MATLAB Experiment II – System Modelling

2.1 Introduction

The MATLAB control and signal processing toolboxes contain many helpful commands for the analysis of control systems. In this laboratory exercise, you will be introduced to the MATLAB functions for obtaining the partial-fraction expansion of a continuous-time control system, $H(s)$, the zeros and poles of $H(s)$ and the corresponding pole-zero diagram (*s*-plane plot).

2.2 Partial Fraction Expansion of H(s)

MATLAB has a command to obtain the partial-fraction expansion of a transfer function *H*(*z*).
Consider the following transfer function:

$$H(s) = \frac{B(s)}{A(s)} = \frac{\text{num}}{\text{den}} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_n}{a_0 s^n + a_1 s^{n-1} + \cdots + a_n},$$

where a_k and b_k are the k^{th} coefficients of the transfer-function polynomials in descending powers of the complex variable *s*. Note that some of these coefficients may be zero. In MATLAB, the row vectors, num and den, specify the coefficients of the numerator and denominator of the transfer function. That is,

$$\text{num} = [b_0 \;\; b_1 \ldots b_n]$$
$$\text{den} = [a_0 \;\; a_1 \ldots a_n]$$

The command

```
[r, p, k] = residue(num, den)
```

finds the residues (or zeros), the poles and direct terms of a partial-fraction expansion of the discrete-time, transfer function, *H*(*z*), and places them in the column vectors r, p and k, respectively. If there are no multiple roots, then the partial-fraction expansion of *H*(*z*) is given by:

$$H(s) = \frac{B(s)}{A(s)} = \frac{r_1}{s - p_1} + \cdots + \frac{r_n}{s - p_n} + k(s).$$

If $p_k = p_{k+1} = \cdots = p_{k+m-1}$, the pole p_k is a pole of multiplicity *m*. In this case, the expansion includes terms of the form:

$$\frac{r_k}{s - p_k} + \frac{r_{k+1}}{(s - p_k)^2} + \cdots + \frac{r_{k+m-1}}{(s - p_k)^m}.$$

Example. Consider the following transfer function

$$H(s) = \frac{B(s)}{A(s)} = \frac{18s^3}{18s^3 + 3s^2 - 4s - 1}.$$

In order to obtain the zeros, poles and direct term for this transfer function, we write the following in MATLAB:

```
>> num = [18 0  0  0];
>> den = [18 3 -4 -1];
>> [r, p, k] = residue(num, den)
```

```
r =
    0.1800
   -0.3467
    0.0444
p =
    0.5000
   -0.3333
   -0.3333
k =
     1
```

This is the MATLAB partial-fraction expansion of the transfer function $H(s)$:

$$H(s) = \frac{18s^3}{18s^3 + 3s^2 - 4s - 1} = \frac{0.18}{s - 0.5} - \frac{0.3467}{s + 0.3333} + \frac{0.0444}{(s + 0.3333)^2} + 1$$

The residue function can also be used to form the transfer-function polynomials (numerator and denominator) from the partial-fraction expansion using the command

```
[num, den] = residue(r, p, k)
```

where r, p and k are the vectors containing the residues, poles and direct terms, respectively. The outputs num and den are vectors containing the numerator and denominator polynomial coefficients, respectively. Therefore, given the previous MATLAB output, the transfer function for the corresponding partial-fraction expansion is obtained in MATLAB as follows:

```
>> [num, den] = residue(r, p, k);
>> printsys(num, den, 's')

num/den =

   s^3 - 4.996e-016 s^2 - 3.8858e-016 s - 8.3267e-017
   --------------------------------------------------
        s^3 + 0.16667 s^2 - 0.22222 s - 0.055556
```

where the command

```
[num, den] = printsys(num, den, 's')
```

prints the num/den in terms of the ratio of polynomials in the variable *s*.

Question: Why does the numerator of transfer function produced by MATLAB differ from $H(s)$ given above?

2.3 The Zeros and Poles of H(s)

One way of obtaining the poles, zeros and gain of a transfer function in MATLAB is through the use of the function:

```
[z, p, k] = tf2zp(num, den).
```

This function returns the vectors z and p containing the poles and zeros, as well as a real number, k, which is the gain factor of the transfer function.

Example. Consider the system defined by:

$$H(s) = \frac{B(s)}{A(s)} = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}.$$

The zeros (z), poles (p) and gain (k), of $H(s)$ can be obtained with the following MATLAB code:

```
>> num = [0 0 4 16 12];
>> den = [1 12 44 48 0];
>> [z, p, K] = tf2zp(num, den)
z =
    -3
    -1
p =
         0
   -6.0000
   -4.0000
   -2.0000
K =
     4
```

The zeros are at $s = -3$ and $s = -1$. The poles are at $s = 0, -6, -4$, and -2. The gain K is 4. If the zeros, poles, and gain K are given, then the following MATLAB program will yield the original num/den. So, given the previous MATLAB outputs, we can obtain the numerator and denominator polynomials of $H(z)$, thus:

```
>> [num, den] = zp2tf(z,p,K);
>> printsys(num, den, 's')
num/den =

         4 s^2 + 16 s + 12
   ----------------------------
   s^4 + 12 s^3 + 44 s^2 + 48 s
```

2.4 Block Diagrams and Step Response

In control systems analysis, we frequently need to simplify a network of interconnected transfer functions and into a single transfer function which is then used in subsequent calculations for analysis purposes. There are three different types of connections between transfer function that are usually encountered in practice: cascade-connected, parallel-connected and feedback-connected (closed-loop) transfer functions. MATALB has convenient commands to obtain these transfer functions. To obtain the transfer functions of the cascaded, parallel, feedback and unity feedback systems, the following commands are used, respectively:

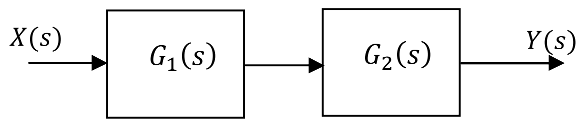
```
[num, den] = series(num1, den1, num2, den2)
[num, den] = parallel(num1, den1, num2, den2)
[num, den] = feedback(num1, den1, num2, den2)
[num, den] = cloop(num1, den1, -1)
```

Example. Suppose the following transfer functions are connected as shown in Figure 1:

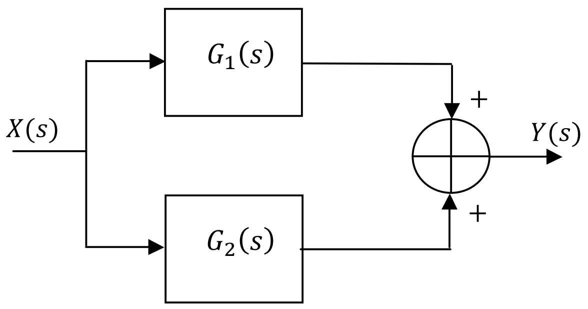
$$G_1(s) = \frac{\text{num1}}{\text{den1}} = \frac{10}{s^2 + 2s + 10}$$

$$G_2(s) = \frac{\text{num2}}{\text{den2}} = \frac{5}{\text{s+5}}$$

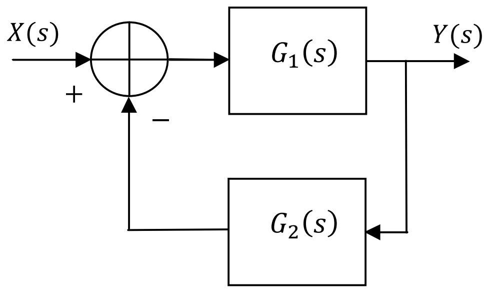
(a)



(b)



(c)



(d)

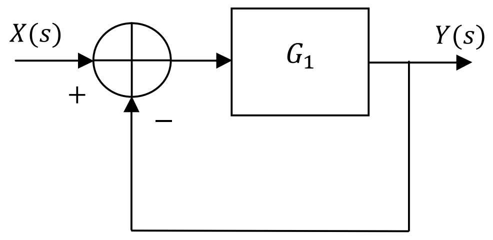


Figure 1: (a) Cascade (series) system, (b) parallel system and (c) feedback system.

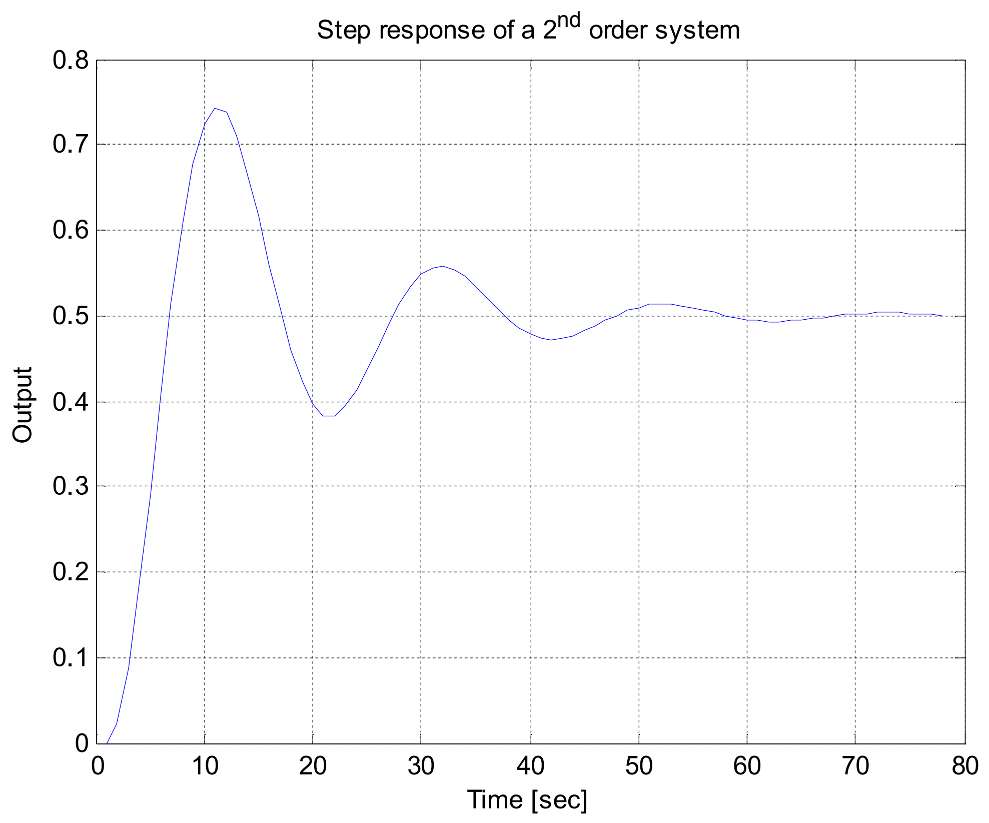
The following program gives the overall transfer function $Y(s)/X(s)$ for each arrangement of $G_1(s)$ and $G_2(s)$.

```
>> num1 = [0 0 10];
>> den1 = [1 2 10];
>> num2 = [0 5];
>> den2 = [1 5];
>> [num, den] = series(num1, den1, num2, den2);
>> printsys(num, den)
num/den =

              50
   -----------------------
   s^3 + 7 s^2 + 20 s + 50

>> [num, den] = parallel(num1, den1, num2, den2);
>> printsys(num, den)
num/den =

      5 s^2 + 20 s + 100
   -----------------------
   s^3 + 7 s^2 + 20 s + 50

>> [num, den] = feedback(num1, den1, num2, den2);
>> printsys(num, den)
num/den =

           10 s + 50
   ------------------------
   s^3 + 7 s^2 + 20 s + 100


>> [num, den] = cloop(num1, den1, -1);
>> printsys(num, den)

num/den =

         10
   --------------
   s^2 + 2 s + 20
```

The `step` function calculates the unit step response of a linear system. The `step` function is very important, since control system performance specifications are often given in terms of the unit step response. For example, the step response of the unity feedback system can be investigated using the following code:

```
[y,x] = step(num,den);
>> figure,plot(y),grid
>> xlabel('Time [sec]'),ylabel('Output')
>> title('Step response of a 2^n^d order system')
```

Use the HTML help provided by MATLAB to gain a more in depth understanding of this function; in particular, look at the other arguments accepted by this function.


Step response of a 2nd order system
Output
Time [sec]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
50
60
70
80