Example 2: Creating Transfer Function and Zero-Pole-Gain Models

Suppose we are given transfer functions written in two forms:

$$G = \frac{2s+1}{3s^2+2s+4}$$

$$H = 7\frac{(s+1)(s+3)}{(s+2)(s+2)(s+4)}$$

The numerator and denominator in G is expanded out and the numbers appearing are the coefficients. On the other hand, in H, the numerator and denominators have been factored out and it includes a multiplier gain (7 in this case). Note, however that the roots have the opposite signs of the numbers shown, e.g. the roots of the numerators are: -1 and -3.

We use the **tf** command to build a model for G,

```
>> G=tf([2,1],[3,2,4])

Transfer function:
    2 s + 1
---------------
3 s^2 + 2 s + 4
```

For *H*, we use the **zpk** command:

```
>> H=zpk([-1,-3],[-2,-2,-4],7)

Zero/pole/gain:
7 (s+1) (s+3)
-------------
 (s+2)^2 (s+4)
```

2. Converting information between different models

There are two ways of converting one group of information to another group of information. In table 2, the various commands are summarized

Table 1.

Command Function	**Converting From**	**Converting To**	**Input Arguments**	**Output Arguments**
ss2tf	state space	transfer function	**[A,B,C,D]**	**[num,den]**
ss2zp		zero pole gain		**[z,p,k]**
tf2ss	transfer function	state space	**[num.den]**	**[A,B,C,D]**
tf2zp		zero pole gain		**[z,p,k]**
zp2ss	zero pole gain	state space	**[z,p,k]**	**[A,B,C,D]**
zp2tf		transfer function		**[num,den]**

Example 3. Using state space information to build a transfer function object

Using the matrices entered earlier from example 1, we use the function **`ss2tf`** to obtain the coefficients of the numerator and denominator needed to create **`F`**, a transfer function object.

```
>> [num,den]=ss2tf(A,B,C,D);
>> F=tf(num,den);
>> F

Transfer function:
0.5 s^2 + 5.5 s - 18
--------------------
   s^2 + 3 s - 10
```

On the other hand, if one just wishes to convert one object to another type of object then the commands **`tf, ss, zpk`** will automatically convert them to the target type.

Example 4. Automatic conversion

Using the state space object, **ht_model**, we can immediately convert this to a transfer function, say **F2**,

```
>> F2=tf(ht_model)

Transfer function:
0.5 s^2 + 5.5 s - 18
--------------------
   s^2 + 3 s - 10
```

which is the same as **F** in example 3.

3. Including delays into the control objects

To include delay to a transfer function, we include two more arguments in the **tf** command. For example, let $Q(s) = e^{-2s} F(s)$, with F given in example 3,

```
>> Q=tf(num,den,'Inputdelay',2);
>> Q

Transfer function:
            0.5 s^2 + 5.5 s - 18
exp(-2*s) * --------------------
               s^2 + 3 s - 10
```

Alternatively, you can set/change the properties by using the **set** command:

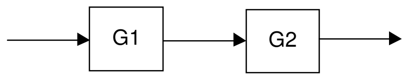
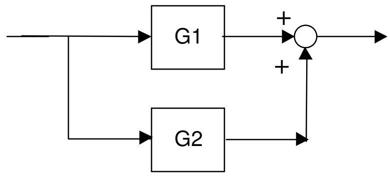
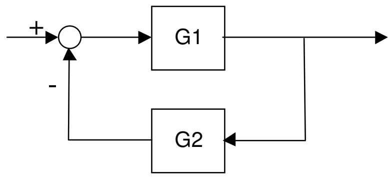
```
>> set(Q,'Inputdelay',0.5)
>> Q

Transfer function:
              0.5 s^2 + 5.5 s - 18
exp(-0.5*s) * --------------------
                 s^2 + 3 s - 10
```

4. Combining different objects

The various control objects can be connected in series, parallel or negative feedback. This is summarized in Table 2.

Table 2.

Command Function	**Combination**
`series(G1,G2)`	
`parallel(G1,G2)`	
`feedback(G1,G2)`	

Example 5. Combination of transfer functions.

```
>> G1=tf([1],[10,1])

Transfer function:
   1
--------
10 s + 1

>> G2=tf([1 2],[2 3 2])

Transfer function:
     s + 2
--------------
2 s^2 + 3 s + 2

>> G3=series(G1,G2)

Transfer function:
          s + 2
--------------------------
20 s^3 + 32 s^2 + 23 s + 2
```

Alternatively, the Matlab control toolbox offers the basic arithmetic operations of control objects, namely addition, multiplication and inversion, as summarized in Table 3.

Table 3.

Operation	Example
addition	`G3=G1+G2`
muliplication	`G3=G1*G2`
inversion	`G3=inv(G1)`

Example 6. Operation of transfer functions.

a) multiplication and series combination are equivalent: (**G1** and **G2** are transfer function objects given in example 5)

```
>> G3=series(G1,G2)

Transfer function:
          s + 2
--------------------------
20 s^3 + 32 s^2 + 23 s + 2

>> G3=G1*G2

Transfer function:
          s + 2
--------------------------
20 s^3 + 32 s^2 + 23 s + 2
```

b) direct calculation of the negative feedback:

```
>> G4=G1*inv(1+G1*G2)

Transfer function:
      20 s^3 + 32 s^2 + 23 s + 2
--------------------------------------
200 s^4 + 340 s^3 + 272 s^2 + 64 s + 4
```

Actually, the result is not the minimal representation. To see this, change the transfer function object to the zero-pole-gain object, and you can observe that there is a zero-pole cancellation of **(s+0.1)** was not performed. The **feedback(G1,G2)** function statement actually reduces the orders automatically (see below).

```
>> G4=zpk(G4)

Zero/pole/gain:
       0.1 (s+0.1) (s^2  + 1.5s + 1)
-------------------------------------------
(s+0.2244) (s+0.1) (s^2  + 1.376s + 0.8913)

>> G3=zpk(feedback(G1,G2))

Zero/pole/gain:
       0.1 (s^2  + 1.5s + 1)
-----------------------------------
(s+0.2244) (s^2  + 1.376s + 0.8913)
```

6. Obtaining plots and responses

Having created the object, figures of different plots and responses can be obtained by using the functions given in Table 4.

Table 4.

Function	Description
`nyquist(G1)`	Nyquist plot
`bode(G1)`	Bode plot
`nichols(G1)`	Nichols plot
`[gm,pm,wcg,wcp]=margin(G1)`	gm=Gain margin Pm=Phase margin
`step(G1)`	Step response
`impulse(G1)`	Impulse response

Example 6. Nyquist and Bode Plots.

```
>> G2=zpk([-10],[-0.01],1)

Zero/pole/gain:
 (s+10)
--------
(s+0.01)
```

For Nyquist plot,

```
>> nyquist(G2)
>> axis equal
```

For Bode plot,

```
>> bode(G2)
```

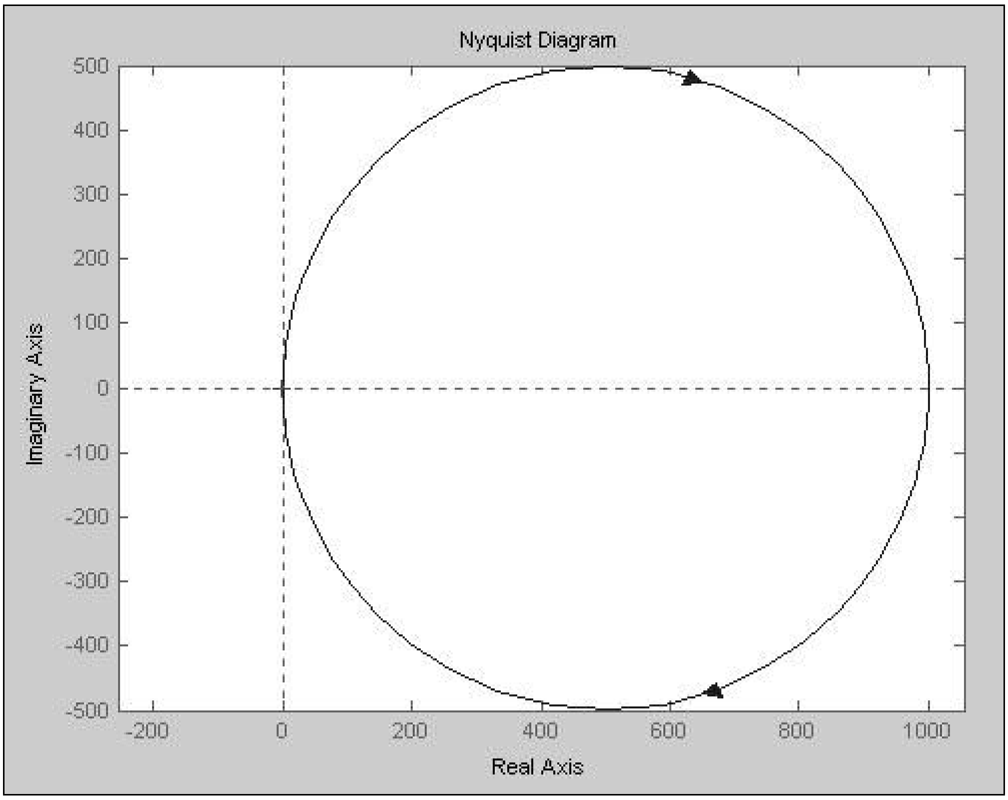


Figure 2.

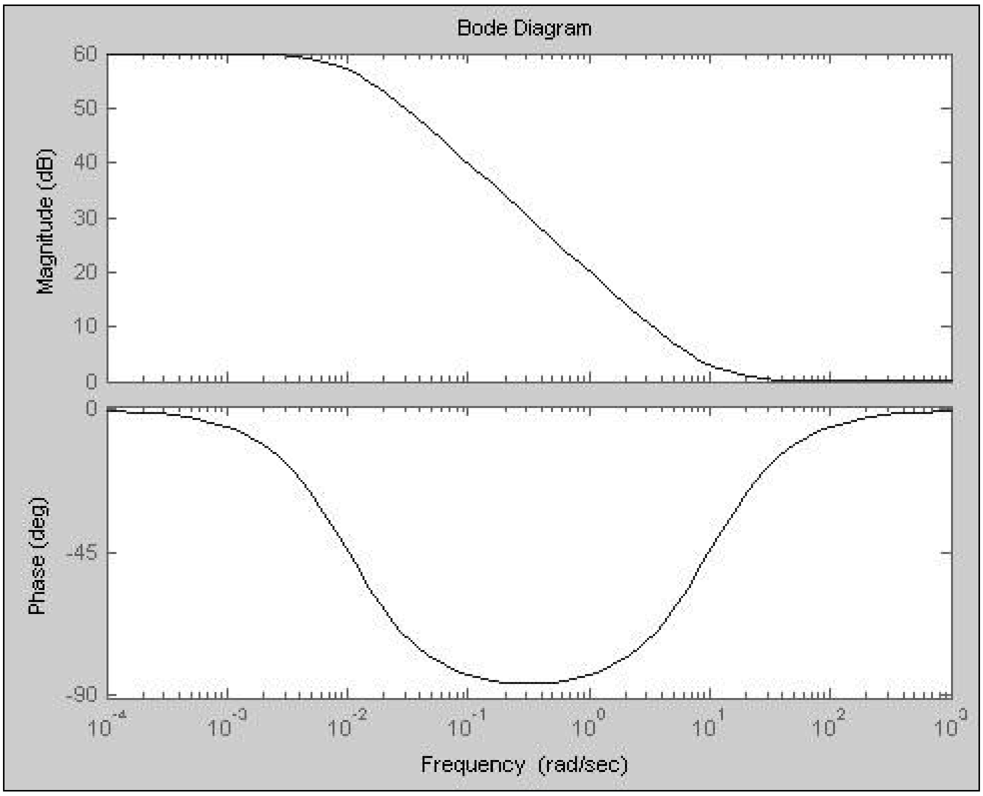


Figure 3.

Obtaining a step process,

```
>> PROCESS=tf([1],[1 4 4 2])

Transfer function:
          1
---------------------
s^3 + 4 s^2 + 4 s + 2

>> step(PROCESS)
```

The plot is shown in Figure 4.

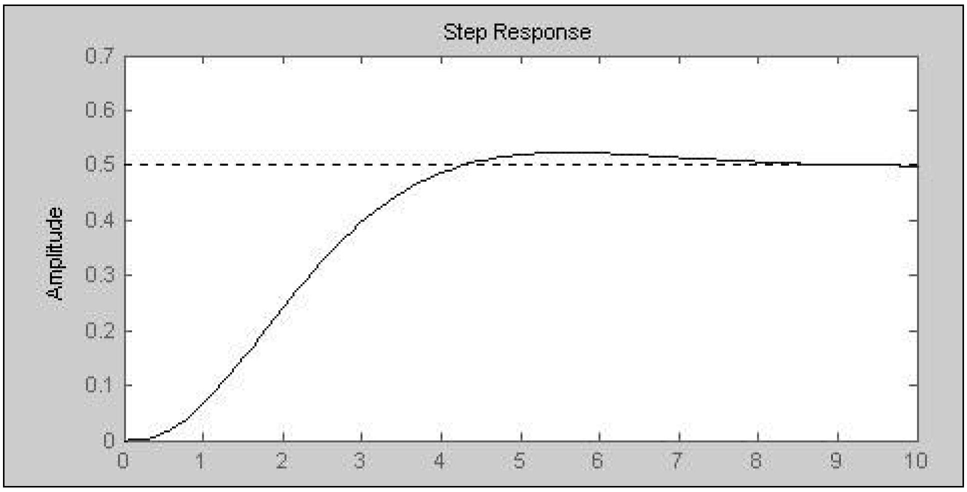


Figure 4.

7. Obtaining other information from control objects

Sometimes, information stored in the objects are needed, e.g. the poles and zeros of the transfer function. A short summary of commands is listed in Table 5, where G is a control object (please note the use of curly brackets instead of parenthesis):

Table 5.

Control Object Type	**Function Statement**	**Results**
zero-pole-gain	`ans = G.p{1}`	ans = vector of poles of G
zero-pole-gain	`ans = G.z{1}`	ans = vector of zeros of G
zero-pole-gain	`ans = G.k{1}`	ans = gain of G
transfer function	`ans = G.num{1}`	ans = vector of numerator coefficients
transfer function	`ans = G.den{1}`	ans = vector of denominator coefficients
state space	`ans = G.a`	ans = matrix A in state space formulation