ECE 3413 Lab 01
Introduction to Matlab and Simulink

Leomar Durán

1st February 2023

1 Introduction

The purpose of this experiment is to serve as a practical introduction to Matlab and Simulink and their graphical user interface.

Matlab is a linear algebra tool, and Simulink is a tool for modeling systems using block diagrams. They may each be used for modeling control systems among other types of systems.

In this example, we use Matlab to model a polynomial as a row vector of coefficients. We also use Matlab in conjunction with Simulink to prepare, model and contrast 2 control systems.

2 Procedure

2.1 Roots and corresponding phase angles of a polynomial

In this part of the lab, we model a polynomial in Matlab. This is done by using a row vector. Of the operations that can be performed on a row vector, 2 are the `conv` and `roots` functions.

2.1.1 Convolution

The `conv` function is used to convolve two vectors representing polynomial multiplication.

For example, we have the polynomials

$$P_1(s) = s^2 + 10s + 24,$$
$$P_2(s) = s^4 + 26s^3 + 231s^2 + 766s + 560.$$

These may be represented by the row vectors

$$\vec{v_1} := \begin{bmatrix} 1 & 10 & 24 \end{bmatrix},$$
$$\vec{v_2} := \begin{bmatrix} 1 & 26 & 231 & 766 & 560 \end{bmatrix}.$$

Then the convolution $\mathbf{v}_1 * \mathbf{v}_2$ represents the product $(P_1P_2)(s)$ as follows

```
       [   1  26 231 766 560 ]
[ 24  10   1]                          = 1
    [ 24  10   1]                      = 10+26 = 36
        [ 24  10   1]                  = 24 + 260 + 231 = 515
            [ 24  10   1]              = 624 + 2310 + 766 = 3700
                [ 24  10   1]          = 5544 + 7660 + 560 = 13764
                    [ 24  10   1]      = 18384 + 5600 = 23984
                        [ 24  10   1] = 13440
-------------------------------------------------------------------
[1 10 24]*[1 26 231 766 560] = [1 36 515 3700 13764 23984 13440],
```

representing the polynomial

$$P(s) = s^6 + 36s^5 + 515s^4 + 3700s^3 + 13764s^2 + 23984s + 13440.$$

2.1.2 The roots of the polynomial

The `roots` function in Matlab returns the roots of the polynomial $P(s)$, that is, the values of s s.t. $P(s) = 0$.

2.2 Transfer functions

In part 2 of the lab, we model a transfer function and modify one of its coefficients to produce a new output.

We use Matlab to set up the parameters for the transfer functions before using Simulink to model them because Matlab allows for a cleaner interface to set up the parameters.

Additionally, we can use Matlab to perform sanity checks along the way because it echos the value of every statement that is not suppressed by the semicolon character (`;`).

For example, we can echo the poles, zeros, numerators and denominators of the transfer functions to ensure that we set them up correctly.

Simulink is then used to model the system visually as a block diagram. The transfer functions are compared by their step responses, their response to a step signal. The signal starts at 1 V, then jumpts to 2 V at 1 ms.

2.2.1 Transfer functions in Matlab

Matlab has transfer function objects. These can be represented by objects as a ratio of polynomials using the `tf` function, or as their zeros, poles and gain using `zpk`. In this lab, we make use of the `tf` objects.

2.2.2 Poles

The poles are the roots of the denominator of the transfer function. Matlab has the function `pole`, which accepts a transfer function and returns it poles.

2.2.3 Zeros

The zeros are the roots of the numerator of the transfer function. Matlab has the function `zero`, which accepts a transfer function and returns it zero.

2.2.4 The modified transfer function

We are given the transfer function

$$H(s) = \frac{s^2 + 10s + 24}{s^4 + 26s^3 + 231s^2 + 766s + 560}.$$

I have modified it by negating its pole with the minimum magnitude.

3 Results

3.1 Part 1 – Roots and corresponding phase angles of a polynomial

3.1.1 Given the polynomial

$$P(s) = (s^2 + 10s + 24)(s^4 + 26s^3 + 231s^2 + 766s + 560),$$

```
P = 1x7
           1          36         515        3700       13764 ...
       23984       13440
```

3.1.2 1. we find the roots of the resulting polynomial

```
P_roots = 6x1
  -10.0000
   -8.0000
   -7.0000
   -6.0000
   -4.0000
   -1.0000
```

3.1.3 2. and their corresponding phase angles

	P_roots	P_root_angles_in_deg
1	-10	180
2	-8	180
3	-7	180
4	-6	180
5	-4	180
6	-1	180

This section is performed as a Matlab live script.

The polynomial convolved in Matlab is the same from subsubsection 2.1.1, and we found the expected resulting row vector of coefficients.

As for the roots, we also expected that the roots would have a phase angle of 180°, which means that the roots would be a negative real number in the rectangular coordinate system. The reason we expected these roots is because the polynomial has all positive coefficients. This is one type of stable transfer function.

3.2 Transfer functions

3.2.1 The roots and poles

Given the transfer function

$$H(s) = \frac{s^2 + 10s + 24}{s^4 + 26s^3 + 231s^2 + 766s + 560},$$

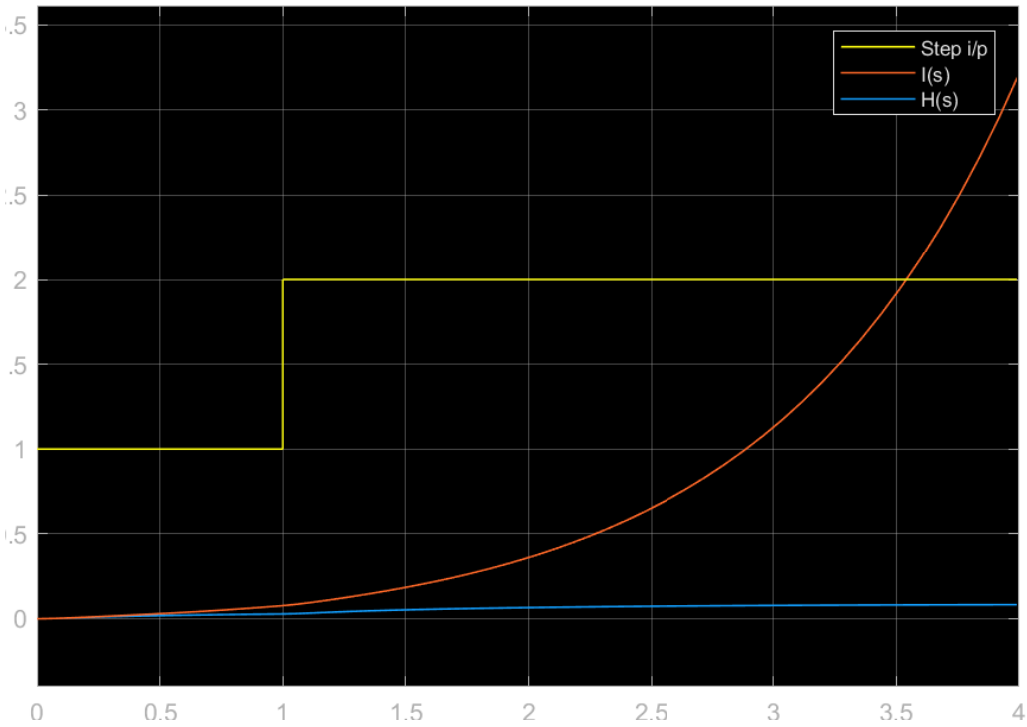

Figure 1: Modeling a simple input-output system, a stable step response and an unstable step response in voltage vs time in milliseconds.

we find that the poles are in

$$\begin{bmatrix} -10.0000 \\ -8.0000 \\ -7.0000 \\ -1.0000 \end{bmatrix}$$

and the zeroes are in

$$\begin{bmatrix} -6.0000 \\ -4.0000 \end{bmatrix}.$$

Further, we modify this transfer function by negating the pole with the least magnitude (-1.0000) producing

$$I(s) = \frac{s^2 + 10s + 24}{s^4 + 24s^3 + 181s^2 + 354s - 560}.$$

The result can be seen in Fig. 1.

The step function jumps at 1 ms. By 3 ms, $H(s)$ has settled at about 1 V. However, $I(s)$ is at 1.129 V and goes beyond 2 V at time 3.543 ms. It continues to grow exponentially. This happens because $H(s)$ is a stable transfer function whereas $I(s)$ is unstable.

4 Discussion

This experiment was straightforward in its procedure. We learned about how Matlab handles multiplication of polynomials, and we also experienced a transfer function responding to a signal in a simulated system.

This experiment introduced the idea of stable and unstable transfer functions. A stable transfer function can be used to attenuate a signal before it gets too far out of bounds. However, an unstable function may not help with this.

Engineers may have more precise formulas to help choose the correct poles and zeros and properly design a transfer function.

A Appendix

A.1 Part 1 – Roots and corresponding phase angles of a polynomial, Matlab Live Script

Part 1 — Roots and corresponding phase angles of a polynomial

```
clear
```

Given the polynomial

$$P(s) = (s^2 + 10s + 24)(s^4 + 26s^3 + 231s^2 + 766s + 560),$$

```
P = conv([1 10 24], [1 26 231 766 560])
```

```
P = 1×7
           1          36         515        3700       13764       23984 ···
```

1. we find the roots of the resulting polynomial

```
P_roots = roots(P)
```

```
P_roots = 6×1
  -10.0000
   -8.0000
   -7.0000
   -6.0000
   -4.0000
   -1.0000
```

2. and their corresponding phase angles

```
P_root_angles_in_deg = rad2deg(angle(P_roots));
tab_P_root_angles = table(P_roots, P_root_angles_in_deg)
```

```
tab_P_root_angles = 6×2 table
```

	P_roots	P_root_angles_in_deg
1	-10	180
2	-8	180
3	-7	180
4	-6	180
5	-4	180
6	-1	180

A.2 Part 2 – Transfer function parameters

```
%% lab01-intro/part02_01_transfer_function_params.m
% Parameters for exploring the effects of transfer functions on a step
% signal.
% By      : Leomar Duran <https://github.com/lduran2>
% When    : 2023-02-01t07:28
% For     : ECE 3412 Classical Control Systems
% Version : 0.1.0

clear

% H(s) = (ss + 10s + 24)/(s^4 + 26s^3 + 231ss + 766s + 560)
Hpoly = [0 0 1 10 24 ; 1 26 231 766 560]; % [V^0]
H = tf(Hpoly(1,:), Hpoly(2,:))

% find the poles and zeros
H_poles = pole(H)
H_zeros = zero(H)

% parameters of step input
step_init = 1;  % [V]
step_final = 2; % [V]
step_time = 1;  % [s]

% copy size of Ipoly
Ipoly = zeros(size(Hpoly));
% find and negate the minimum magnitude pole
[~, i_negate] = min(abs(H_poles))
I_poles = H_poles;
I_poles(i_negate) = -I_poles(i_negate)
% copy Hpoly's numerator, but use new poles
Ipoly = [ Hpoly(1,:) ; poly(I_poles) ]
% show the new transfer function (just for show)
I = tf(Ipoly(1,:), Ipoly(2,:))
```

A.3 Part 2 – Modeling a system with a transfer function in Simulink