Mathematically modelling systems

It is time to take an actual look at systems now. But to do that, we need to have some way to display them. There are several ways to do this. We'll examine these methods in this chapter.

1 Block diagrams

1.1 Block diagram notations

As we now know, systems manipulate variables. They take input variables $u(t)$ and do all sort of stuff with them, to find the output variables $y(t)$. They can, for example, add them up ($y(t) = u_1(t) + u_2(t)$). They can subtract them ($y(t) = u_1(t) - u_2(t)$). They can multiply them ($y(t) = Ku(t)$). They can even integrate them ($y(t) = \int_0^t u(\tau)\, d\tau + y(0)$).

But how can we write/display this? One common-used method to do that, is by using **block diagrams**. Examples of parts of such diagrams are shown in figure 1. As you can see, addition and subtraction of two signals is denoted by a circle with a plus or a minus next to it. Other operations (like multiplying and integrating) are usually noted by a square, with the corresponding factor in it.

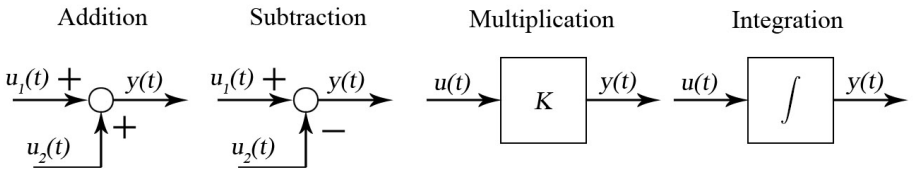

Figure 1: Examples of block diagram parts.

There is, however, a slight problem. Evaluating the integral in a block diagram is often rather troubling. Luckily, there is an easy solution to this problem. We simply examine block diagrams in the Laplace domain. Adding up and subtracting remains unchanged, as well as multiplying by constants. However, we can now simply replace the integral by a factor $1/s$. (Remember that integrating meant dividing by s in the Laplace domain.) That simplifies things quite a lot.

1.2 A basic closed-loop system

Let's examine a basic **closed-loop system**, as is displayed in figure 2. As you can see, it has a so-called **feedback loop**. The system also has an **input signal** $U(s)$ and an **output signal** $Y(s)$. (Note that, since we are now working in the Laplace domain, we use capital letters.)

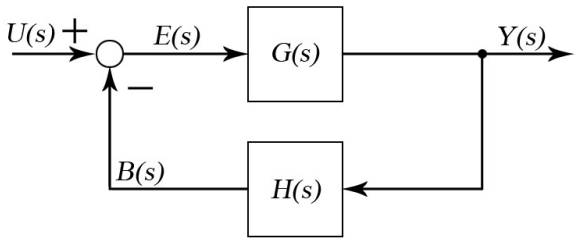

Figure 2: A basic closed loop system.

Let's take a look at what's happening in this system. The system recieves a certain input signal $U(s)$. It compares this with the so-called **feedback signal** $B(s)$. (In fact, it takes the difference.) This gives the **actuating error signal** $E(s)$. This error signal is then passed on to some **control function** $G(s)$, which returns the output $Y(s)$. However, the function $Y(s)$ isn't only outputted. It is also transformed by the function $H(s)$ and fed back to the system as the feedback signal.

1.3 Controller types

In the closed-loop system, there was the control function $G(s)$. Given an error $E(s)$ (or equivalently, $e(t)$), this function determines the output $Y(s)$ (or similarly, $y(t)$). The manner in which it does this, is called the **control action**. We, as designers of the system, are supposed to define this control action. There are several controllers/control actions that are very common. Let's take a look at some of them.

- In the **proportional controller** (the **P-controller**), the output is proportional to the error. This means that $y(t) = K_p e(t)$. In the Laplace domain this becomes $G(s) = Y(s)/E(s) = K_p$. The variable K_p is called the **proportional gain**.
- In the **proportional + integral controller** (the **PI-controller**) we have

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt, \qquad \text{or, in the Laplace domain,} \qquad G(s) = \frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right). \tag{1.1}$$

  The variable T_i is called the **integral time**.
- In the **proportional + derivative controller** (the **PD controller**) we have

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}, \qquad \text{or, in the Laplace domain,} \qquad G(s) = \frac{U(s)}{E(s)} = K_p\left(1 + T_d s\right). \tag{1.2}$$

  Surprisingly, T_d is called the **derivative time**.
- Finally, in the **proportional + integral + derivative controller** (the **PID controller**) we combine the previous two cases. We then have

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt}. \tag{1.3}$$

  In the Laplace domain this of course becomes

$$G(s) = \frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right). \tag{1.4}$$

2 Transfer functions

Performing calculations with block diagrams is rather difficult. That's why we now examine another modelling method: the transfer function.

2.1 Definition of the transfer function

The **transfer function** $F(s)$ of a system is defined as the ratio between the output and the input in the Laplace domain. So,

$$F(s) = \frac{Y(s)}{U(s)}. \tag{2.1}$$

By the way, to find the above transfer function, we do have to assume that all initial conditions are zero.

Let's try to find the transfer function for the system of figure 2. First, we can find that

$$Y(s) = E(s)G(s) = (U(s) - B(s))G(s) = (U(s) - H(s)Y(s))G(s). \tag{2.2}$$

Solving for $Y(s)/U(s)$ gives the rather important **closed-loop transfer function**

$$F(s) = \frac{G(s)}{1 + G(s)H(s)}. \tag{2.3}$$

By the way, from the transfer function, we can also find the **order** of the system. The order of the system is the highest power of s in the denominator of $F(s)$. So if, for example, $F(s) = (a_0+a_1s)/(b_0+b_1s+b_2s^2)$, then the order is 2. (This is due to the b_2s^2 term.)

2.2 Transfer functions of parts

We can also find transfer functions for parts of systems. We once more examine the closed-loop system. There is, for example, the **open-loop transfer function**. This is the transfer function if the loop would not be closed. It is thus

$$\text{open loop transfer function} = \frac{B(s)}{E(s)} = G(s)H(s). \tag{2.4}$$

Similarly, the **feedforward transfer function** is defined as the multiplication of all blocks directly between the input and the output. It is thus

$$\text{feedforward transfer function} = \frac{Y(s)}{E(s)} = G(s). \tag{2.5}$$

2.3 The impulse response function

Another very important function is the **impulse response function** $f(t)$. This is the inverse Laplace transform of the transfer function $F(s)$. (Note that $f(t)$ is not in the Laplace domain, but is an actual function.) To find it, we simply have to find $\mathscr{L}^{-1}(F(s))$.

Why is the impulse response function so important? To find that out, we suppose that the input of the system is only a unit impulse function (thus $u(t) = \delta(t)$). We then have $U(s) = 1$ and thus also $F(s) = Y(s)$. In other words $f(t)$ will then be the output of the system! The impulse response function is thus the output of the system when the input is a unit impulse.

And why is this so important? Well, suppose that we know this input response function. It is then possible (using some mathematical tricks) to find the response (output) of the system for any input function. And that is of course what we want. We will, however, investigate response analysis in later chapters.

2.4 Response of a system to initial conditions

Let's suppose we have a system with a known transfer function $F(s)$. Also the input function $u(t)$ (or equivalently, $U(s)$) is known. We can then determine the output, using the method described above. (Although we will go into greater detail in later chapters.) However, we have made one rather important assumption: the initial conditions are zero. But what should we do if this is not the case? We then write our transfer function $F(s)$ as

$$F(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + b_1s + b_2s^2 + \dots}{a_0 + a_1s + a_2s^2 + \dots}. \tag{2.6}$$

It can be shown that the above equation corresponds to the differential equation

$$b_0u(t) + b_1\dot{u}(t) + b_2\ddot{u}(t) + \ldots = a_0y(t) + a_1\dot{y}(t) + a_2\ddot{y}(t) + \ldots. \tag{2.7}$$

We now assume that the input $u(t)$ is zero (thus also $U(s) = 0$), but that the initial conditions are not zero. This simplifies the above differential equation. If we then again transform it to the Laplace domain, we can find that

$$Y(s) = \frac{a_1y(0) + a_2(sy(0) + \dot{y}(0)) + a_3(s^2y(0) + s\dot{y}(0) + \ddot{y}(0)) + \ldots}{a_0 + a_1s + a_2s^2 + \ldots}. \tag{2.8}$$

Find the inverse Laplace transform of this function and you will know the response $y(t)$ to the initial conditions. By the way, if there are both initial conditions and a non-zero input, you can use the principle of superposition. Simply add both solutions up.

2.5 Transforming a block diagram to a transfer function

Let's suppose we have a system with some very complicated block diagram. How do we find the transfer function? This is, in fact, a rather difficult problem. However, there are a few rules you can use to simplify things.

- When two blocks $G_1(s)$ and $G_2(s)$ are placed in sequence (with no branching points or joining points in between), you can replace them by one block. This block is the multiple $G_1(s)G_2(s)$ of the two blocks.
- When two blocks $G_1(s)$ and $G_2(s)$ are placed parallel to each other (again without branching/joining), and are then joined together, you can replace them by one block. This block then contains their sum $G_1(s) + G_2(s)$ or their difference $G_1(s) - G_2(s)$. (This depends on whether there is a plus or a minus sign at the joining of the blocks.)
- When there is a closed-loop part in the system (as shown in figure 2), you can replace it by one block. This block will then contain $G(s)/(1 + G(s)H(s))$.

It often occurs that you can't simplify your block diagram any further with the above rules. This is because branching/joining points are interfering. When this is the case, you have to move these branching/joining points. However, this often causes the values of blocks to change. You may wonder, what do these blocks change to? Well, there are two rules which you should always keep in mind.

- The product of all blocks around any loop must stay the same.
- The product of all blocks in the feedforward direction (the path from input to output) must stay the same.

By applying these rules, you should be able to simplify the entire block diagram to one block. This block then contains the transfer function of the entire system. And this is exactly what you wanted to find.

3 The state-space representation

A more modern method is the state-space method. Especially computers are rather good at using this method. It is also slightly less abstract than a transfer function, because it directly uses differential equations.

3.1 State variables and the state vector

Let's examine a system. This system has r inputs $u_1(t), \ldots, u_r(t)$ and m outputs $y_1(t), \ldots, y_m(t)$. For this system, we can define a set of n **state variables**. These state variables should completely describe the system. However, there may not be any redundancy: if we remove any of these variables, the system isn't completely defined anymore. If this is the case, then these n variables form the **state** of a system.

Working with n state variables can be a bit troublesome. We therefore usually put them in a vector

$$\mathbf{x}(t) = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^T. \tag{3.1}$$

This vector is the so-called **state vector**. Similarly, we can define an **input vector** $\mathbf{u}(t)$ and an **output vector** $\mathbf{y}(t)$.

You may wonder, how many state variables do we need? This amount n (also called the **order** of the system) is equal to the number of integrators in our system. In fact, the n state variables $x_1, \ldots, x_n$ are the outputs of these integrators. This may sound rather vague at the moment. However, it will be clearer once we deal with a practical example. We will do that soon.

3.2 State-space equations

So, we have these n variables. They aren't just random variables. They are governed by rules. From these rules, we should be able to derive n differential equations, being

$$\dot{x}_i(t) = f_i(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_r, t), \tag{3.2}$$

for every i from 1 to n. We should also be able to derive the outputs $y_i(t)$ from these state variables. So,

$$y_i(t) = g_i(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_r, t), \tag{3.3}$$

for every i from 1 to m. This all seems rather difficult. However, things often simplify a lot. Most of the times the functions f_i and g_i are linear. (We then have a so-called **linear system**.) In this case, we can write

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) \qquad \text{and} \qquad \mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t). \tag{3.4}$$

And things get even better now. For the problems we will consider, the matrices A, B, C and D are constant in time as well. (The system is then **time-invariant**.) So we have

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \qquad \text{and} \qquad \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t). \tag{3.5}$$

The first of these equations is the **state equation**. The second is the **output equation**. The matrices A, B, C and D have names too. A is the $(n \times n)$ **state matrix**, B is the $(n \times r)$ **input matrix**, C is the $(m \times n)$ **output matrix** and D is the $(m \times r)$ **direct transmission matrix**.

3.3 Transforming a state-space system to a transfer function

There is a link between state-space systems and transfer functions. We can transform a state-space system to a transfer function. To do that, we first have to take the Laplace transform of equation (3.5). This gives us

$$s\mathbf{X}(s) - \mathbf{x}(0) = A\mathbf{X}(s) + B\mathbf{U}(s) \qquad \text{and} \qquad Y(s) = C\mathbf{X}(s) + D\mathbf{U}(s). \tag{3.6}$$

From the first equation, we can solve for $\mathbf{X}(s)$. We then insert it into the second one. This gives us

$$\mathbf{Y}(s) = (C(sI - A)^{-1}B + D)\mathbf{U}(s). \tag{3.7}$$

Since $\mathbf{Y}(s) = F(s)\mathbf{U}(s)$, we find that

$$F(s) = C(sI - A)^{-1}B + D. \tag{3.8}$$

Just like D, also $F(s)$ is an $m \times r$ matrix. You may wonder how that is possible. We wanted a transfer function, and now we have a transfer function matrix. That is because we now have multiple inputs and outputs. In fact, the matrix entry $F_{ij}(s)$ gives the transfer function for the output i, due to input j. (This, of course, follows directly from the vector equation $\mathbf{Y}(s) = F(s)\mathbf{U}(s)$.) If our system only has one input and one output, then $F(s)$ will just be a normal transfer function.

You also may have wondered, does the inverse of $(sI-A)$ always exist? In fact, it doesn't. If $\det(sI-A) = 0$, then this inverse does not exist. $F(s)$ is then not defined. From the linear algebra course, we remember that this is only the case when s is an eigenvalue of A. However, from the part on transfer functions we also remember an important fact: $F(s)$ is not defined when s is a pole. What can we conclude from this? It means that the eigenvalues of A are the poles of $F(s)$!

3.4 A practical example of a state-space system

It is time to consider a practical example. Let's consider the system in figure 3. It has one input $u(t)$, being the deflection of the rightmost point. As output, we want to have $\dot{y}_1(t)$ and $y_2(t)$. We want to model this system using the state-space method. To find the state-space system, we must find the matrices A, B, C and D. So how do we do that?

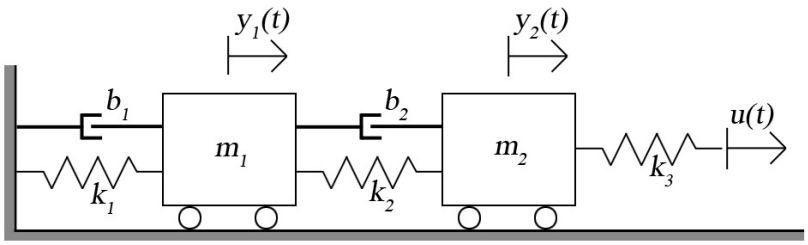

Figure 3: An example system.

The first step is by finding the equation of motion. For the first car we have

$$m_1\ddot{y}_1 = -k_1y_1 - k_2(y_1 - y_2) - b_1\dot{y}_1 - b_2(\dot{y}_1 - \dot{y}_2). \tag{3.9}$$

Sometimes the signs in the above equation get confusing. If this troubles you, you can apply a trick. For example, let's look at the spring with stiffness k_1. Ask yourself, if $y_1 > 0$, will the force due to the spring be to the left or to the right? In this case the force will point to the left. (The spring is stretched out. It therefore pulls at the block.) However, y_1 is positive if it goes to the right. Since the directions are opposite, there must be a minus sign.

Similarly, we can find for the second car that

$$m_2\ddot{y}_2 = -k_2(y_2 - y_1) - b_2(\dot{y}_2 - \dot{y}_1) - k_3(y_2 - u). \tag{3.10}$$

Now we want to put these equations into a state-space system. But what state variables should we define? And how many? Well, we can recall from earlier in this chapter that the amount of state variables was equal to the amount of integrators. How many integrators do we have in this case? We notice that there is a second derivative of $y_1(t)$. So to get $y_1(t)$ itself, we need two integrators. Similarly, we need two integrators for $y_2(t)$. We thus have four integrators. We should therefore define four state variables: x_1, x_2, x_3 and x_4.

But what state variables shall we define? A logical choice would be to define $x_1 = y_1$ and $x_2 = y_2$. But what about the other two variables? Well, you may have noticed that there are second derivatives in our equations. And in the state-space equations there are no second derivatives. How do we deal with this? We simply define $x_3 = \dot{y}_1$ and $x_4 = \dot{y}_2$. This implies that $\dot{x}_3 = \ddot{y}_1$ and $\dot{x}_4 = \ddot{y}_2$. Using these definitions, we can rewrite equations (3.9) and (3.10) to

$$\dot{x}_3 = \frac{-k_1 - k_2}{m_1}x_1 + \frac{k_2}{m_1}x_2 + \frac{-b_1 - b_2}{m_1}x_3 + \frac{b_2}{m_1}x_4, \tag{3.11}$$

$$\dot{x}_4 = \frac{k_2 x_1}{m_2} + \frac{-k_2 - k_3}{m_2}x_2 + \frac{b_2}{m_2}x_3 + \frac{-b_2}{m_2}x_4 + \frac{k_3}{m_2}u. \tag{3.12}$$

Now we are ready to find the state equation. It will be

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-k_1-k_2}{m_1} & \frac{k_2}{m_1} & \frac{-b_1-b_2}{m_1} & \frac{b_2}{m_1} \\ \frac{k_2}{m_2} & \frac{-k_2-k_3}{m_2} & \frac{b_2}{m_2} & \frac{-b_2}{m_2} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k_3}{m_2} \end{bmatrix} \mathbf{u}. \tag{3.13}$$

So we have found A and B! That's great news. Now only C and D remain. To find them, we have to find expressions for the outputs of the system, being $\dot{y}_1(t)$ and $y_2(t)$. This is, in fact, easier than it seems. We simply have $\dot{y}_1 = x_3$ and $y_2 = x_2$. This implies that

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \mathbf{u}. \tag{3.14}$$

And now we have found C and D as well. So this is how you find the state-space equations of a system. Just write down the equations of motion, define your state variables, and find the matrices A, B, C and D.