Chapter 9

Transfer functions and block diagrams

In this chapter we will show how transfer functions can be used together with a graphic representation of system interconnection called block diagram. We conclude using block diagrams as a tool for a short introduction to control.

9.1 More on transfer functions

Remember Definition 4.1 about what is a transfer function of a SISO system modelled by a differential equation: it is the ratio between the Laplace transform of the output and the Laplace transform of the input, assuming initial conditions equal to zero. Also remember that behind each transfer function there is a differential equation, and that differential equations are models of real things.

Transfer functions are differential equations

Using transfer functions, we can easily study the behaviour of a system abstracting from its physical reality. This is the approach we will take from now on. This said, notice that remembering the actual system that is being studied can be useful to check if results are possible or not. Remember that models approximate reality, not the other way round. Also remember Example 4.1 about the spring stretched to an impossible length (it breaks, of course), or Remark 6.5 about pipes where the flow cannot be negative. We would not have found that just by looking at our models, which are linear.

Ratio of polynomials in s

Theorem 9.1. The transfer function of a SISO LTI continuous in time can be expressed as the ratio of two polynomials in s.

Proof. Let the input of the SISO system be $u(t)$ and its output be $y(t)$. Because the system is LTI and continuous in time, it is modelled by a linear differential equation:

$$a_0 y(t) + a_1 \frac{\mathrm{d}y(t)}{\mathrm{d}t} + a_2 \frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} + a_3 \frac{\mathrm{d}^3 y(t)}{\mathrm{d}t^3} + \ldots = b_0 u(t) + b_1 \frac{\mathrm{d}u(t)}{\mathrm{d}t} + b_2 \frac{\mathrm{d}^2 u(t)}{\mathrm{d}t^2} + b_3 \frac{\mathrm{d}^3 u(t)}{\mathrm{d}t^3} + \ldots$$

$$\Leftrightarrow \sum_{k=0}^{n} a_k \frac{\mathrm{d}^k y(t)}{\mathrm{d}t^k} = \sum_{k=0}^{m} b_k \frac{\mathrm{d}^k u(t)}{\mathrm{d}t^k} \tag{9.1}$$

In the last expression, n and m are the highest derivative orders of the equation. Assuming zero initial conditions and applying the Laplace transform, this becomes

$$a_0Y(s) + a_1Y(s)s + a_2Y(s)s^2 + a_3Y(s)s^3 + \ldots = b_0U(s) + b_1U(s)s + b_2U(s)s^2 + b_3U(s)s^3 + \ldots$$

$$\Leftrightarrow \sum_{k=0}^{n} a_kY(s)s^k = \sum_{k=0}^{m} b_kU(s)s^k \tag{9.2}$$

Rearranging terms,

$$\frac{Y(s)}{U(s)} = \frac{b_0 + b_1s + b_2s^2 + b_3s^3 + \ldots}{a_0 + a_1s + a_2s^2 + a_3s^3 + \ldots} = \frac{\sum_{k=0}^{m} b_ks^k}{\sum_{k=0}^{n} a_ks^k} \quad \square \tag{9.3}$$

$\square$

Remark 9.1. Some authors change the order of the coefficients, and instead of (9.3) write

$$\frac{Y(s)}{U(s)} = \frac{\sum_{k=0}^{m} b_{m-k}s^k}{\sum_{k=0}^{n} a_{m-k}s^k} \tag{9.4}$$

This is a mere detail of notation. $\square$

Normalising transfer function coefficients

Remark 9.2. (9.3) corresponds to an infinite number of representations of a same transfer function. It suffices to multiply both numerator and denominator by a constant. But it is common to normalise coefficients so that $a_0 = 1$, or $a_n = 1$, or $b_0 = 1$. $\square$

Example 9.1. Consider the microprecision control setup test in Figure 3.9 from Example 3.19. The transfer function from one of the actuators to the position of the mass has been identified as

$$G(s) = \frac{9602}{s^2 + 4.27s + 7627} \tag{9.5}$$

This was normalised so that $a_2 = 1$, $n = 2$. We could also normalise a_0 or b_0:

$$\begin{aligned} G(s) &= \frac{1.2589}{131.11 \times 10^{-6}s^2 + 559.85 \times 10^{-6}s + 1} \\ &= \frac{1}{104.14 \times 10^{-6}s^2 + 444.70 \times 10^{-6}s + 0.7943} \quad \square \end{aligned} \tag{9.6}$$

Getting the transfer function back

It is easy to find the differential equation from a transfer function. When the transfer function is represented merely by a letter, meaning that it is a function of s, as in (9.5) above, it still corresponds to the ratio of the Laplace transforms of output and input.

Example 9.2. (9.5) can be rewritten as

$$\frac{Y(s)}{U(s)} = \frac{9602}{s^2 + 4.27s + 7627} \Leftrightarrow Y(s)(s^2 + 4.27s + 7627) = 9602U(s)$$
$$\Leftrightarrow Y(s)s^2 + 4.27Y(s)s + 7627Y(s) = 9602U(s) \tag{9.7}$$

which is the Laplace transform of the differential equation governing the plant:

$$y''(t) + 4.27y'(t) + 7627y(t) = 9602u(t) \quad \square \tag{9.8}$$

Definition 9.1. A transfer function is **proper** if the order of the polynomial in the numerator is equal to or less than the order of the polynomial in the denominator.

Proper transfer function

A transfer function is **strictly proper** if the order of the polynomial in the numerator is less than the order of the polynomial in the denominator.

Strictly proper transfer function

In the notation of (9.3), the transfer function is proper if $m \leq n$, and strictly proper if $m < n$. □

For reasons we shall address in Chapter 10, we will be working only with proper transfer functions, and most of the times with strictly proper transfer functions.

Definition 9.2. The **order** of a transfer function is the highest order of the polynomials in the numerator and the denominator. If the transfer function is proper, its order is the order of the denominator. □

Order of a transfer function

Remark 9.3. The order of a transfer function is also the order of the differential equation from which it was formed. In fact, s^k corresponds to a derivative of order k. □

Remark 9.4. Notice that some transfer functions can be simplified because numerator and denominator have common factors. Eliminating them reduces the order of the transfer function. □

Example 9.3. Here are examples of proper transfer functions of:

- Order 0

$$G_a(s) = 20 \tag{9.9}$$

- Order 1

$$G_b(s) = \frac{19}{s + 18} \tag{9.10}$$
$$G_c(s) = \frac{17s + 16}{s + 15} \tag{9.11}$$
$$G_d(s) = \frac{14}{s} \tag{9.12}$$
$$G_e(s) = \frac{13s + 12}{s} \tag{9.13}$$

- Order 2

$$G_f(s) = \frac{11}{s^2 + 10s + 9} \tag{9.14}$$

$$G_g(s) = \frac{8s + 7}{s^2 + 6s + 5} \tag{9.15}$$

$$G_h(s) = \frac{4s^2 + 3s + 2}{s^2 + s - 1} \tag{9.16}$$

$$G_i(s) = \frac{s^2 - 2s + 1}{s^2} \tag{9.17}$$

$$G_j(s) = \frac{s^2 - 3s - 4}{s^2 - 5s - 6} \tag{9.18}$$

They have all been normalised so that the coefficient of the highest order monomial in the denominator is 1 (i.e. $a_n = 1$). Transfer functions $G_b(s)$, $G_d(s)$, $G_f(s)$, $G_g(s)$, and $G_i(s)$ are strictly proper; the other ones are not.

$G_j(s)$ is of order 2 but can be simplified and become of order 1:

$$G_j(s) = \frac{s^2 - 3s - 4}{s^2 - 5s - 6} = \frac{(s-4)(s+1)}{(s-6)(s+1)} = \frac{s-4}{s-6} \quad \square \tag{9.19}$$

Zeros
Poles

Transfer functions are often put in the following form, that explicitly shows the **zeros** of the transfer function (i.e. the zeros of the polynomial in the numerator) and the **poles** of the transfer function (i.e. the zeros of the polynomial in the denominator):

$$\frac{Y(s)}{U(s)} = \frac{b_m(s-z_1)(s-z_2)(s-z_3)\ldots}{a_n(s-p_1)(s-p_2)(s-p_3)\ldots} = \frac{b_m \prod_{k=1}^{m}(s-z_k)}{a_n \sum_{k=0}^{n}(s-p_k)} \tag{9.20}$$

Here the zeros are $z_k,\ k = 1, 2, \ldots m$ and the poles are $p_k,\ k = 1, 2, \ldots m$. Because both inputs and outputs are real, transfer function coefficients are real, and consequently the poles and zeros are either real or pairs of complex conjugates. (Remember Remark 2.6.) So in (9.20) it is usual to multiply such pairs, presenting a second order term instead of two complex terms.

Example 9.4. The second order transfer functions in Example 9.3 can be rewritten as

$$G_f(s) = \frac{11}{(s+9)(s+1)} \tag{9.21}$$

$$G_g(s) = \frac{8s+7}{(s+5)(s+1)} \tag{9.22}$$

$$G_h(s) = \frac{4\left(s + \frac{3+\sqrt{23}j}{8}\right)\left(s + \frac{3-\sqrt{23}j}{8}\right)}{\left(s + \frac{1+\sqrt{5}}{2}\right)\left(s + \frac{1-\sqrt{5}}{2}\right)} = \frac{4s^2 + 3s + 2}{\left(s + \frac{1+\sqrt{5}}{2}\right)\left(s + \frac{1-\sqrt{5}}{2}\right)} \tag{9.23}$$

$$G_i(s) = \frac{(s-1)^2}{s^2} \tag{9.24}$$

For $G_j(s)$, see (9.19). Notice that, in the case of $G_h(s)$, only the second expression is usual; the first one, explicitly showing the two complex conjugate zeros, is not. □

Remark 9.5. From Definition 9.2 results that the order of a proper transfer function is the number of its poles. □

The following MATLAB functions use transfer functions in this form:

Transfer function from zeros, poles, gain

- `zpk` creates a transfer function from its zeros, poles, and the $\frac{b_m}{a_n}$ ratio in (9.20), here called gain k, and also converts a transfer function created with `tf` into this form;
- `pole` finds the poles of a transfer function;
- `tzero` finds the zeros of a transfer function.

Example 9.5. Transfer function (9.15) or (9.22)

- has one zero, $8s + 7 = 0 \Leftrightarrow s = -\frac{7}{8} = -0.875$,
- has two poles, $(s + 5)(s + 1) = 0 \Leftrightarrow s = -5 \vee s = -1$,
- verifies $k = \frac{b_m}{a_n} = \frac{8}{1} = 1$.

MATLAB's command ***zpk***

It can be created, converted to a ratio of two polynomials as in (9.3), and converted back to the (9.20) form as follows:

```
>> G_g = zpk(-7/8, [-5 -1], 8)

G_g =

  8 (s+0.875)
  -----------
  (s+5) (s+1)

Continuous-time zero/pole/gain model.

>> G_g = tf(G_g)

G_g =

     8 s + 7
  -------------
  s^2 + 6 s + 5

Continuous-time transfer function.

>> G_g = zpk(G_g)

G_g =

  8 (s+0.875)
  -----------
```

```
  (s+5) (s+1)

Continuous-time zero/pole/gain model.
```

MATLAB's commands pole and tzero

Its poles and zeros can be found as follows:

```
>> tzero(G_g)
ans =
   -0.8750
>> pole(G_g)
ans =
    -5
    -1
```

It does not matter whether a transfer function was created with `tf` or with `zpk` (or with any other function to create transfer functions that we did not study yet): `pole` and `tzero` work just the same.

Another way of finding the poles and the zeros is to access the numerator and the denominator, and then using `roots` to find the roots of these polynomials. The transfer function must be in the `tf` form this time, the only one that has the `num` and `den` fields:

```
>> G_g = tf(G_g);
>> G_g.num{1}
ans =
     0     8     7
>> roots(ans)
ans =
   -0.8750
>> G_g.den{1}
ans =
     1     6     5
>> roots(ans)
ans =
    -5
    -1
```

Notice that the `{1}` is necessary since MATLAB presumes that the transfer function is MIMO and thus has many transfer functions relating the many inputs with the many outputs. The cell array index accesses the first transfer function, which, as the system is SISO, is the only one. □

Multiplying transfer functions

A very important property of transfer functions for the rest of this chapter has already been mention in Section 8.2 and illustrated in Example 8.2: if two systems $G_1(s) = \frac{y_1(s)}{u_1(s)}$ and $G_2(s) = \frac{y_2(s)}{u_2(s)}$ are interconnected so that the output of one is the input of the other, $y_1(s) = u_1(s)$, then the resulting transfer function is

$$\frac{y_2(s)}{u_1(s)} = \frac{y_2(s)}{u_2(s)}\,\frac{y_1(s)}{u_1(s)} = G_1(s)\,G_2(s) \tag{9.25}$$

Multiplication of $\mathscr{L}$ is convolution in t

Remark 9.6. Remember that the multiplication of two Laplace transforms does not correspond to the multiplication of the original functions, but rather

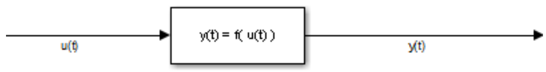

Figure 9.1: Generic block.

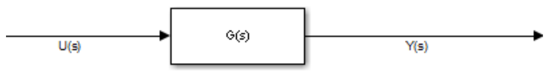

Figure 9.2: Linear block.

to their convolution, as we have shown in (2.78). Operation convolution is defined in (2.76). (This is sometimes a source of confusion, because the sum of two Laplace transforms is the sum of the original functions, as $\mathscr{L}$ is linear.) $\square$

9.2 Block diagrams

Block diagrams are graphical representations of the relations between variables and functions. In our case, functions will be systems, and variables will be signals (which are themselves, as you remember, functions of time, or space). Figure 9.1 shows a generic system (represented by a block) relating two signals (represented by lines with arrows).

The practical thing to do for LTI systems is to represent them using their transfer functions, and consequently to represent signals by their Laplace transforms. The block in Figure 9.2 means that $Y(s) = G(s)U(s)$. This is yet another advantage of using the Laplace transform: the (Laplace transform of the) output is the product of the (transfer function of the) system and the (Laplace transform of the) input.

Example 9.6. The mechatronic system in Example 8.2 had four transfer functions, as follows:

$$G_1(s) = \frac{I(s)}{V_i(s)} = \frac{\frac{n_2}{n_1}}{R + Ls} \tag{9.26}$$

$$G_2(s) = \frac{F_2(s)}{I(s)} = \alpha \tag{9.27}$$

$$G_3(s) = \frac{F_1(s)}{F_2(s)} = \frac{b}{a} \tag{9.28}$$

$$G_4(s) = \frac{X_1(s)}{F_1(s)} = \frac{1}{m_1 s^2 + K} \tag{9.29}$$

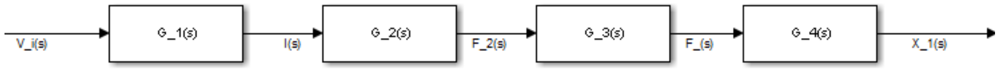

Figure 9.3: Block diagram of Example 9.6, corresponding to the mechatronic system in Figure 8.2 from Example 8.2.

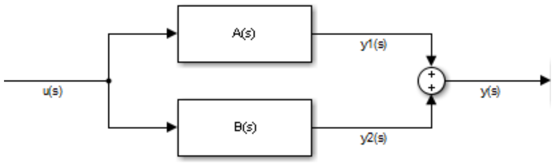

Figure 9.4: Block diagram with two blocks in parallel.

The corresponding block diagram is shown in Figure 9.3. In fact,

$$I(s) = G_1(s)V_i(s) \tag{9.30}$$
$$F_2(s) = G_2(s)I(s) \tag{9.31}$$
$$F_1(s) = G_3(s)F_2(s) \tag{9.32}$$
$$X_1(s) = G_4(s)F_1(s) \quad \square \tag{9.33}$$

Blocks in series
Blocks in cascade

The Example above shows that several interconnected systems correspond to a sequence of blocks. By similarity with electrical circuits, blocks in such a sequence are said to be in **series** or in **cascade**. This is because of the property of transfer functions illustrated in (9.25). Clearly, two blocks A and B in series are equivalent to one block AB.

Adding signals is represented as shown in Figure 9.4, where

$$y = y_1 + y_2 = Au + Bu = (A + B)u \tag{9.34}$$

Blocks in parallel

By similarity with electrical circuits, blocks A and B are said to be in **parallel**. Clearly, they are equivalent to one block $A+B$. Signal subtraction is indicated similarly.

Feedback
Loop
Direct branch
Feedback branch

The block configurations in Figure 9.5, wherein the input of a block depends on its output, is called **feedback loop** or just **feedback**: feedback, because the output is fed back to the block it originates from; and loop, because of the configuration of the diagram. In that Figure, A is called **direct branch** and B **feedback branch**. The two block diagrams only differ because of the sign affecting signal $d(s)$:

Negative feedback

- when $b(s) = a(s) - d(s)$, there is **negative feedback**;

Positive feedback

- when $b(s) = a(s) + d(s)$, there is **positive feedback**.

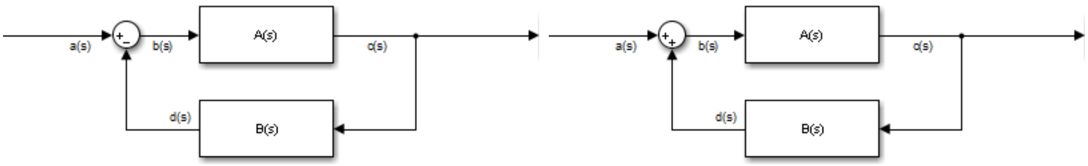

Figure 9.5: Block diagrams with feedback loops. Left: negative feedback. Right: positive feedback.

Negative feedback is far more common; when feedback is mentioned without specifying whether it is positive or negative, you can safely presume it is negative. Notice that, for both:

- the input of the loop is $a(s)$; *Input of the feedback loop*
- the output of the loop is $c(s)$; *Output of the feedback loop*
- the input of the direct branch is $b(s) = a(s) \mp d(s)$;
- the output of the direct branch is $a(s)$;
- the input of the feedback branch is $c(s)$;
- the output of the feedback branch is $d(s)$.

Consequently, for negative feedback,

$$\begin{aligned} c = Ab = A(a-d) = A(a-Bc) = Aa - ABc \\ \Rightarrow c + ABc = Aa \Rightarrow c = a\frac{A}{1+AB} \end{aligned} \tag{9.35}$$

and, for positive feedback,

$$\begin{aligned} c = Ab = A(a+d) = A(a+Bc) = Aa + ABc \\ \Rightarrow c - ABc = Aa \Rightarrow c = a\frac{A}{1-AB} \end{aligned} \tag{9.36}$$

Centrifugal governor

Example 9.7. The centrifugal governor (see Figure 9.6) is a control system which had widespread use to control the pressure in boilers. It rotates because of the pressure of the steam. The faster it rotates, the more the two spheres go up, thereby opening a valve relieving steam pressure. Consequently the regulator spins slower, the balls go down, and this closes the valve, so pressure is no longer relieved and goes up again. This is negative feedback: an increase of any variable has as consequence the decrease of another variable that caused the original increase, and vice-versa. □

Figure 9.6: Centrifugal governor of a boiler in the former Barbadinhos water pumping station (currently the Water Museum), Lisbon.

Example 9.8. Audio feedback (or "howl") is an example of positive feedback. Surely you must have heard it often, whenever there is a sound system amplifying the sound detected by a microphone which is too close to the loudspeakers, so that even background noise is amplified to the point of being received again by the microphone and amplified further — see Figure 9.7. The amplitude of the resulting sound does not become infinite because at some point the amplifier and/or the loudspeakers saturate, but the "howl" can damage the equipment or, more importantly, the listeners' auditory systems. □

Example 9.9. Biological processes provide numerous examples of both positive and negative feedback. We will go back to this in Chapter 14. □

The best way to simplify block diagrams is to write the corresponding equations and do so analytically.

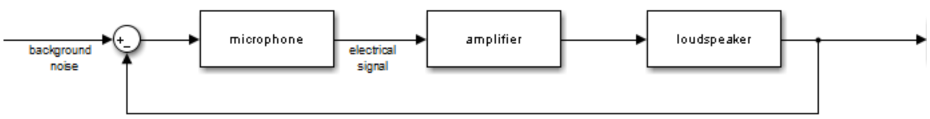

Figure 9.7: How audio feedback occurs.

Example 9.10. In the block diagram of Figure 9.8 we make

$$G_1(s) = 2 \tag{9.37}$$
$$G_2(s) = \frac{s+10}{s^2+0.5s+5} \tag{9.38}$$
$$G_3(s) = \frac{1}{s+1} \tag{9.39}$$
$$G_4(s) = \frac{20(s-0.5)}{(s-1)(s-3)} \tag{9.40}$$
$$G_5(s) = \frac{1}{s} \tag{9.41}$$
$$\tag{9.42}$$

(The block for $G_1(s)$ is triangular because SIMULINK, which we will mention below, uses triangles for constants, but this convention is unusual; when drawing blocks by hand, they are all usually rectangles.) Then

$$e = G_2c = G_2G_1b = G_2G_1(a-d) = G_1G_2(a-G_3e) \Rightarrow (1+G_1G_2G_3)e = G_1G_2a \Rightarrow e = a\frac{G_1G_2}{1+G_1G_2G_3} \tag{9.43}$$

$$h = G_5g = G_5(e+f) = G_5\left(a\frac{G_1G_2}{1+G_1G_2G_3} + G_4a\right) = a\left(\frac{G_1G_2G_5}{1+G_1G_2G_3} + G_4G_5\right) \tag{9.44}$$

Finally, the whole block diagram corresponds to transfer function

$$\frac{h(s)}{a(s)} = \frac{2\frac{s+10}{s^2+0.5s+5}\frac{1}{s}}{1+2\frac{s+10}{s^2+0.5s+5}\frac{1}{s+1}} + \frac{20(s-0.5)}{s(s-1)(s-3)} \tag{9.45}$$

It is usually a good idea to put the result in one of the forms (9.3) or (9.20). Since calculations are rather complicated, we can use MATLAB:

```
>> s = tf('s');
>> (2/s*(s+10)/(s^2+0.5*s+5))/(1+2/(s+1)*(s+10)/(s^2+0.5*s+5))+...
20*(s-0.5)/((s-1)*(s-3)*s)

ans =


   22 s^7 + 45 s^6 + 200 s^5 + 617.5 s^4 + 380.5 s^3 + 1960 s^2 - 950 s

  ----------------------------------------------------------------------

  s^9 - 2 s^8 + 8.25 s^7 - 10.75 s^6 - 55.25 s^5 + 33.75 s^4 - 350 s^3

                                                              + 375 s^2


Continuous-time transfer function.
```

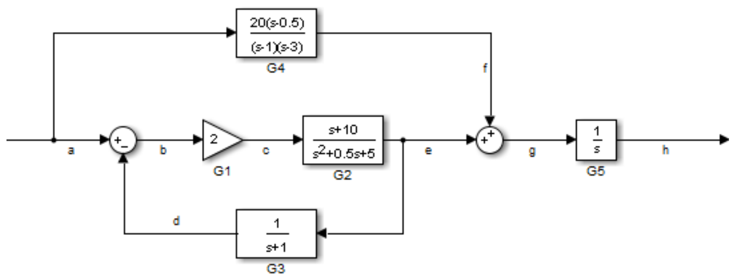

Figure 9.8: Block diagram of Example 9.10.

```
>> zpk(ans)

ans =

  22 s (s+2.931) (s-0.4226) (s^2 + 0.5s + 5) (s^2 - 0.9629s + 6.972)
  -----------------------------------------------------------------
       s^2 (s-3) (s+2.5) (s-1) (s^2 + 0.5s + 5) (s^2 - s + 10)

Continuous-time zero/pole/gain model.
```

As you can see from the last result, it is possible to eliminate s and $s^2+0.5*s+5$ from both the numerator and the denominator. So $\frac{h(s)}{a(s)}$ is of sixth order. □

MATLAB has commands to combine transfer functions:

- operators + and * add and multiply transfer functions (remember that two blocks in series correspond to the product of their transfer functions);
- `feedback` receives the direct and the feedback branches and gives the transfer function of the negative feedback loop.

MATLAB's *command* `feedback`

Example 9.11. We can verify our calculations of Example 9.10 as follows:

```
>> G1 = 2;
>> G2 = (s+10)/(s^2+0.5*s+5);
>> G3 = 1/(s+1);
>> G4 = 20*(s-0.5)/((s-1)*(s-3));
>> G5 = 1/s;
>> loop_from_a_to_e = feedback(G1*G2, G3)

loop_from_a_to_e =

      2 s^2 + 22 s + 20
  --------------------------
  s^3 + 1.5 s^2 + 7.5 s + 25

Continuous-time transfer function.
```

Figure 9.9: Commonly used blocks of SIMULINK.

```
>> from_a_to_g = loop_from_a_to_e + G4

from_a_to_g =

      22 s^4 + 34 s^3 + 73 s^2 + 411 s - 190
  -----------------------------------------------
  s^5 - 2.5 s^4 + 4.5 s^3 - 0.5 s^2 - 77.5 s + 75

Continuous-time transfer function.

>> from_a_to_h = from_a_to_g * G5

from_a_to_h =

        22 s^4 + 34 s^3 + 73 s^2 + 411 s - 190
  ---------------------------------------------------
  s^6 - 2.5 s^5 + 4.5 s^4 - 0.5 s^3 - 77.5 s^2 + 75 s

Continuous-time transfer function.

>> zpk(from_a_to_h)

ans =

  22 (s+2.931) (s-0.4226) (s^2 - 0.9629s + 6.972)
  -----------------------------------------------
       s (s+2.5) (s-3) (s-1) (s^2 - s + 10)

Continuous-time zero/pole/gain model.
```

This is the same transfer function we found above, with the poles and zeros common to the numerator and denominator eliminated. □

SIMULINK

MATLAB's most powerful tool for working with block diagrams is SIMULINK. All the block diagrams above have been created with SIMULINK, and then cropped so as not to show what SIMULINK calls source and sink, which are not part of what is shown in standard block diagrams (you must not include them when drawing block diagrams by hand). To use SIMULINK, access its library in MATLAB by clicking the corresponding button or typing `simulink`. The library looks like very different in different versions of MATLAB, but its organisation is similar: the most commonly used blocks are in one of the several subsets of the `Simulink` library; then there are libraries corresponding to the

Commonly used SIMULINK *blocks*

toolboxes you have installed. Figure 9.9 shows the blocks you will likely need:

- The `Transfer Fcn` block, from the `Continuous` subset of the `Simulink` library, creates a transfer function like function `tf`.
- The `Zero-Pole` block, from the `Continuous` subset of the `Simulink` library, creates a transfer function like function `zpk`.
- The `LTI System` block, from the `Control System Toolbox` library, creates a transfer function using function `tf` or function `zpk`. It is also possible just to put there a variable with a transfer function, created in the command line.
- The `Sum` block (name hidden by default), from the `Math operations` subset of the `Simulink` library, is a sum point.
- The `Gain` block, from the `Math operations` subset of the `Simulink` library, multiplies a signal by a constant.
- The `From Workspace` block, from the `Sources` subset of the `Simulink` library, provides a signal to run a simulation. The signal is either a structure (see the block's dialogue for details) or a matrix with time instants in the first column and the corresponding values of the signal in the second column (these will be interpolated).
- The `Scope` block, from the `Sinks` subset of the `Simulink` library, plots the signal it receives. It can be configured to record the data to a variable in the workspace, which is most practical to reuse it later.
- The `Mux` block (from "multiplexer", name hidden by default), from the `Signal Routing` subset of the `Simulink` library, joins two (or more) signals into one. The result is a vector-valued signal. If two real-values signals are multiplexed, the result is a vector-valued signal with dimension 2.

To use a block, create an empty SIMULINK file and drag it there. Connect blocks with arrows by clicking and dragging from one block's output to another's input. Double-click a block to see a dialogue where you can fill in the arguments you would use in a MATLAB command written in the command line (i.e. after the `>>`). Most of the times you can use numbers or variables; you just have to create the variables before you create the model. Right-clicking a block shows a context menu with many options, among which those of showing or hiding the block's name, or rotating it. You can edit a block's name by clicking it, and add a label to a signal by double-clicking it.

To run a simulation, choose its duration in the box on the top of the window, or go to `Simulation > Model Configuration Parameters`. Then click the Play button, or use command `sim` with the name of the (previously saved) file with the block diagram model.

Example 9.12. Let us simulate the mechatronic system of Examples 8.2 and 9.6, given by (9.26)–(9.29). The SIMULINK file is as shown in Figure 9.10 and its running time was set to 3 s; variables have been used and must be defined before running the simulation, but this means that they are easier to change. Block `From Workspace` has matrix `[0 1]`, meaning that at time 0 it will output value

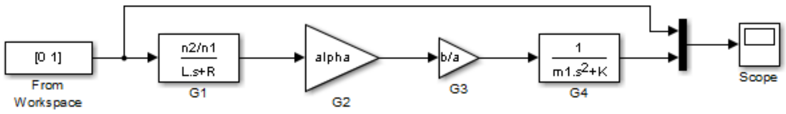

Figure 9.10: SIMULINK file of Example 9.12.

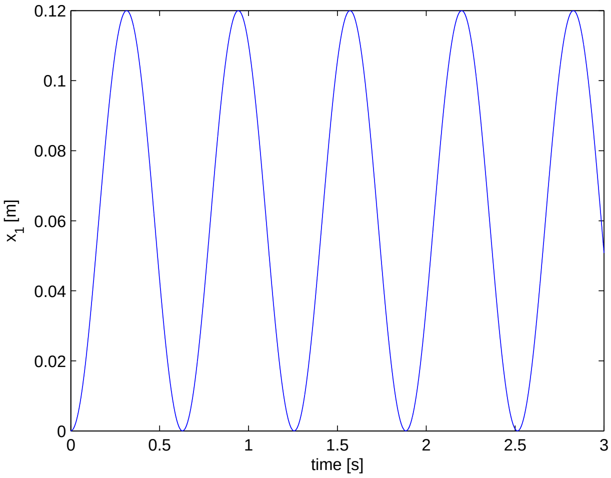

Figure 9.11: Output of Example 9.12.

1, and since no other value is provided this one will be kept. So we are finding the response of the system to a Heaviside function (2.5), or rather to a tension of 1 V being applied when the simulation begins. Block `Scope` is configured to save data to variable `Data`. The following commands create the variables, run the simulation, and plot again the results which you could also see in the `Scope` itself:

```
>> n2 = 200; n1 = 100; L = 1e-2; R = 100; alpha = 100; b = 0.3; a = 0.1; m1 = 1; K = 100;
>> sim('prob3_ficha3_2011_modif_2')
>> figure, plot(Data.time,Data.signals.values(:,2)), xlabel('time [s]'), ylabel('x_1 [m]')
```

See Figure 9.11. We could have expected these oscillations with constant amplitude, and you will know why in Chapter 10. □

Remark 9.7. Notice that the input signal was specified in time and the output variable was obtained as a function of time, but the differential equations were specified as transfer functions, i.e. not as relations in variable t but in the Laplace transform variable s. This is the way SIMULINK works. However, do not forget that, since in a block diagram system dynamics is indicated by transfer functions, signals too must be given by their Laplace transforms, as functions

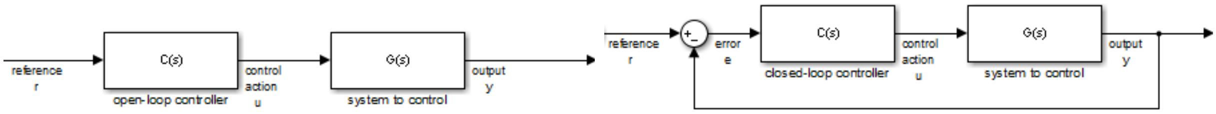

Figure 9.12: Left: open loop control. Right: closed loop control.

of s. It is correct to say that $y(s) = G(s)u(s)$; it makes no sense at all to write $y(t) = G(s)u(t)$ mixing t and s. □

The dialogue `Model Configuration Parameters`, which can also be accessed through a button, allows specifying many other things, among which:

- the numerical method used to solve the differential equations;
- the maximum and minimum time steps used by the numerical method;
- a tolerance that will not be exceeded by the numerical method's estimate of the errors incurred.

Notice that some numerical methods use fixed time steps. These may be used with differential equations, but are the only ones that can be used with difference equations (corresponding to digital models).

9.3 Control in open-loop and in closed-loop

There are two generic configurations for control systems: **open-loop control** and **closed-loop control**, shown in Figure 9.12. Every control system is a variation of one of these two configurations, or a combination thereof. Both add, to the system we want to control, another system called **controller**, intended to make the controlled system's output $y(t)$ follow some specified reference, or desired output, $r(t)$ (remember Section 3.1). In a perfectly controlled system, $y(t) = r(t)$, $\forall t$. The output of the controller is the system's input in the strict sense (the input must be a manipulated variable).

Open-loop control

In open-loop control the controller receives the reference that the system should follow, and decides from this desired output what control action to take. This control action will be the input of the system. It is not checked whether or not the system's output does follow the reference. So, if there is some unexpected deviation from the reference, this does not change the control action. Open-loop control only uses blocks in series.

In open-loop control,

$$y(s) = G(s)u(s) = G(s)C(s)r(s) \tag{9.46}$$

and since we want $y(s) = r(s)$ then we should have $C(s) = G^{-1}(s)$, i.e. the controller should be an **inverse model** of the system to control. Notice that if the model of the system is proper then the controller is not proper; you will learn why this brings problems in Chapter 10.

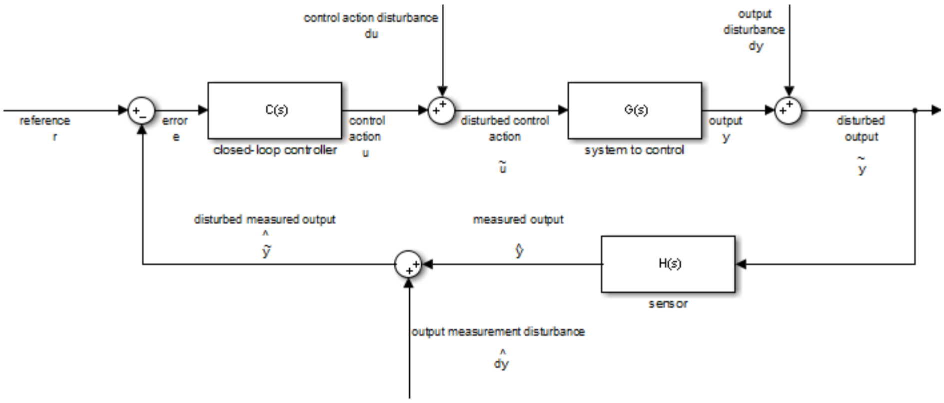

Figure 9.13: Closed-loop control with disturbances and sensor dynamics.

ed-loop control

Closed-loop control uses negative feedback. The reference is compared with the system output. Ideally, the error should be zero. What the controller receives is this error, so the control action is based on the error.

Proportional control

The simplest closed-loop controller is proportional: $C(s) = K \in \mathbb{R}$. With **proportional control**, if the error is small, the control action is small too; if the error is large, the control action is also large. There are techniques to choose an appropriate value of K, and also to develop more complex controllers, with poles and zeros, which you will learn in other courses.

Actually, no control system is that simple. Figure 9.13 shows a more realistic situation, including the following additions:

Sensor dynamics

- $H(s)$ is the **sensor** that measures output y. A perfect sensor measures the output exactly: $\hat{y}(t) = y(t),\ \forall t$; and hence $H(s) = 1$. No sensor is perfect, but it is often possible to assume $H(s) = 1$ even so (in which case the block does not need to be there). If this is not the case, $H(s)$ must be explicitly taken into account.

Control action disturbance

- $d_u(t)$ is a disturbance that affects the control action. This means that the control action is not precisely received by the controlled system. For instance, if the control action is a force, this means that there are other forces acting upon the system. Or, if the control action is a current, there are unintended fluctuations of the value determined by the controller.

System output disturbance

- $d_y(t)$ is a disturbance that affects the system output. This means that the output is affected by something else other than the system. For instance, if the output is a flow, there is some other source of fluid, or some bleeding of fluid somewhere, that must be added or subtracted. Or, if the output is a position, there may be vibrations that have to be superimposed.

Output measurement disturbance

- $d_{\hat{y}}(t)$ is a disturbance that affects the sensor's measurement of the system output. Just like $u(t)$ can suffer a disturbance, so can $\hat{y}(t)$.

Additive disturbances
Multiplicative disturbances

Remark 9.8. Disturbances in Figure 9.13 follow what is called an additive model, since the disturbance is added to the signal it disturbs. Other models use multiplicative disturbances, that are multiplied rather than summed. Here we will stick to additive disturbances, which result in linear models. □

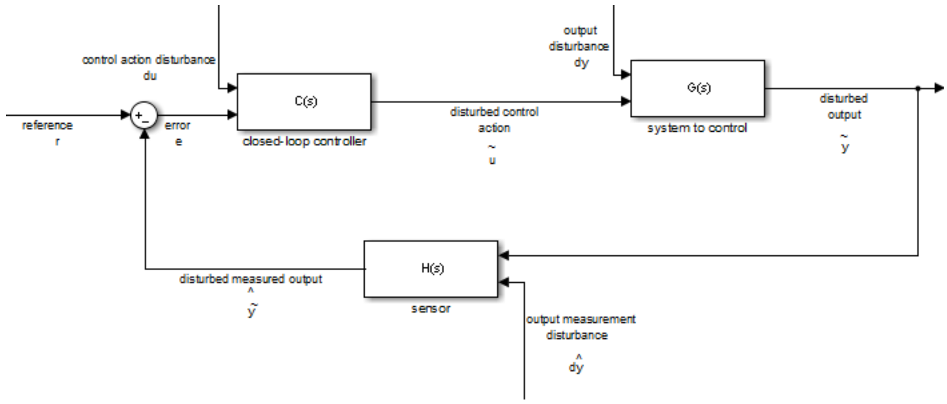

Figure 9.14: The same as Figure 9.13, but with MIMO systems.

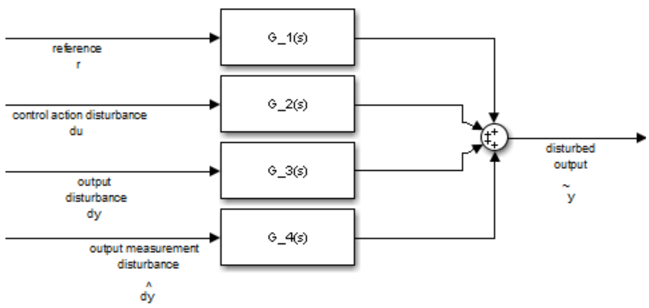

Figure 9.15: The same as Figure 9.13, but using transfer functions (9.48)–(9.51).

Remark 9.9. We saw in Chapter 3 that MIMO systems may have some inputs in the general sense that are disturbances and others that are manipulated variables. Figure 9.14 represents disturbances using MISO systems. The block diagram in Figure 9.13 reflects the same situation using only SISO systems. The price to pay for using SISO systems is less freedom in establishing mathematical relations between disturbances and outputs. □

The output of the block diagram in Figure 9.13 is

$$\begin{aligned}
\tilde{y} &= d_y + y = d_y + G\tilde{u} = d_y + G(d_u + u) = d_y + Gd_u + GCe \\
&= d_y + Gd_u + GC(r - \hat{\tilde{y}}) = d_y + Gd_u + GCr - GC(d_{\hat{y}} + \hat{y}) \\
&= d_y + Gd_u + GCr - GCd_{\hat{y}} - GCH\tilde{y} \\
\Rightarrow (1 + GCH)\tilde{y} &= d_y + Gd_u + GCr - GCd_{\hat{y}} \qquad (9.47) \\
\Rightarrow \tilde{y} &= \frac{1}{1+GCH}d_y + \frac{G}{1+GCH}d_u + \frac{GC}{1+GCH}r + \frac{-GC}{1+GCH}d_{\hat{y}}
\end{aligned}$$

Because of the linearity of the relations involved, (9.47) gives the same result as if four transfer functions were involved as seen in Figure 9.15:

$$G_1 = \frac{\tilde{y}}{r} = \frac{GC}{1+GCH} \qquad (9.48)$$

$$G_2 = \frac{\tilde{y}}{d_u} = \frac{G}{1+GCH} \qquad (9.49)$$

$$G_3 = \frac{\tilde{y}}{d_y} = \frac{1}{1+GCH} \qquad (9.50)$$

$$G_4 = \frac{\tilde{y}}{d_{\hat{y}}} = \frac{-GC}{1+GCH} \qquad (9.51)$$

Notice that each of the four transfer functions above can be obtained assuming that all inputs but one of them are zero. If the system were not linear, that would not be the case.

Glossary

> D'altra parte gli aveva detto la sera prima che lui possedeva un'dono: che gli bastava udire due che parlavano in una lingua qualsiasi, e dopo un poco era capace di parlare come loro. Dono singolare, che Niceta credeva fosse stato concesso solo agli apostoli.
>
> Umberto Eco (1932 — †2016), *Baudolino*, 2

block diagram diagrama de blocos
blocks in cascade blocos em cascata
blocks in parallel blocos em paralelo
closed-loop anel fechado, malha fechada
direct branch ramo direto
disturbance perturbação
feedback retroação
feedback branch ramo de retroação

feedback loop anel de retroação, malha de retroação
inverse model modelo inverso
open-loop anel aberto, malha aberta
order ordem
proper transfer function função de transferência própria
proportional control controlo proporcional
strictly proper transfer function função de transferência estritamente própria

Exercises

1. For each of the transfer functions below, answer the following questions:
   - What are its poles?
   - What are its zeros?
   - What is its order?
   - Is it a proper transfer function?
   - Is it a strictly proper transfer function?
   - What is the differential equation it corresponds to?

   (a) $\frac{s}{s^2+12s+20}$

   (b) $\frac{s+1}{s-5}$

   (c) $\frac{s^2+2s+10}{s^3-5s^2+15.25s}$

   (d) $\frac{10}{(s+1)^2(s^2+5s+6)}$

   (e) $\frac{s^2+2}{s^2(s+3)(s+50)}$

   (f) $\frac{(s^4+6s^3+8.75s^2)}{(s^2+4s+4)^2}$

2. Find the following transfer functions for the block diagram in Figure 9.16:

   (a) $\frac{y(s)}{d(s)}$

   (b) $\frac{y(s)}{r(s)}$

   (c) $\frac{y(s)}{m(s)}$

   (d) $\frac{y(s)}{n(s)}$

   (e) $\frac{u(s)}{d(s)}$

   (f) $\frac{u(s)}{r(s)}$

   (g) $\frac{u(s)}{m(s)}$

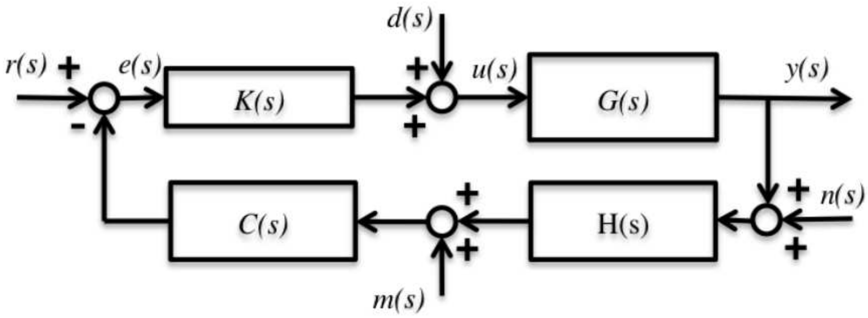

Figure 9.16: Block diagram of Exercise 2.

(h) $\frac{u(s)}{n(s)}$

(i) $\frac{e(s)}{d(s)}$

(j) $\frac{e(s)}{r(s)}$

(k) $\frac{e(s)}{m(s)}$

(l) $\frac{e(s)}{n(s)}$

3. Figure 9.17 shows a variation of closed-loop control called internal model control (IMC). It has this name because it requires knowing a model of the system to control, as well as an inverse model of the system to control.In that block diagram:

    - $G(s)$ is the plant to control,
    - $G^*(s)$ is the model of the plant to control,
    - $G^{-1}(s)$ is the inverse model of the plant to control.

    (a) Show that, if the model is perfect, i.e. if $G^*(s) = G(s)$, then the error is given by $E(s) = R(s) - D(s)$.

    (b) Show that, if, additionally, the inverse model is perfect, i.e. $G^{-1}(s)G(s) = 1$, then the output is $Y(s) = R(s)$.

    (c) Show that, whether the models are perfect or not, the block diagram of IMC in Figure 9.17 is equivalent to the block diagram of closed-loop control in Figure 9.12, if $C(s) = \frac{G^{-1}(s)}{1-G^{-1}(s)G^*(s)}$.

4. Figure 9.18 shows a variation of closed-loop control called cascade control (or master–slave control, though that designation is out of favour nowadays). In that block diagram, the plant to control is $G(s) = G_1(s)G_2(s)$, and it possible to measure both $Y_1(s)$ and $Y_2(s)$. Each of the two parts of the system to control is controlled separately.

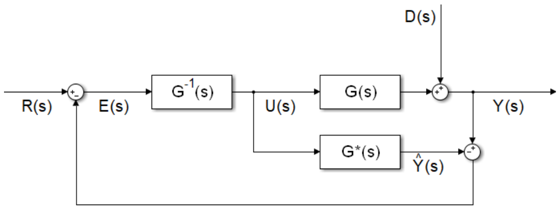

Figure 9.17: Internal model control (IMC).

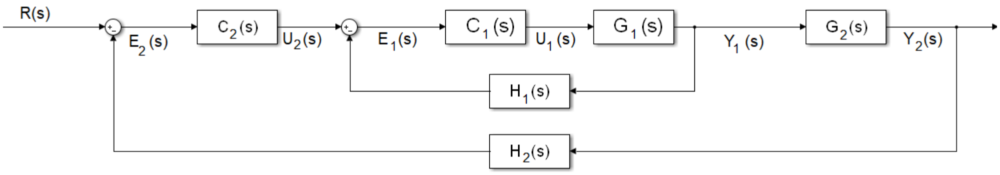

Figure 9.18: Cascade (or master–slave) control.

(a) Find transfer function $\dfrac{Y_1(s)}{U_2(s)}$.

(b) Use that result to find transfer function $\dfrac{Y_2(s)}{R(s)}$.

5. Redraw the block diagram of Figure 9.10 from Example 9.12 as follows:

   - use the values of the variables given in Example 9.12,
   - let the input $V_i(s)$ be a manipulated variable,
   - let there be some reference $r(t)$ for $x_1(t)$ to follow,
   - add proportional control K.

   Then find transfer function $\frac{X_1(s)}{R(s)}$ as a function of K.

6. For each of the two block diagrams in Figure 9.19:

   (a) Find transfer function $\frac{Y(s)}{R(s)}$.

   (b) Let $A(s) = \frac{1}{s}$, $B(s) = \frac{10}{s+1}$, $C(s) = 2$, $D(s) = \frac{s+0.1}{s+2}$. Find the value of $\frac{Y(s)}{R(s)}$.

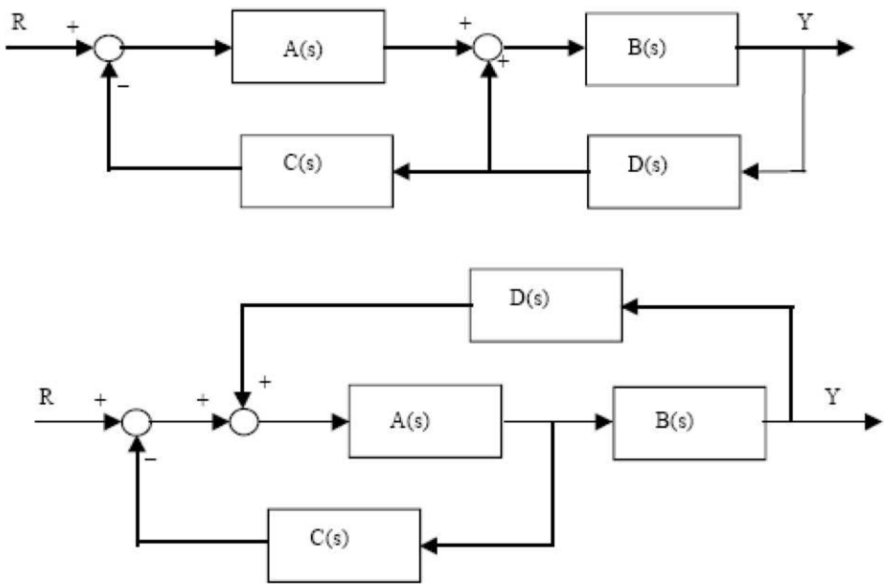

Figure 9.19: Block diagrams of Exercise 6.