf\_g, t); step(tf\_h, t);

step(tf\_i, t); step(tf\_j, t);

legend('Set 3: 1', 'Set 3: 2', 'Set 3: 3', 'Set 3: 4')

grid



**Set 1:** Undamped Systems; The speed of the system’s response increases directly proportional to the change in natural frequency.

**Set 2:** Critically Damped Systems; The speed of the system’s response increases directly proportional to the change in natural frequency.

**Set 3:** As natural frequency remains constant, there is no change in system’s response speed, however, damping ratios:

## Exercise 10

Using the formulas given above, find the values of damping ratio and natural frequency that result in and .

syms T\_s T\_p p\_OS zeta w\_n;

eq1 = T\_s == 4/(zeta\*w\_n);

eq2 = T\_p == pi/(w\_n\*sqrt(1-zeta^2));

eq3 = p\_OS == 100\*exp(-(zeta\*pi/sqrt(1-zeta^2)));

eq4 = zeta == -log(p\_OS/100)/(sqrt(pi^2+(log(p\_OS/100))^2));

eq4 = subs(eq4, p\_OS, 10);

zeta\_ans = double(solve(eq4, zeta));

eq1 = subs(eq1, {T\_s, zeta}, [1, zeta\_ans]);

w\_n\_ans = double(solve(eq1, w\_n));

**Output**

zeta\_ans =

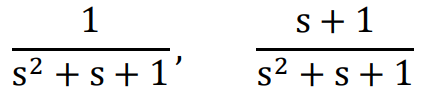
    0.5912

w\_n\_ans =

    6.7664

## Exercise 11

For the systems given below, find the natural frequency, the damping ratio, and transient characteristics. Also plot their step responses on a single graph:



Comment on your observations.

tf\_a = tf(1, [1 1 1]);

tf\_b = tf([1 1], [1 1 1]);

figure

t = 0:0.01:20;

hold on

step(tf\_a, t)

step(tf\_b, t)

grid

legend('tf\_a', 'tf\_b')

The addition of extra zero to the 2nd order system causes the speed of the step response to be faster than the original system.



damp\_a:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -5.00e-01 + 8.66e-01i     5.00e-01       1.00e+00         2.00e+00

 -5.00e-01 - 8.66e-01i     5.00e-01       1.00e+00         2.00e+00

damp\_b:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -5.00e-01 + 8.66e-01i     5.00e-01       1.00e+00         2.00e+00

 -5.00e-01 - 8.66e-01i     5.00e-01       1.00e+00         2.00e+00

step\_a:

        RiseTime: 1.6390

    TransientTime: 8.0759

     SettlingTime: 8.0759

      SettlingMin: 0.9315

      SettlingMax: 1.1629

        Overshoot: 16.2929

       Undershoot: 0

             Peak: 1.1629

         PeakTime: 3.5920

step\_b:

         RiseTime: 0.9409

    TransientTime: 7.5054

     SettlingTime: 7.5054

      SettlingMin: 0.9403

      SettlingMax: 1.2984

        Overshoot: 29.8352

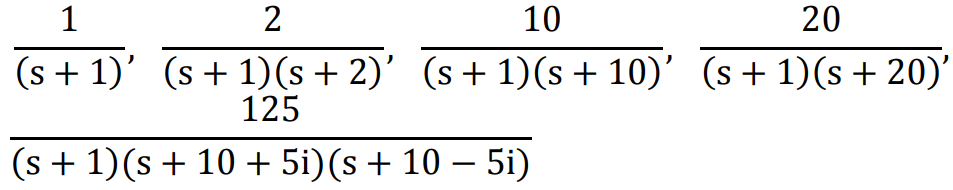
       Undershoot: 0

             Peak: 1.2984

         PeakTime: 2.3947

## Exercise 12

For the systems given below, find the natural frequency, the damping ratio, and transient characteristics. Also plot their step responses on a single graph.



Comment on your observations.

tf\_a = zpk([], -1, 1);

tf\_b = zpk([], [-1 -2], 2);

tf\_c = zpk([], [-1 -10], 10);

tf\_d = zpk([], [-1 -20], 20);

tf\_e = zpk([], [-1 -10-5\*1i -10+5\*1i], 125);

figure

t = 0:0.01:8;

hold on

step(tf\_a, t)

step(tf\_b, t)

step(tf\_c, t)

step(tf\_d, t)

step(tf\_e, t)

grid

legend('tf\_a', 'tf\_b', 'tf\_c', 'tf\_d', 'tf\_e')

The addition of extra pole to the 1st order system causes the speed of the step response to be slower than the original system.



damp\_a:

   Pole        Damping       Frequency      Time Constant

                           (rad/seconds)      (seconds)

 -1.00e+00     1.00e+00       1.00e+00         1.00e+00

damp\_b:

   Pole        Damping       Frequency      Time Constant

                           (rad/seconds)      (seconds)

 -1.00e+00     1.00e+00       1.00e+00         1.00e+00

 -2.00e+00     1.00e+00       2.00e+00         5.00e-01

damp\_c:

   Pole        Damping       Frequency      Time Constant

                           (rad/seconds)      (seconds)

 -1.00e+00     1.00e+00       1.00e+00         1.00e+00

 -1.00e+01     1.00e+00       1.00e+01         1.00e-01

damp\_d:

   Pole        Damping       Frequency      Time Constant

                           (rad/seconds)      (seconds)

 -1.00e+00     1.00e+00       1.00e+00         1.00e+00

 -2.00e+01     1.00e+00       2.00e+01         5.00e-02

damp\_e:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -1.00e+00                 1.00e+00       1.00e+00         1.00e+00

 -1.00e+01 - 5.00e+00i     8.94e-01       1.12e+01         1.00e-01

 -1.00e+01 + 5.00e+00i     8.94e-01       1.12e+01         1.00e-01

step\_a:

        RiseTime: 2.1970

    TransientTime: 3.9121

     SettlingTime: 3.9121

      SettlingMin: 0.9045

      SettlingMax: 1.0000

        Overshoot: 0

       Undershoot: 0

             Peak: 1.0000

         PeakTime: 10.5458

step\_b:

         RiseTime: 2.5901

    TransientTime: 4.6002

     SettlingTime: 4.6002

      SettlingMin: 0.9023

      SettlingMax: 0.9992

        Overshoot: 0

       Undershoot: 0

             Peak: 0.9992

         PeakTime: 7.7827

step\_c:

         RiseTime: 2.2150

    TransientTime: 4.0174

     SettlingTime: 4.0174

      SettlingMin: 0.9005

      SettlingMax: 0.9993

        Overshoot: 0

       Undershoot: 0

             Peak: 0.9993

         PeakTime: 7.3591

step\_d:

         RiseTime: 2.2000

    TransientTime: 3.9634

     SettlingTime: 3.9634

      SettlingMin: 0.9040

      SettlingMax: 0.9993

        Overshoot: 0

       Undershoot: 0

             Peak: 0.9993

         PeakTime: 7.3222

step\_e:

         RiseTime: 2.2117

    TransientTime: 4.0769

     SettlingTime: 4.0769

      SettlingMin: 0.9001

      SettlingMax: 0.9994

        Overshoot: 0

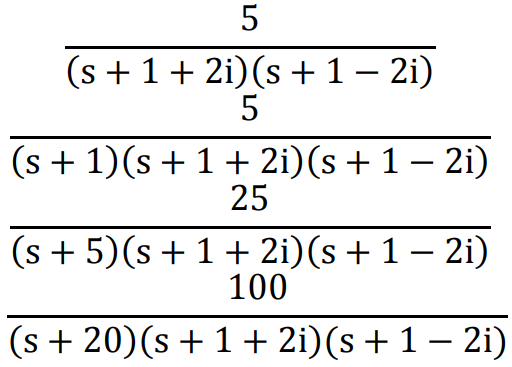
       Undershoot: 0

             Peak: 0.9994

         PeakTime: 7.5433

## Exercise 13

For the systems given below, find the natural frequency, the damping ratio, and transient characteristics. Also plot their step responses on a single graph:



Comment on your observations.

tf\_a = zpk([], [-1-2\*1i -1+2\*1i], 5);

tf\_b = zpk([], [-1 -1-2\*1i -1+2\*1i], 5);

tf\_c = zpk([], [-5 -1-2\*1i -1+2\*1i], 25);

tf\_d = zpk([], [-20 -1-2\*1i -1+2\*1i], 100);

figure

t = 0:0.01:8;

hold on

step(tf\_a, t)

step(tf\_b, t)

step(tf\_c, t)

step(tf\_d, t)

grid

legend('tf\_a', 'tf\_b', 'tf\_c', 'tf\_d')

The addition of extra pole to the 2nd order system causes the speed of the step response to be slower than the original system.



damp\_a:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -1.00e+00 - 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -1.00e+00 + 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

damp\_b:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -1.00e+00                 1.00e+00       1.00e+00         1.00e+00

 -1.00e+00 - 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -1.00e+00 + 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

damp\_c:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -1.00e+00 - 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -1.00e+00 + 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -5.00e+00                 1.00e+00       5.00e+00         2.00e-01

damp\_d:

         Pole              Damping       Frequency      Time Constant

                                       (rad/seconds)      (seconds)

 -1.00e+00 - 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -1.00e+00 + 2.00e+00i     4.47e-01       2.24e+00         1.00e+00

 -2.00e+01                 1.00e+00       2.00e+01         5.00e-02

step\_a:

         RiseTime: 0.6903

    TransientTime: 3.7352

     SettlingTime: 3.7352

      SettlingMin: 0.9149

      SettlingMax: 1.2079

        Overshoot: 20.7866

       Undershoot: 0

             Peak: 1.2079

         PeakTime: 1.5658

step\_b:

         RiseTime: 1.5889

    TransientTime: 4.4520

     SettlingTime: 4.4520

      SettlingMin: 0.9076

      SettlingMax: 0.9993

        Overshoot: 0

       Undershoot: 0

             Peak: 0.9993

         PeakTime: 7.7827

step\_c:

         RiseTime: 0.7733

    TransientTime: 3.9210

     SettlingTime: 3.9210

      SettlingMin: 0.9145

      SettlingMax: 1.1843

        Overshoot: 18.4293

       Undershoot: 0

             Peak: 1.1843

         PeakTime: 1.8052

step\_d:

         RiseTime: 0.6967

    TransientTime: 3.7853

     SettlingTime: 3.7853

      SettlingMin: 0.9094

      SettlingMax: 1.2064

        Overshoot: 20.6433

       Undershoot: 0

             Peak: 1.2064

         PeakTime: 1.6118

# Conclusion

The purpose of this lab report was to learn how to compute the transient and steady state characteristics of a system in MATLAB. Transient and steady state characteristics are important for analyzing the performance and stability of a system under different inputs and conditions. MATLAB is a powerful tool for simulating and visualizing the behavior of systems using numerical methods and graphical tools. In this report, we used MATLAB to model a first-order system and a second-order system with different parameters and inputs. We then calculated and plotted the transient and steady state characteristics such as rise time, settling time, overshoot, peak time, steady state error, etc. We also compared the results with the theoretical values obtained from analytical solutions.