AEROSPACE SYSTEMS AND CONTROL THEORY SAMMARY

BASED ON LECTURES BY DR IR C DE VISSER

SAM VAN ELSLOO

MAY - JUNE 2017

ナマケモノは生活

Preface

Please note that the first two chapters are rather introductory and you may be wondering what the stuff is useful for, so just keep in mind that it only be in the third chapter that we'll actually start doing something useful.

Furthermore, the summary is obviously not finished yet, and will do so when I have time for it. I don't know when that'll be, however.

Contents

1 Introduction

1.1 Control loops

We distinguish between two control loops, shown in figure 1.1:

- In open-loop control, the controller receives a **reference/input signal** r; the controller produces a **control signal/control input** u, which then leads to an **output signal** y. An example would be a burglar alarm: once it receives its reference signal, i.e. there's an intruder in your house, it gives a control signal to the sound maker, which in turn produces the output signal: a whole lot of noise. However, this output is not used as feedback: once your alarm goes off, it never stops buzzing, not even when the intruder is gone. Other examples would be a simple water boiler, which you give a reference signal on (namely that it should be on), after which heats up the water (the control signal), which in turn produces an output signal (the temperature of the water inside). Again, the water boiler automatically turns off; it then doesn't turn on again if the temperature of the water drops below a certain temperature.
- In closed-loop control, the same process happens; however, this time, the output is compared to the reference signal. Then, based on the computed **control error** ($e = r - y$), a decision is made whether to continue issuing the control signal or not. Take for example your fridge: you manually set it to a certain temperature it should keep, e.g. 4 °C (this is the reference signal). Now, suppose the temperature inside the fridge is first 20 °C, then the fridge will obviously first cool down the system (this is the control signal). This produces a output signal, namely the temperature of the fridge. Initially, this will be temperatures like 18 °C, 15 °C, 12 °C etc. and the controller decides to keep cooling down. After a while, the temperature drops below 4 °C, and since the control error is now smaller than zero, the controller (the thermostat) may decide that it's time to stop cooling down. This will cause the temperature to increase again, after which the controller decides again that it's time to cool down, and then it'll drop below 4 °C again, etc. The contrast with an open-loop system should be clear: whereas a burglar alarm doesn't do anything with the stuff it outputs, a fridge does: it checks the output temperature to decide what it should do.

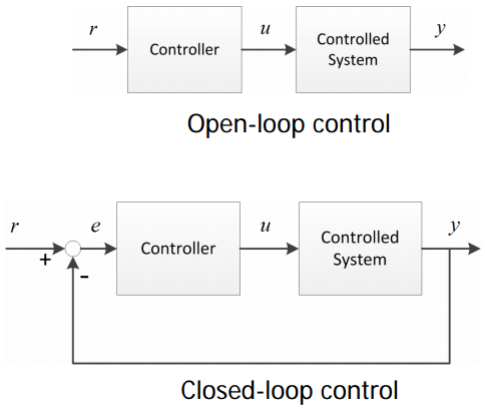


Figure 1.1: Open and closed loop control.

1.2 Control system design

In order to understand the system, we must analyse its properties, but also the properties of its inputs and outputs.

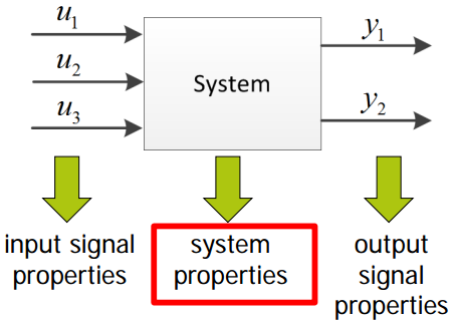


Figure 1.2: Properties.

1.2.1 *System properties*

Static vs. Dynamic systems

Static vs. **dynamic systems** are shown in figure 1.3. Should speak for itself, really.

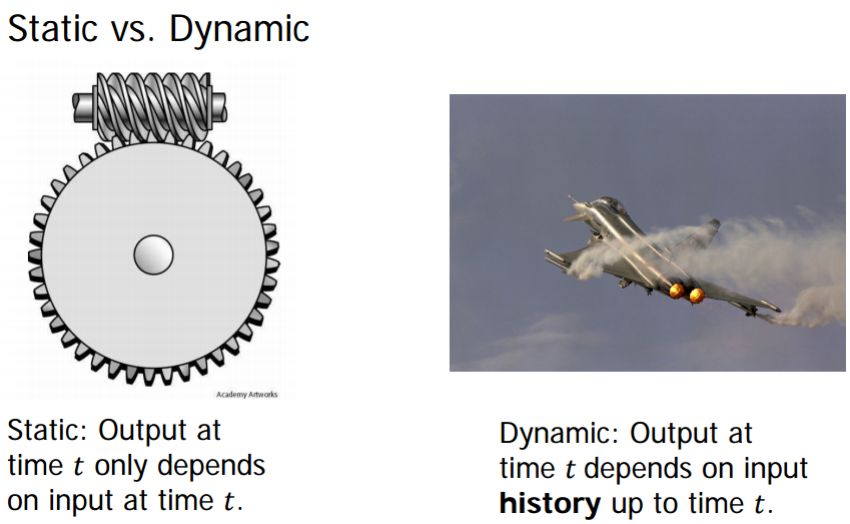


Figure 1.3: Static vs. dynamic systems.

Linear vs. nonlinear in the inputs/outputs

Linear vs. **nonlinear** should be a familiar term. It's shown in figure 1.4, and we'll discuss it in more detail in the next chapter, cause when a differential equation is linear, it does not mean the system is also linear (but we'll get to that later).

Time invariant vs. time varying

Time invariant vs. **time varying** is shown in figure 1.5. Again, should be pretty obvious.

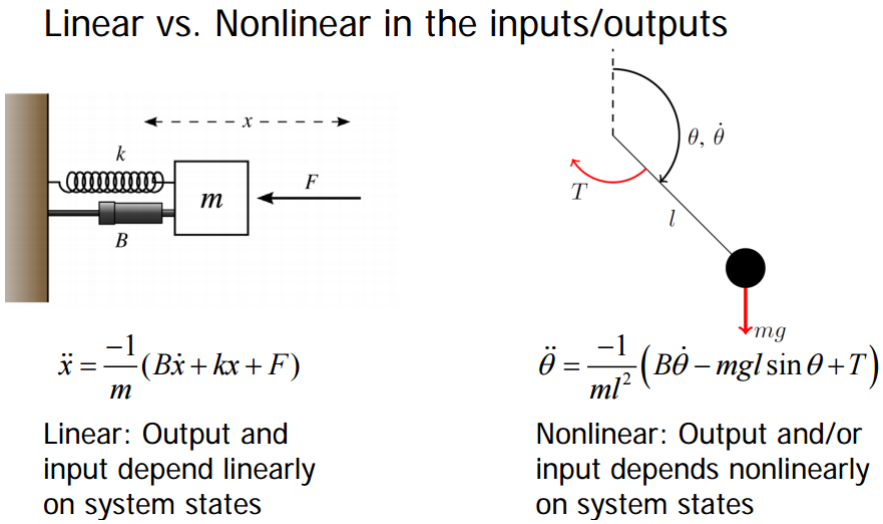


Figure 1.4: Linear vs. non-linear systems.

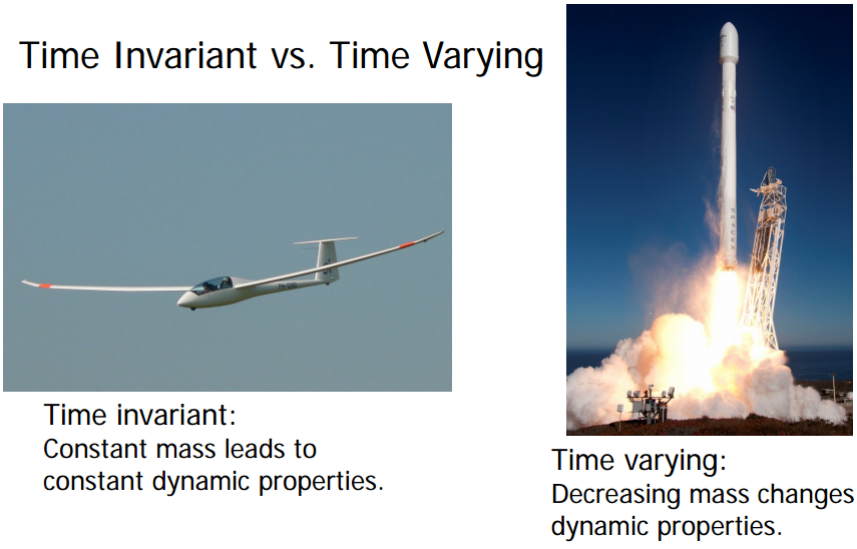


Figure 1.5: Time invariant vs. time varying systems.

1.2.2 *Signal properties*

Continuous vs. discrete

Signals can be **continuous** or **discrete**, as shown in figure 1.6. Continuous signals have a value for every time t; discrete signals only have it at certain times.

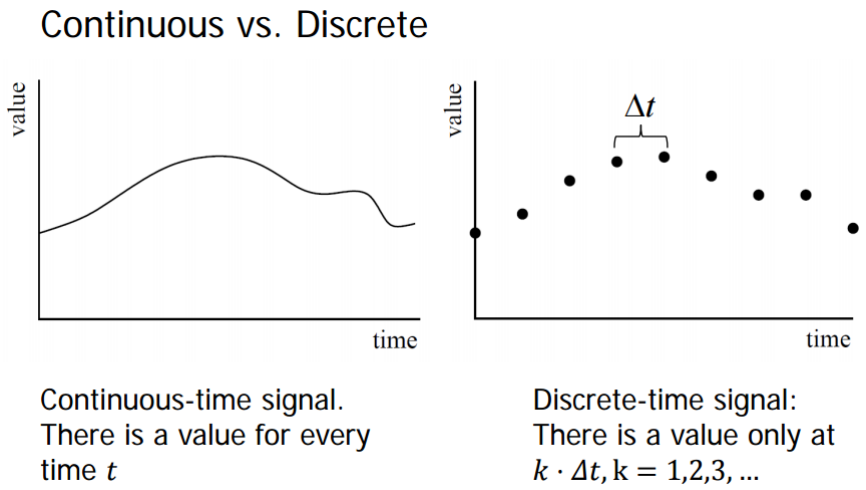


Figure 1.6: Continuous vs. discrete signals.

Stochastic vs. deterministic

For **stochastic signals**, the signal is totally random. For **deterministic signals**, the signal is a known function of time, as shown in figure 1.7.

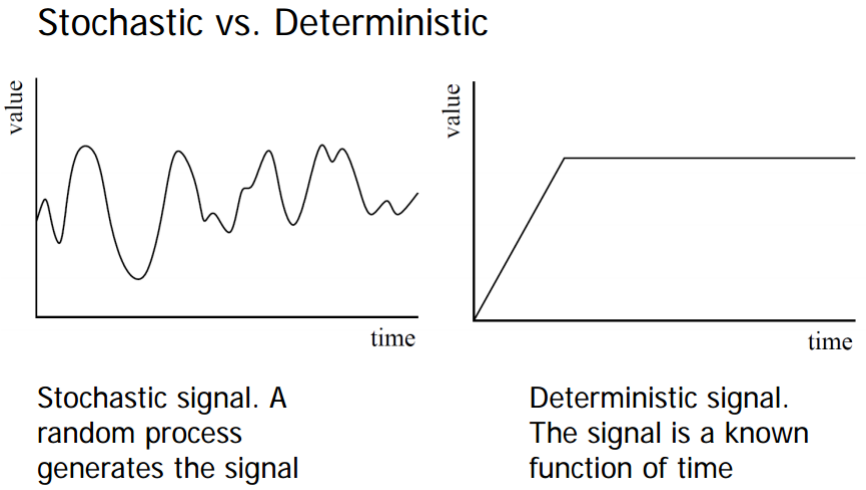


Figure 1.7: Stochastic vs. deterministic signals.

Analogue vs. digital

Analogue signals can assume any value, **digital signals** only assume certain values, as shown in figure 1.8.

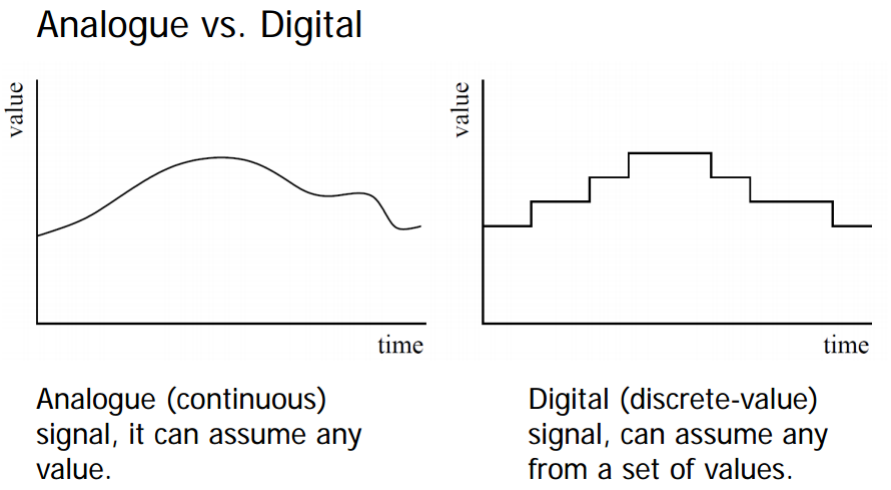


Figure 1.8: Analogue vs. digital signals.

A-periodic vs. periodic

As shown in figure 1.9, **periodic signals** are periodic, **a-periodic** aren't.

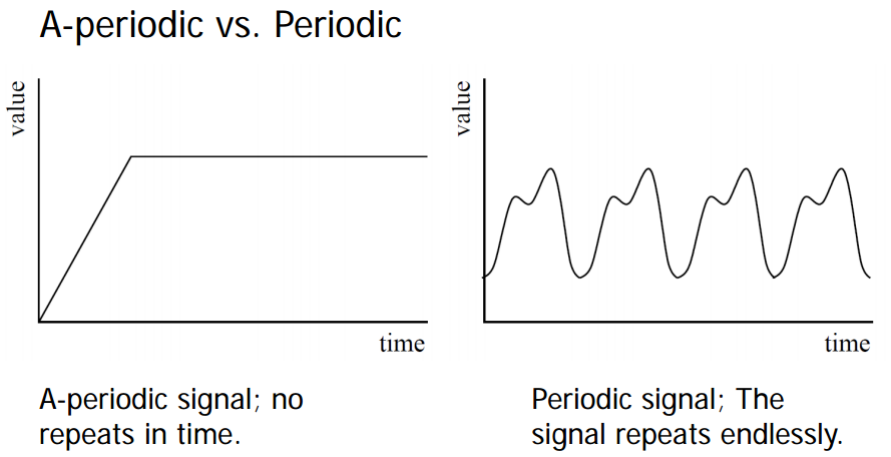


Figure 1.9: A-periodic vs. periodic signals.

1.3 Conclusion

In this course, we focus on linear time invariant system (LTI) models, as they are the simplest. Furthermore, we use easy input and output signals: continuous and deterministic, and we'll discuss both periodic and a-periodic signals.

2 The Laplace Transform and Transfer Functions

Yeah you had most of this already, so I'll only discuss it shortly. Don't worry too much about application, we'll see it extensively in use in the third chapter.

2.1 Linear time-invariant system properties

Before we do Laplace, it is important to realize what exactly classifies as a linear time-invariant system, because those are the only systems we can apply the subsequent mathematical methods on. Linear time-invariant systems have three properties:

2.1.1 Homogeneity

First of all, the solution needs to be homogeneous: if the input is doubled, then the output also needs to be doubled, as shown in figure 2.1.

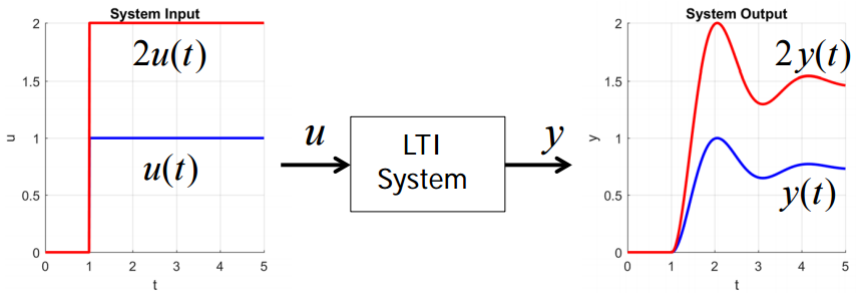


Figure 2.1: Homoegneity property.

2.1.2 Superposition

Second of all, the system needs to allow for superposition: a system input $A + B$ must lead to the same output as the output of A and B individually, as shown in figure 2.2.

2.1.3 Time invariance

If you shift an input a time T, the output is is also shifted a time T, as shown in figure 2.3.

Now, these all should be pretty logical. Nevertheless, be aware of how quickly you call something linear time-invariant or not. Suppose we take a throwback to the vibrations course, and we have the following equation of motion, corresponding to a mass-damper system at which a constant force F is applied:

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = F$$

Now, the solution to this problem is

$$x(t) = c_1 e^{-\zeta\omega_n t}\sin\left(\omega_d t\right) + c_2 e^{-\zeta\omega_n t}\cos\left(\omega_d t\right) + F$$

Note that this system is not linear, even though the differential equation is linear! When we double F (this is the only input in the system; the other symbols are all system properties), we don't double $x(t)$ as well, clearly.

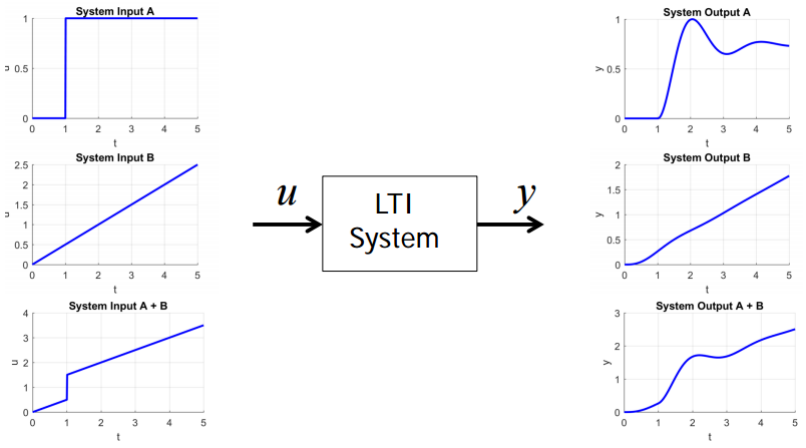


Figure 2.2: Superposition property.

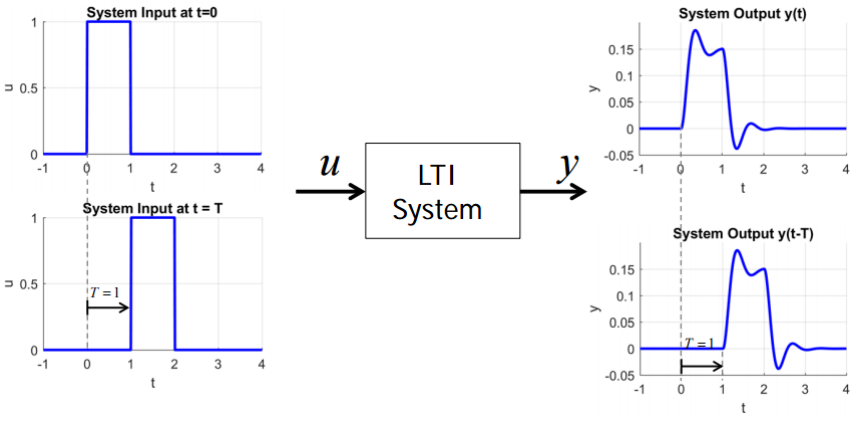


Figure 2.3: Time invariance property.

Only if the initial conditions are $x(0) = 0$ and $\dot{x}(0) = 0$, then we get the following integration constants c_1 and c_2:

$$x(0) = c_1 e^{-\zeta\omega_n \cdot 0} \sin(\omega_d \cdot 0) + c_2 e^{-\zeta\omega_n \cdot 0} \cos(\omega_d \cdot 0) + F = c_2 + F = 0$$

so that $c_2 = -F$. We then have:

$$\begin{aligned}
x(t) &= c_1 e^{-\zeta\omega_n \cdot t} \sin(\omega_d \cdot t) + F\left(1 - e^{-\zeta\omega_n t} \cos(\omega_d t)\right) \\
\dot{x}(t) &= c_1\left(\omega_d e^{-\zeta\omega_n \cdot t} \cos(\omega_d \cdot t) - \zeta\omega_n e^{-\zeta\omega_n \cdot t} \sin(\omega_d \cdot t)\right) \\
&\quad + F\left(\zeta\omega_n e^{-\zeta\omega_n t} \cos(\omega_d t) + \omega_d e^{-\zeta\omega_n t} \sin(\omega_d t)\right) \\
\dot{x}(0) &= c_1\left(\omega_d e^{-\zeta\omega_n \cdot 0} \cos(\omega_d \cdot 0) - \zeta\omega_n e^{-\zeta\omega_n \cdot 0} \sin(\omega_d \cdot 0)\right) \\
&\quad + F\left(\zeta\omega_n e^{-\zeta\omega_n \cdot 0} \cos(\omega_d \cdot 0) + \omega_d e^{-\zeta\omega_n \cdot 0} \sin(\omega_d \cdot 0)\right) \\
&= c_1 + F\zeta\omega_n = 0
\end{aligned}$$

so that $c_2 = -\frac{F}{\zeta\omega_n}$. Thus, the output becomes

$$x(t) = F\left(1 - \frac{1}{\zeta\omega_n} e^{-\zeta\omega_n t} \sin(\omega_d t) - e^{-\zeta\omega_n t} \cos(\omega_d t)\right)$$

which *is* linear: if F is doubled, then so is $x(t)$.

So, note that it is required that the system is initially at rest to make it linear time-invariant (and thus a linear differential equation is not necessarily linear time-invariant, although a non-linear differential equation is automatically not linear time-invariant). This is also a requirement to apply the convolution integral, from which the Laplace transform is derived (well actually not but in this course Laplace transform is derived from convolution integral); in other words, we'll assume in this course, unless otherwise stated, that the initial conditions are zero.

2.2 *Laplace transform*

Okay so suppose we have the following linear differential equation:

$$a\ddot{y} + b\dot{y} + c = u(t)$$

where $u(t)$ is an input function (e.g. $u(t) = \cos(t)$ and $y(0) = \dot{y}(0) = 0$. How did we do Laplace for this? Well, the left side of the equation would simply become

$$aY(s)s^2 + bY(s)s + cY(s) = U(s)$$

where you looked up $U(s)$ from a table of Laplace transforms; an example of such a table is shown in figure 2.4. You'd then rewrite it to

$$Y(s) = \frac{U(s)}{as^2 + bs + c}$$

and you'd do some partial fractions and you'd look up the inverse transforms. All of this should sound rather familiar, and all of this still works in this course, of course.

Now, there's a rather important point of attention. Note that we basically get

$$Y(s) = U(s)H(s)$$

where I define

TRANSFER FUNCTION

The **Transfer function** $H(s)$ is given by

$$H(s) = \frac{1}{as^2 + bs + c} = \frac{Y(s)}{U(s)} \tag{2.1}$$

The **input function** is denoted by $U(s)$. The **output function** is denoted by $Y(s)$.

$f(t) = \mathcal{L}^{-1}\{F(s)\}$	$F(s) = \mathcal{L}\{f(t)\}$	Notes
1. 1	$\frac{1}{s}, \quad s > 0$	Sec. 6.1; Ex. 4
2. e^{at}	$\frac{1}{s-a}, \quad s > a$	Sec. 6.1; Ex. 5
3. t^n, $n =$ positive integer	$\frac{n!}{s^{n+1}}, \quad s > 0$	Sec. 6.1; Prob. 31
4. t^p, $p > -1$	$\frac{\Gamma(p+1)}{s^{p+1}}, \quad s > 0$	Sec. 6.1; Prob. 31
5. $\sin at$	$\frac{a}{s^2+a^2}, \quad s > 0$	Sec. 6.1; Ex. 7
6. $\cos at$	$\frac{s}{s^2+a^2}, \quad s > 0$	Sec. 6.1; Prob. 6
7. $\sinh at$	$\frac{a}{s^2-a^2}, \quad s > \lvert a\rvert$	Sec. 6.1; Prob. 8
8. $\cosh at$	$\frac{s}{s^2-a^2}, \quad s > \lvert a\rvert$	Sec. 6.1; Prob. 7
9. $e^{at}\sin bt$	$\frac{b}{(s-a)^2+b^2}, \quad s > a$	Sec. 6.1; Prob. 13
10. $e^{at}\cos bt$	$\frac{s-a}{(s-a)^2+b^2}, \quad s > a$	Sec. 6.1; Prob. 14
11. $t^n e^{at}$, $n =$ positive integer	$\frac{n!}{(s-a)^{n+1}}, \quad s > a$	Sec. 6.1; Prob. 18
12. $u_c(t)$	$\frac{e^{-cs}}{s}, \quad s > 0$	Sec. 6.3
13. $u_c(t)f(t-c)$	$e^{-cs}F(s)$	Sec. 6.3
14. $e^{ct}f(t)$	$F(s-c)$	Sec. 6.3
15. $f(ct)$	$\frac{1}{c}F\left(\frac{s}{c}\right), \quad c > 0$	Sec. 6.3; Prob. 25
16. $\int_0^t f(t-\tau)g(\tau)\,d\tau$	$F(s)G(s)$	Sec. 6.6
17. $\delta(t-c)$	e^{-cs}	Sec. 6.5
18. $f^{(n)}(t)$	$s^nF(s) - s^{n-1}f(0) - \cdots - f^{(n-1)}(0)$	Sec. 6.2; Cor. 6.2.2
19. $(-t)^n f(t)$	$F^{(n)}(s)$	Sec. 6.2; Prob. 29

Figure 2.4: Laplace table of transforms.

Note that $H(s)$ only depends on the system properties! If you think back to vibrations, if we'd have

$$m\ddot{y} + c\dot{y} + ky = u(t)$$

then we'd have

$$H(s) = \frac{1}{ms^2 + cs + k}$$

which is only dependent on the properties of the system, and not on the input or output! This is also the significance of

$$Y(s) = \frac{U(s)}{H(s)}$$

Basically, the input $U(s)$ is transferred via multiplication by $H(s)$ to the output $Y(s)$.

Example 1

Consider the differential equation

$$\ddot{y} + 4\dot{y} + 3y = u$$

1. Convert this equation to a transfer function
2. Calculate the response to a scaled Dirac impulse with a size of 2 in the Laplace domain.
3. Convert this response to the time domain, using partial fraction expansion & the Laplace transform $e^{-at} \to \frac{1}{s+a}$.

For 1., we have

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

For 2., the response to a scaled Dirac impulse with size of 2 is simply 2.[a]. Thus, we get

$$Y(s) = U(s)\,H(s) = \frac{2}{s^2 + 4s + 3}$$

We can solve this to partial fractions as follows:

$$Y(s) = \frac{2}{s^2 + 4s + 3} = \frac{2}{(s+3)(s+1)} = 2 \cdot \left(\frac{a}{s+3} + \frac{b}{s+1}\right)$$

We must thus have

$$\begin{aligned} a(s+1) + b(s+3) &= 1 \\ as + a + bs + 3b &= 1 \end{aligned}$$

yielding the system of equations

$$\begin{aligned} s(a+b) &= 0 \\ a + 3b &= 1 \end{aligned}$$

The first one leads to $a = -b$, the second one then leads to

$$-b + 3b = 2b = 1$$

so that $b = \frac{1}{2}$ and $a = -\frac{1}{2}$. This leads to

$$Y(s) = 2 \cdot \left(\frac{-\frac{1}{2}}{s+3} + \frac{\frac{1}{2}}{s+1}\right) = \frac{1}{s+1} - \frac{1}{s+3}$$

For 3., the inverse transform then becomes

$$Y(s) = e^{-t} + e^{-3t}$$

[a] Look at figure 2.4: Dirac impulse with size of 2 uses line 17 of the table; the transform of $\delta(t)$ is simply $e^{-0s} = 1$, but since its of size 2, we simply multiply by 2.

Finally, because these are terms we'll come across rather often:

UNIT IMPULSE AND UNIT STEP FUNCTION

The **unit impulse function** is the Dirac delta function; it causes an impulse of magnitude 1 (e.g. when looking at a mass-spring system, an impulse of 1 Ns). The Laplace transform of it is equal to 1 if it occurs at $t = 0$, and in general, the Laplace transform of $\delta(t - c)$ is e^{-cs}.

The **unit step function** is the 'switch' function; a unit step function located at $t = 0$ has Laplace transform $\frac{1}{s}$, and in general, the Laplace transform of a unit step function with its switch at $t = c$ is equal to $\frac{e^{-cs}}{s}$ (assuming there's no function behind it).

3 Block Diagrams of Open and Closed Loop Systems

3.1 What is a block diagram?

3.1.1 Introduction to block diagrams

Block diagrams are a powerful tool to visualize and analyse complex dynamic systems: we can visualize the system $y = A \cdot u$ as shown in figure 3.1.

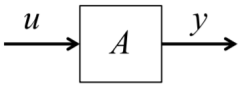


Figure 3.1: Simple block diagram.

We can complicate this further; $z = A \cdot B \cdot u + C \cdot x$ as shown in figure 3.2. Reading this diagram backwards would also give $z = \theta + \psi$, with $\theta = B \cdot y = B \cdot A \cdot u$ and $\psi = C \cdot x$. Note that we used a summation point, and we have two plusses at the connecting point (where ψ and θ connect), thus both are positive.

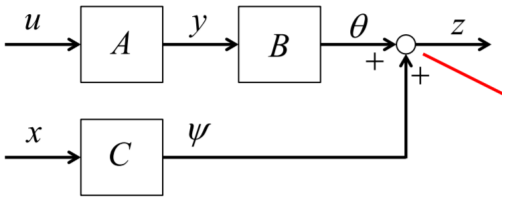


Figure 3.2: Slightly less simple block diagram.

Now, we can complicate it even further, as shown in figure 3.3: we still have $z = \theta + \psi$, but now $\theta = \int y\,dt = \int A \cdot u\,dt$, and $\psi = \frac{d}{dt}x$, so that we have

$$z = \int A \cdot u\,dt + \frac{dx}{dt}$$

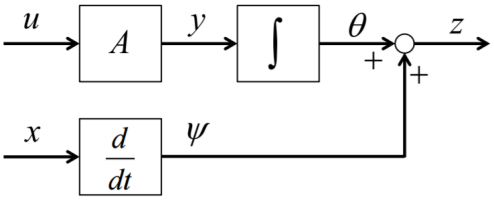


Figure 3.3: Even less simple block diagram.

The fun doesn't stop here, because we can transform the block diagram to the Laplace domain, as shown in figure 3.4. What would then be the output?

$$\begin{aligned} z(s) &= \theta(s) + \psi(s) \\ \theta(s) &= \frac{1}{s} \cdot A(s)\,u(s) \\ \psi(s) &= sx(s) \\ z(s) &= \frac{1}{s} \cdot A(s)\,u(s) + sx(s) \end{aligned}$$

It should be clear that one then could find the inverse transform of this to find $z(t)$. Furthermore, please bear in mind:

- Integration corresponds to multiplication with $\frac{1}{s}$.
- Differentiation corresponds to multiplication with s.

Honestly, don't bother too much as to why this is true.

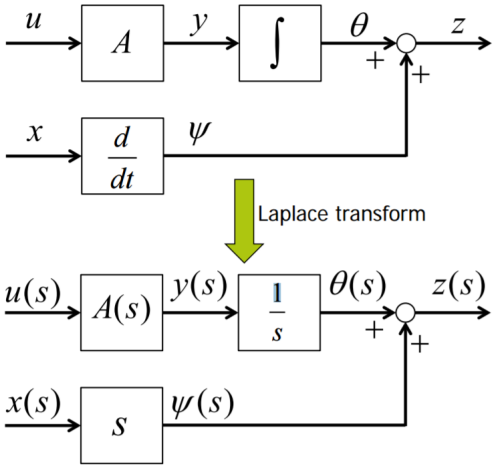


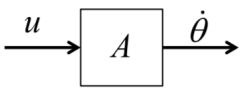
Figure 3.4: Laplace transformed block diagram.

3.1.2 *Examples of block diagrams*

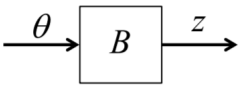
Now, what can we do with this? Well, most importantly, it allows us to relatively easily solve coupled equations. Suppose we have the system of coupled equations:

$$\begin{aligned} \dot{\theta} &= A \cdot u \\ z &= B \cdot \theta \end{aligned}$$

These can be visualized with block diagrams as shown in figure 3.5a and **??**. Now, note that to get from $\dot{\theta}$ to θ, we need to integrate. Thus, we could combine the systems to what is shown in figure, also including the Laplace transforms 3.6.



(a) Subsystem 1: $\dot{\theta} = A \cdot u$.



(b) Subsystem 2: $z = B \cdot \theta$.

Figure 3.5: System of coupled equations.

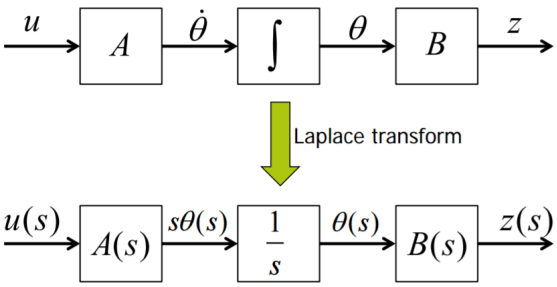


Figure 3.6: Combined block diagram.

This straightforwardly leads to

$$z(s) = B(s) \cdot \frac{1}{s} \cdot A(s) \cdot u(s)$$

From this, the transfer function, defined as the output $z(s)$ divided by the input $u(s)$, easily follows:

$$\frac{z(s)}{u(s)} = \frac{A(s)\,B(s)}{s}$$

You may wonder, is the first result not more useful? Then we can find $z(t)$ rather straightforwardly. Yes, that's true, but we'll soon see that in more complicated systems, the transfer function is a very important thing.

Let's now do a more complicated example. Unfortunately, there are some rather large errors in the derivation of the differential equations which basically make them wrong, but as the lectures use these wrong equations, I'll continue working with them anyway (as it's impossible for me to change everything). However, if you during the derivation of the equations think, but hmm isn't this wrong, then yes it probably is.

Example 1

Suppose we have the plane on the runway as shown in figure 3.7, where:

- V is the velocity of the main landing gear;
- l the wheelbase of the airplane (distance between NLW and MLG);
- y the horizontal distance between the middle of the runway and the main landing gear;
- ψ the angle between runway center line and the line connecting the main landing gear and nose landing gear.
- δ_w the steering angle; i.e. the angle between the line connecting the main landing gear and nose landing gear, and the direction the nose landing gear points in.

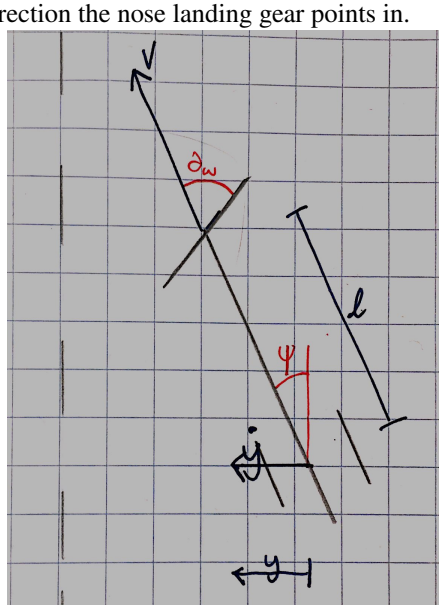
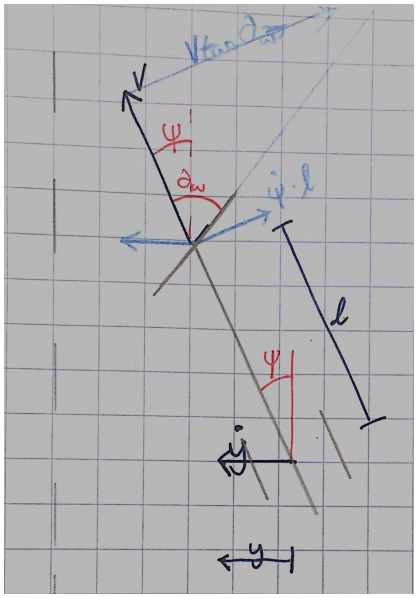


Figure 3.7: Sketch of situation.

First, we want to find the differential equations governing this problem. We do this by making the sketch as shown below.



If you don't see what we did: we placed the velocity vector of the main landing gear at the nose landing gear (yes I'm pretty sure that's not really acceptable). The velocity component in horizontal direction, equal to $\dot{y}$, is then $V \sin\psi$. Furthermore, the velocity vector at the nose landing gear points in the same direction as the direction in which the nose landing gear points, as there should be no slip (yes this sentence is a imperial horseton of bullcrap). The component of the velocity perpendicular to the aircraft center line is then equal to $V \tan\delta_w$, which should also be equal to $\dot{\phi} \cdot l$ (the angular velocity of the nose landing gear around the main landing gear). This leads to the following equations, which can be linearised as:

$$\begin{aligned} \dot{\psi} &= \frac{V}{l}\tan\left(\delta_w\right) \approx \frac{V}{l}\delta_w \\ \dot{y} &= V\sin\left(\psi\right) \approx V\psi \end{aligned}$$

If you don't really get the derivation for these, honestly don't bother with them, I don't think we should be able to do them ourselves.
Now, what can we do with them? We can write this as two subsystems, as shown in figure 3.8.

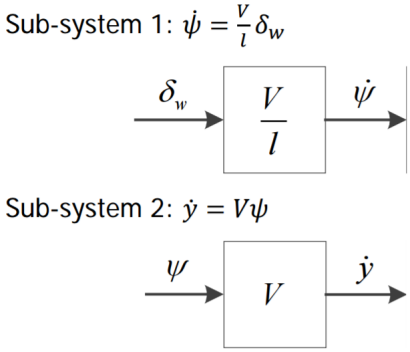


Figure 3.8: Block diagrams.

Now, we can combine these two subsystems: ψ is the integration of $\dot{\psi}$, thus we can extend the system to as shown in figure 3.9. Remember that $1/s$ corresponds with integration.

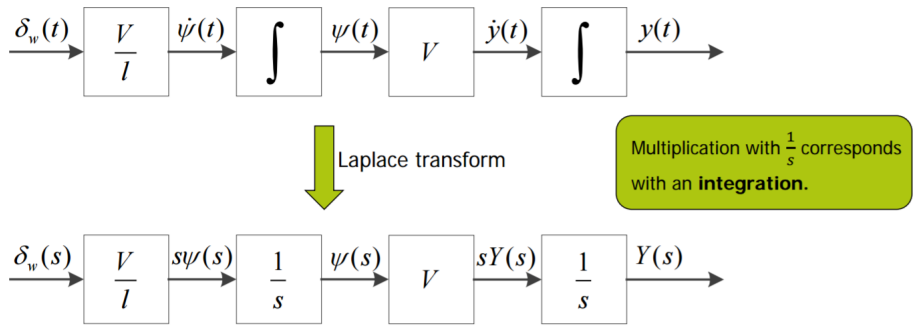


Figure 3.9: Block diagrams combined, including Laplace transform.

This means that we get

$$Y(s) = \frac{1}{s} \cdot V \cdot \frac{1}{s} \cdot \frac{V}{l} \cdot \delta_w(s)$$

Or, that the transfer function becomes

$$\frac{Y(s)}{\delta_w(s)} = \frac{1}{s^2}\frac{V}{l}$$

Note that we can now draw it as a very simple block diagram, as shown in figure 3.10.

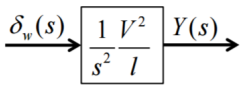


Figure 3.10: Equivalent block diagram.

Now, suppose we have an unit impulse as input; then $U(s) = 1$ (i.e. a Dirac delta function[1]). In that case, we get

$$Y(s) = U(s)\,H(s) = 1 \cdot \frac{1}{s^2} \cdot \frac{V^2}{l}$$

Comparing with the transform $t^n \rightarrow \frac{n!}{s^{n+1}}$, we see that we have $n = 1$ and thus[2]

$$y(t) = t \cdot \frac{V^2}{l}$$

Alternatively, if we an unit step[3], for which the transformation is $U(s) = \frac{1}{s}$, we get

$$Y(s) \quad = \quad U(s)\,H(s) = \frac{1}{s} \cdot \frac{1}{s^2} \cdot \frac{V^2}{l} = \frac{1}{s^3}\frac{V^2}{l} = \frac{1}{2} \cdot \frac{2}{s^3}\frac{V^2}{l}$$

We recognize the same transformation, but now with $n = 2$; thus, we get

$$y(t) = \frac{1}{2} \cdot t^2 \cdot \frac{V^2}{l}$$

3.2 *Block diagrams of closed loop systems*

Now, let's go to something more fancy, and let's include some control: a runway position autopilot. The loop then becomes closed, as shown in figure 3.11: the output value of the lateral position $Y(s)$ is compared to the reference value of it $Y_{ref}(s)$, leading to an error $e(s)$. The box with the K then relates the error to the wheel deflection $\delta_w(s)$ by scaling it with a constant K (which is a parameter you may choose yourself). This means that instead of the wheel deflection angle being the input, the control reference position is the input.

[1] Physically, this means in this case that you give a strong steer to the wheel to steer it to the left (or right), then release the wheel again.

[2] Note that this solution makes sense: suppose you're driving a car; if you give a strong twist to the wheel suddenly, then the car will simply turn suddenly to the left, and continue in that direction; thus, the lateral position will vary linearly with time.

[3] Which would correspond to suddenly turning left and keep turning left, keeping the wheels at the same turning angle.

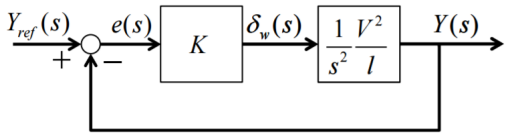


Figure 3.11: Closed loop by including autopilot.

Now, how would this change our equation? Well, note that we'd be getting

$$Y(s) = \frac{1}{s^2}\frac{V^2}{l} \cdot K \cdot e(s)$$

with $e(s) = Y_{ref}(s) - Y(s)$. Plugging this in leads to

$$\begin{aligned} Y(s) &= \frac{1}{s^2}\frac{V^2}{l} \cdot K \cdot \left(Y_{ref}(s) - Y(s)\right) \\ Y(s) + \frac{1}{s^2}\frac{KV^2}{l} \cdot Y(s) &= \frac{1}{s^2}\frac{KV^2}{l} \cdot Y_{ref}(s) \\ \frac{Y(s)}{Y_{ref}(s)} &= \frac{\frac{1}{s^2}\frac{KV^2}{l}}{1 + \frac{1}{s^2}\frac{KV^2}{l}} = \frac{\frac{KV^2}{l}}{s^2 + \frac{KV^2}{l}} \end{aligned}$$

Note that $Y(s)/Y_{ref}(s)$ also defines the transfer function between $Y(s)$ and $Y_{ref}(s)$! Thus, we can replace the entire block, including the feedback, with what is shown in figure 3.12.

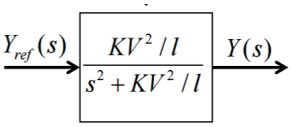


Figure 3.12: Equivalent transfer system.

In other words, we have

$$Y(s) = \frac{\frac{KV^2}{l}}{s^2 + \frac{KV^2}{l}} \cdot Y_{ref}(s)$$

Suppose we now have an unit impulse as $Y_{ref}(s)$, for which the transform is simply 1. We then have

$$Y(s) = \frac{\frac{KV^2}{l}}{s^2 + \frac{KV^2}{l}}$$

You may remember the following Laplace transform from vibrations:

$$\sin(\omega t) \rightarrow \frac{\omega}{s^2 + \omega^2}$$

This means that we have $\omega = \sqrt{\frac{KV^2}{l}}$, and thus

$$\begin{aligned} Y(s) &= \sqrt{\frac{KV^2}{l}} \cdot \frac{\sqrt{\frac{KV^2}{l}}}{s^2 + \frac{KV^2}{l}} \\ y(t) &= \sqrt{\frac{KV^2}{l}} \cdot \sin\left(\sqrt{\frac{KV^2}{l}} \cdot t\right) \end{aligned}$$

which is an undamped periodic motion! This is obviously not desirable, at this means our lateral position will always be oscillating.

Now, fortunately for you, there's an easy way to find equivalent transfer functions for feedback loops. Looking at figure 3.13, we have:

EQUIVALENT TRANSFER SYSTEM FOR FEEDBACK LOOPS

Let G_1 denote the transfer function of the feed-forward path, and G_2 the feedback path. Then

$$\frac{Y(s)}{Y_{ref}(s)} = \frac{G_1(s)}{1 + G_1(s)G_2(s)} = \frac{\text{Feed forward path}}{1 + \text{Feedback path}} \tag{3.1}$$

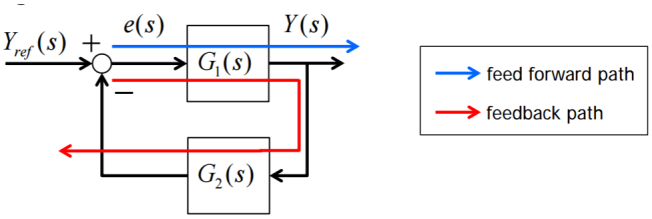


Figure 3.13: Equivalent transfer system for feedback.

It is important that you remember this formula and understand well how to apply it. Look at figure 3.14. The feedforward path is then the path directly from $Y_{ref}(s)$ to $Y(s)$, i.e. $G_3(s)G_1(s)$. The feedback path is only $G_1(s)G_2(s)$ (you start at the summation point and end there again). Thus, we have

$$\frac{Y(s)}{Y_{ref}(s)} = \frac{G_3(s)G_1(s)}{1 + G_1(s)G(2)}$$

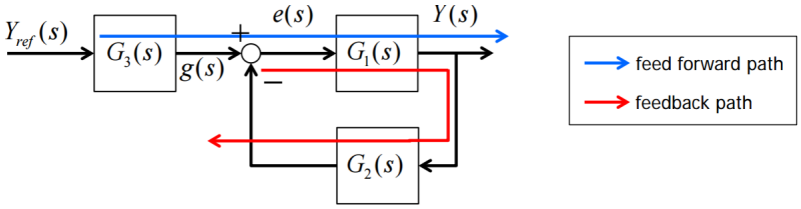


Figure 3.14: Equivalent transfer system for feedback for a more complicated system.

Let's do some more examples.

Example 2

Suppose we have the system shown in figure 3.15.

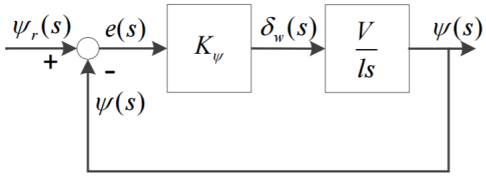


Figure 3.15: System.

Calculate:

1. $\frac{\psi(s)}{\psi_r(s)}$
2. $\frac{e(s)}{\psi_r(s)}$
3. $\frac{\delta_w(s)}{\psi_r(s)}$

Well, let's do them simply one by one. For 1., we have:

- The feed forward path between $\psi(s)$ and $\psi_r(s)$ is $K_\psi \cdot \frac{V}{ls}$.
- The feedback path that's included is equal $K_\psi \cdot \frac{V}{ls}$.

Thus, we get

$$\frac{\psi(s)}{\psi_r(s)} = \frac{K_\psi \cdot \frac{V}{ls}}{1 + K_\psi \cdot \frac{V}{ls}} = \frac{K_\psi \cdot V/l}{s + K_\psi V/l}$$

For 2., we have:

- The feed forward path from $\psi_r(s)$ to $e(s)$ is nothing (i.e. it is equal to 1).
- The feedback path that's included is equal to $K_\psi \cdot \frac{V}{ls}$.

Thus, we get

$$\frac{e(s)}{\psi_r(s)} = \frac{1}{1 + \frac{K_\psi V}{ls}}$$

For 3., we have:

- The feed forward path from $\psi_r(s)$ to $\delta_w(s)$ is K_ψ.
- The feedback path that's included is equal to $K_\psi \cdot \frac{V}{ls}$.

Thus, we get

$$\frac{\delta_w(s)}{\psi_r(s)} = \frac{K_\psi}{1 + \frac{K_\psi V}{ls}}$$

None of this is really difficult; however, do take your time to carefully look at what the paths are as mistakes are easily made.

Now, let's fix our runway position controller. Instead of $Y(s)$, let us be the heading $\psi(s)$ be the output and the reference heading $\psi_{ref}(s)$ be the input. In this case, the block diagram becomes as shown in figure 3.16. If you don't follow why this becomes the block diagram: look back at the example where we derived the differential equations; these were

$$\begin{aligned} \dot{\psi} &= \frac{V}{l}\delta_w \\ \dot{y} &= V\psi \end{aligned}$$

Previously, we combined these two equations as shown in figure 3.9. However, in this case, we will 'stop' the block diagram after the output has become $\psi(s)$, meaning we only get what is shown in figure 3.16 (where the other modifications with respect to 3.9 come from the feedback loop).

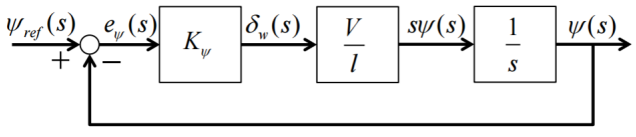


Figure 3.16: Controlling the heading instead of the lateral position.

We then have

$$\frac{\psi(s)}{\psi_{ref}(s)} = \frac{\frac{K_\psi V}{ls}}{1 + \frac{K_\psi V}{ls}} = \frac{K_\psi V/l}{s + K_\psi V/l}$$

or

$$\psi(s) = \psi_{ref}(s) \cdot \frac{K_\psi V/l}{s + K_\psi V/l}$$

Now, suppose we use a step response for $\psi_r(s)$.[4] The transform of that is $\frac{1}{s}$, and thus we get

$$\psi(s) = \frac{1}{s} \cdot \frac{K_\psi V/l}{s + K_\psi V/l} = \frac{a}{s} + \frac{b}{s + K_\psi V/l}$$

Finding a and b:

$$\begin{aligned} \frac{a\left(s + K_\psi V/kl\right) + bs}{s\left(s + K_\psi V/l\right)} &= \frac{K_\psi V/l}{s\left(s + K_\psi V/l\right)} \\ as + bs &= 0 \\ K_\psi V/l \cdot a &= K_\psi V/l \end{aligned}$$

This straightforwardly leads to $a = 1$ and $b = -1$, thus we get

$$Y(s) = \frac{1}{s} - \frac{1}{s + K_\psi V/l}$$

Using the transforms $1 \rightarrow \frac{1}{s}$ and $e^{-at} = \frac{1}{s+a}$, we easily see that $a = K_\psi V/l$ and thus that the solution is

$$y(t) = 1 - e^{-\frac{K_\psi V}{l}t}$$

and thus we have damped aperiodic motion: nice.

3.2.1 *Another controller*

Now, what happens if we introduce another controller, which can generate a reference heading from the lateral deviation: $\psi_r(s) = K_y(Y_r(s) - Y(s))$. Thus, no longer does the pilot have to select the heading angle he wants himself, but he can set the lateral position he wants. First of all, on the output side of the block diagram, we now see some familiar terms appearing again, as shown in figure 3.17: these originate again from figure 3.9 in case you forgot: we now multiply ψ with V to get $\dot{y}$, and then integrate this to get y.

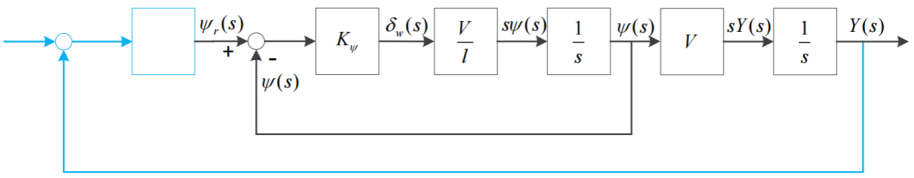


Figure 3.17: Controlling the lateral position as well.

Then the closed-loop straightforwardly becomes as shown in figure 3.18.

Now, how would we find $Y(s)/Y_{ref}(s)$ for this? There are two feedback loops nested in each other, how do we cope with this? Well, look at figure 3.19: for feedback loop in the middle, we already know what the equivalent system was, so we simply substitute that equivalent system in (we derived this literally just before, $\frac{\psi(s)}{\psi_r(s)} = \frac{K_\psi V/l}{s+K_\psi V/l}$).

Then, the feed forward path simply becomes

$$K_y \cdot \frac{K_\psi V/l}{s + K_\psi V/l} \cdot V \cdot \frac{1}{s} = \frac{K_y K_\psi V^2/l}{s^2 + sK_\psi V/l}$$

[4] This corresponds to the following: you're sitting in the cockpit, and you select somewhere on a computer a fixed heading angle ψ_r the plane should follow. This heading angle is constant for the foreseeable future, and naturally, you expect the plane to follow this heading angle after some time.

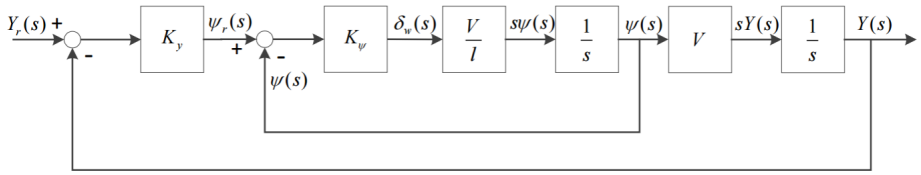


Figure 3.18: Controlling the lateral position as well, with feedback.

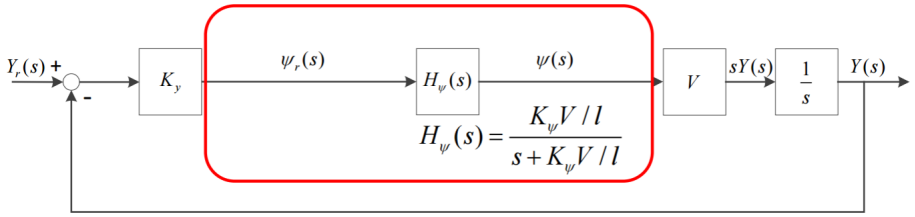


Figure 3.19: Substituting the equivalent system.

The feedback path is

$$K_y \cdot \frac{K_\psi V/l}{s + K_\psi V/l} \cdot V \cdot \frac{1}{s} = \frac{K_y K_\psi V^2/l}{s^2 + sK_\psi V/l}$$

as well, so we get

$$H_y = \frac{Y(s)}{Y_r(s)} = \frac{\frac{K_y K_\psi V^2/l}{s^2+sK_\psi V/l}}{1 + \frac{K_y K_\psi V^2/l}{s^2+sK_\psi V/l}} = \frac{\frac{K_y K_\psi V^2/l}{s^2+sK_\psi V/l}}{\frac{s^2+sK_\psi V/l+K_y K_\psi V^2/l}{s^2+sK_\psi V/l}} = \frac{K_y K_\psi V^2/l}{s^2 + sK_\psi V/l + K_y K_\psi V/l}$$

Now, compare this with vibrations: for a damped-mass-spring system, the transfer function is given by

$$H_y = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Thus, from comparison, we have $\omega_n = \sqrt{K_Y K_\psi V^2/l}$ and $\zeta = \frac{1}{2}\sqrt{K_\psi/(K_Y l)}$. Thus, we have a damped system of which we can change the natural frequency and damping by changing the controller gains K_ψ and K_Y.

In fact, the response to an impulse response can be calculated by comparing it with the transform $e^{-at}\sin(\omega t) \rightarrow \frac{\omega}{(s+a)^2+\omega^2}$ (you need to factorize the denominator): we have

$$\begin{aligned} a &= \frac{K_\psi V}{2l} \\ \omega &= \sqrt{K_y K_\psi V^2/l - \frac{1}{4}K_\psi^2 V^2/l^2} \end{aligned}$$

and thus

$$y(t) = \frac{K_Y K_\psi V^2/l}{\sqrt{K_y K_\psi V^2/l - 0.25K_\psi^2 V^2/l^2}} \cdot e^{-\frac{K_\psi V}{2l}t} \cdot \sin\left(\sqrt{K_Y K_\psi V^2/l - 0.25 \cdot K_\psi^2 V^2/l^2}t\right)$$

I don't expect that you need to be able to do this on the exam (the inverse Laplace transform) for such an ugly transfer function with all those symbolic parameters (it's graded online, and verifying symbolic expressions is pretty hard I think); you have Matlab/Python available to do numerical examples.

3.3 Disturbance inputs

Suppose we'd have a disturbance somewhere, as shown in figure 3.20. What would change?

Well, it's actually pretty simple what you have to do: $Y(s)/Y_{ref}(s)$ stays exactly the same as if $n(s)$ wasn't there. $Y(s)/n(s)$ can also be easily determined using the problem solving guide I gave before:

- The feed forward path to get from $n(s)$ to $Y(s)$ is simply $G_2(s)$.
- The feedback path is included (when we go from $n(s)$, there is a part of the output that goes to feedback, thus we have to include the feedback path) and is equal to $G_2(s)G_3(s)G_1(s)$.

Thus,

$$\frac{Y(s)}{n(s)} = \frac{G_2(s)}{1 + G_3(s)\,G_2(s)\,G_1(s)}$$

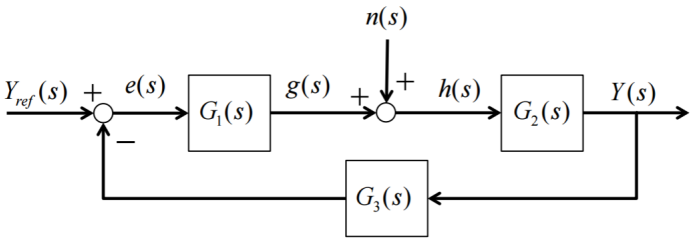


Figure 3.20: Disturbances.

Then, we simply have the equivalent system as shown in figure 3.21. If you don't see it immediately, we'd have

$$Y(s) = \frac{G_2(s)\,G_1(s)}{1 + G_3(s)\,G_2(s)\,G_1(s)} Y_{ref}(s) + \frac{G_2(s)}{1 + G_3(s)\,G_2(s)\,G_1(s)} n(s)$$

for which you could calculate the Laplace inverse transform.

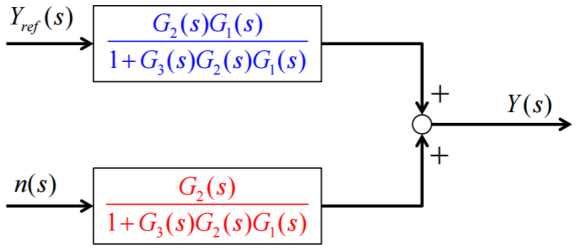


Figure 3.21: Disturbances: equivalent system.

Example 3

Determine the transfer function from $n(s)$ to $Y(s)$ for the disturbed aircraft on the runway, as shown in figure 3.22.

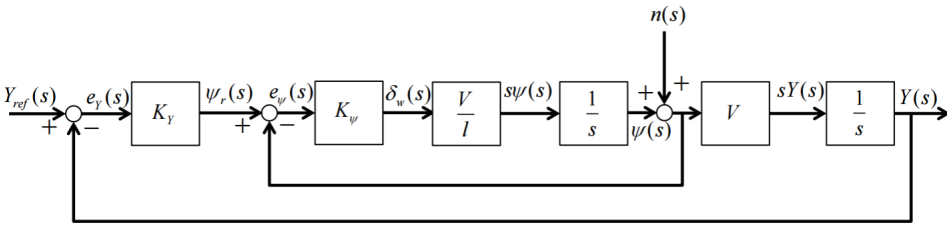


Figure 3.22: System.

First, we have to analyse the feedback loop in the middle. Call the output of this loop $g(s)$. Then the feed forward path to go from $n(s)$ to $g(s)$ is simply equal to 1; the feedback loop is $K_\psi \cdot \frac{V}{l} \cdot \frac{1}{s}$; thus,

$$\frac{g(s)}{n(s)} = \frac{1}{1 + \frac{K_\psi V}{ls}} = \frac{ls}{ls + K_\psi V}$$

However, we must also find the $g(s)/\psi_r(s)$ (we'll quickly see why). There, the feed forward path becomes (the feedback loop stays the same) $K_\psi \cdot \frac{V}{l} \cdot \frac{1}{s}$ so that

$$\frac{\psi_r(s)}{n(s)} = \frac{\frac{K_\psi V}{ls}}{1 + \frac{K_\psi V}{ls}} = \frac{K_\psi V}{ls + K_\psi V}$$

This allows us to redraw the system as shown in figure 3.23.

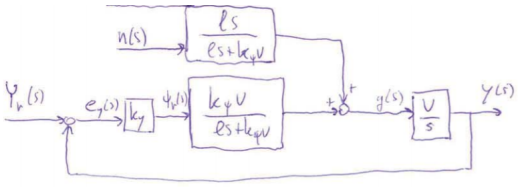


Figure 3.23: Equivalent ystem.

It is then straightforward to calculated $Y(s)/n(s)$. The feed forward path is simply $\frac{ls}{ls+K_\psi V} \cdot \frac{V}{s} = \frac{Vl}{ls+K_\psi V}$. The feedback of is equal to

$$K_y \frac{K_\psi V}{ls + K_\psi V} \frac{V}{s} = \frac{K_y K_\psi V^2}{ls^2 + K_\psi V s}$$

and thus we get

$$\frac{n(s)}{y(s)} = \frac{\frac{Vl}{ls+K_\psi V}}{1 + \frac{K_y K_\psi V^2}{ls^2+K_\psi V s}} = \frac{\frac{Vls}{ls^2+K_\psi V}}{\frac{ls^2+K_\psi V s+K_y K_\psi V^2}{ls^2+K_\psi V s}} = \frac{Vls}{ls^2 + K_\psi V s + K_y K_\psi V^2}$$

3.4 *Manipulation of block diagrams*

There are five basic ways to manipulate block diagrams. The first three are rather obvious. Blocks in series and parallel can easily be combined (these are the first two ways), as shown in figure 3.24.

Furthermore, the equivalent of closed loop elements has already been discussed. That leaves us two new ways of manipulating the diagrams. First of all, we can move branch points, as shown in figure 3.25. Note that the moving makes sense: in the moving back one, you need to ensure that $Y(s)$ stays $X(s)$ multiplied by $G(s)$, thus you must introduce the extra block. Alternatively, when moving it forward, you must divide $X(s)G(s)$ again by $G(s)$ to get $X(s)$.

Similarly, summation points can be moved, as shown in figure 3.26. Again, the moving makes sense: when moving back, you need to keep ensuring that $X_2(s)$ is not multiplied by $G(s)$. Vice versa, when the summation is moved forward, you need to manually multiply $X_2(s)$ by $G(s)$.

Let's do a monster example to see this all in practice. Note that this is way beyond the difficulty of the exam; at the exam, you'll at most see the final version of the autopilot we discussed before (at least that's what he more or less said in the lecture).

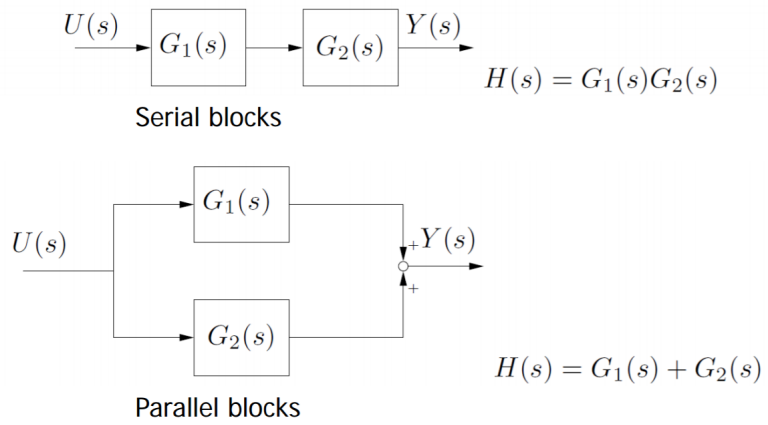


Figure 3.24: Serial and parallel blocks.

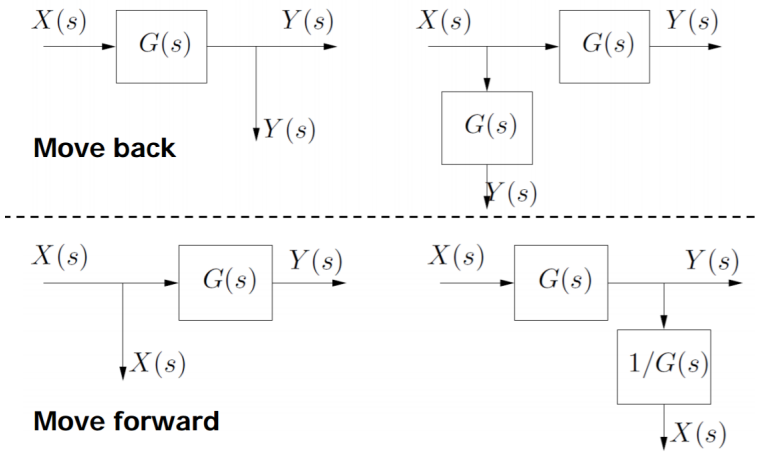


Figure 3.25: Moving branch points.

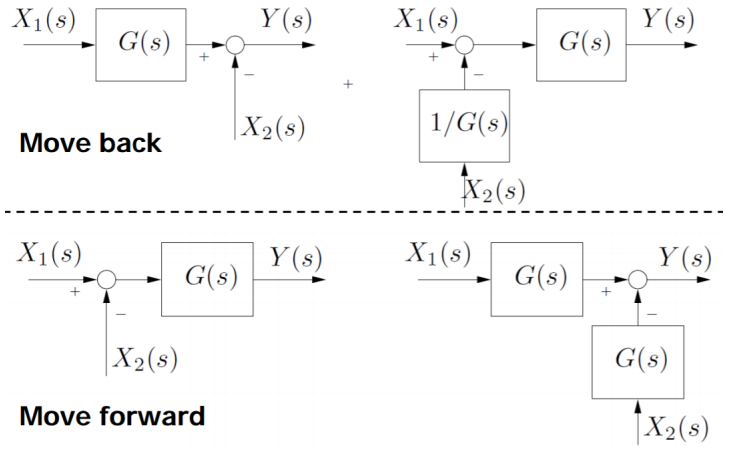


Figure 3.26: Moving branch points.

Example 5

Simplify the block diagram of figure 3.27 to a single block. Enjoy yourself. Please note in general: if two lines are summed with two plusses, then it's simply as if the blocks are in parallel (and then you can simply sum the different blocks). If one of them is plus and the other is minus, it's like a closed loop with feedback, and you have to check the feed forward and feedback paths.

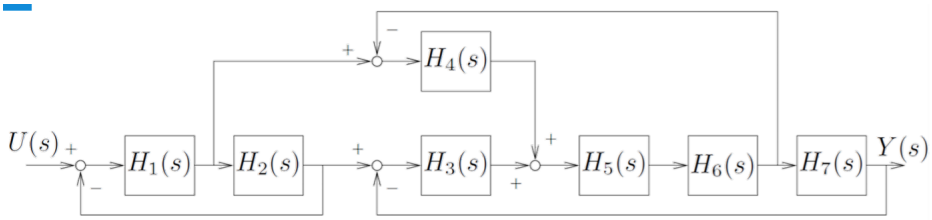


Figure 3.27: Nice block diagram.

Honestly, just follow my lead, it's no shame at all if you didn't come up with all of this yourself. First, we make the changes shown in figure 3.28.

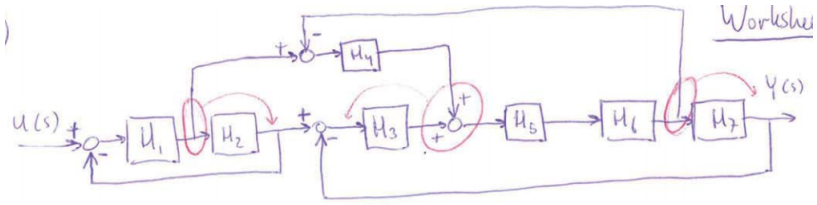


Figure 3.28: Rework I.

What will we changed?

- The branching after H_1 will be moved to beyond H_2; this means that the branch going up to H_4 needs to be divided by H_2. This will allows H_1 and H_2 to be combined to H_1H_2.
- The summation after H_3 needs to occur before H_3; this means that the branch coming from H_4 needs to be divided by H_3.
- The branching after H_6 will be moved to beyond H_7; this means that the branch going to H_4 needs to be divided by H_7.
- H_5 and H_6 will be combined to H_5H_6.

This leads to the block diagram of figure 3.29, where we propose new changes.

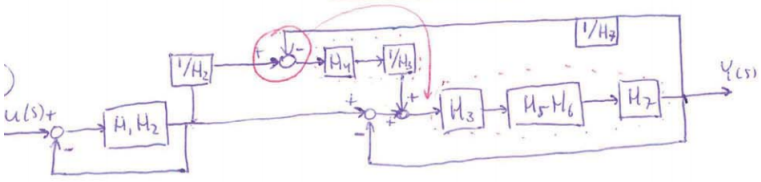


Figure 3.29: Rework II.

What will be changed now?

- The summation in front of H_4 will be moved to in front of H_3; this means that the branch coming from $1/H_7$ needs to be multiplied by H_4/H_3.
- The sequence of blocks $H_4 \rightarrow 1/H_3$ will be combined to H_4/H_3.
- The sequence of blocks $H_3 \rightarrow H_5/H_6 \rightarrow H_7$ will be combined to $H_3H_5H_6H_7$.

This leads to the block diagram of figure 3.30, where we propose new changes.

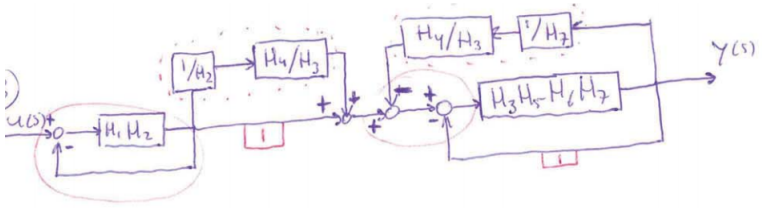


Figure 3.30: Rework III.

What will be changed now?

- The first closed loop will be substituted with

$$\frac{H_1 H_2}{1 + H_1 H_2}$$

- The second closed loop will be substituted with (it is the simple summation of two blocks in parallel, see figure 3.24)

$$1 + \frac{H_4}{H_2 H_3}$$

- The two closed feedback loops over the $H_3 H_5 H_6 H_7$ will be substituted (they are simply two feedback loops in parallel, so you can just sum them) with

$$1 + \frac{H_4}{H_3 H_7}$$

This leads to the block diagram of figure 3.31, where we propose new changes.

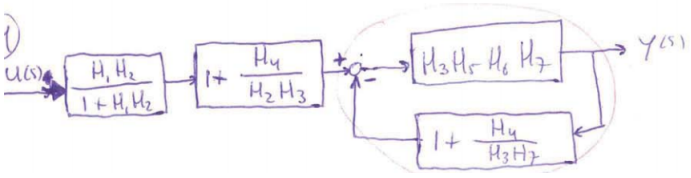


Figure 3.31: Rework IV.

What will be changed now?

- We will replace the closed loop at the end with

$$\frac{H_3 H_5 H_6 H_7}{1 + \left(1 + \frac{H_4}{H_3 H_7}\right) H_3 H_5 H_6 H_7}$$

- We will rewrite the second block as

$$\frac{H_2 H_3 + H_4}{H_2 H_3}$$

This leads to the block diagram of figure 3.31, where we can start combining everything.

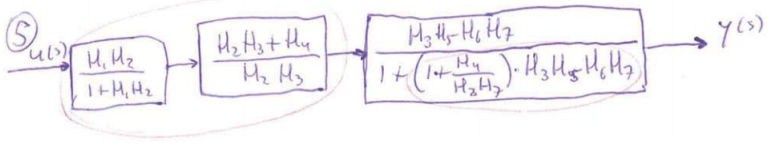


Figure 3.32: Rework V.

Combining the left two blocks leads to

$$\frac{H_1 H_2}{1 + H_1 H_2} \cdot \frac{H_2 H_3 + H_4}{H_2 H_3} = \frac{H_1 H_2 \left(H_2 H_3 + H_4\right)}{\left(1 + H_1 H_2\right) H_2 H_3} = \frac{H_1 \left(H_2 H_3 + H_4\right)}{\left(1 + H_1 H_2\right) H_3}$$

The block on the right can be rewritten to

$$\frac{H_3H_5H_6H_7}{1+H_3H_5H_6H_7+\frac{H_3H_4H_5H_6H_7}{H_3H_7}} = \frac{H_3H_5H_6H_7}{1+H_3H_5H_6H_7+H_4H_5H_6}$$

Combining these leads to

$$\begin{aligned} H_{eq} &= \frac{H_1\left(H_2H_3+H_4\right)\cdot H_3H_5H_6H_7}{\left(1+H_1H_2\right)H_3\cdot\left(1+H_3H_5H_6H_7+H_4H_5H_6\right)} \\ &= \frac{H_1H_5H_6H_7\left(H_4+H_2H_3\right)}{\left(H_1H_2+1\right)\left(H_4H_5H_6+H_3H_5H_6H_7+1\right)} \end{aligned}$$

as shown in figure 3.33.

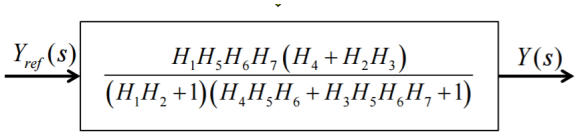


Figure 3.33: Final version.

Again, this is way beyond what they'll ask from you on the exam. Furthermore, just keep in mind that a combination of + and + is simply blocks in parallel, if it's + and - then it's a feedback loop sorta thing.

4 State-space model

State-space models are very popular, and are an alternative way of representing a function.

4.1 Introduction to state-space models

Chances are you have no clue what state-space models are. First of all, state-space models don't use a fancy transform to go to a different domain: you stay in the time-domain. Furthermore, state-space models have the advantage that they are much better able to represent **MIMO systems**: Multiple input, Multiple output systems. Now, what exactly is a state-space model?

It is rather logical to say that we can put all the information needed to determine the future system behaviour without reference to the derivaties of input and output variables. For example, for an aircraft, we could have a **system state** given by the vector $\mathbf{x}(t)$:

$$\mathbf{x} = \begin{bmatrix} \alpha(t) \\ M(t) \\ h(t) \\ \vdots \end{bmatrix}$$

where α is the angle of attack, M the Mach number and h the altitude. Of course, there are many more variables that would describe the state of the aircraft, but note that we're not including any of the derivatives of those variables in this vector.

Now, some terminology, looking at figure 4.1:

ELEMENTS OF A STATE-SPACE MODEL

The elements of a **state-space** model are

- **State variables**: the set of all variables which combine all necessary knowledge of the system at $t = t_0$ such that behaviour of the system can be determined for $t \geq t_0$.
- **State vector x**: an n-dimensional vector containing all state variables.
- **State space**: n-dimensional space whose axes are the state variables.
- **State equations**: a set of first order differential equations in terms of the state variables:

STATE EQUATION

The state equation describes the dynamics of $\mathbf{x}(t)$ and is given by

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{4.1}$$

where

- $\mathbf{x}(t)$ is the **state vector**;
- $\mathbf{u}(t)$ is the **input vector**;
- A is the **state matrix**;
- B is the **input matrix**

Note that the new state can be calculated (using computers) by integrating the state equation:

$$\mathbf{x}(\Delta t) = \mathbf{x}(0) + \int_0^{\Delta t} \dot{\mathbf{x}}(t)\,dt = \mathbf{x}(0) + \int_0^{\Delta t} (A\mathbf{x}(t) + B\mathbf{u}(t))\,dt$$

Note that we could also write this as a block diagram, as shown in figure 4.2.

Let's do an example to see what we have to do:

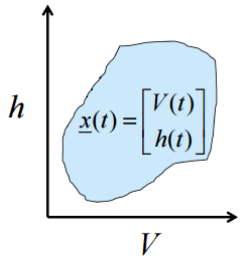


Figure 4.1: Elements of state-space model.

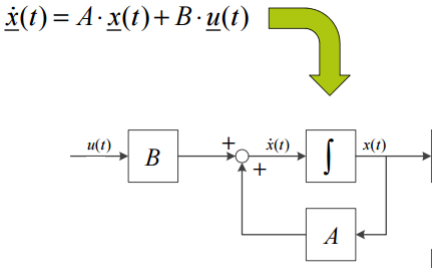


Figure 4.2: Block diagram.

Example 1

Suppose we have the mass-spring damper system shown in figure 4.3.

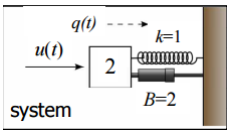


Figure 4.3: Mass-spring damper system.

It should be rather obvious then that the 2nd order ODE would be

$$2\ddot{q}(t) + 2\dot{q}(t) + q(t) = u(t)$$

Writing this as state-space system involves multiple steps. First, we define the state variables, the state vector, and the inputs. As the equation is 2nd order, we need two state variables: these variables will be

$$\mathbf{x}(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

Why? We include the variables of order of up until 2 (not including 2). Furthermore, the input vector is simply

$$\mathbf{u}(t) = \begin{bmatrix} u(t) \end{bmatrix}$$

The second step involves getting the equations for the matrix equation. The first equation would simply be $\dot{x}_1(t) = x_2(t)$. However, the second equation does not follow so easily: $\dot{x}_2(t) = x_3(t)$ would be the next one but we don't have $x_3(t)$. Instead, the second equation is found by plugging the two state vectors definition into the original ODE (note that $\ddot{q} = \dot{x}_2$):

$$\begin{aligned} 2\ddot{q}(t) + 2\dot{q}(t) + q(t) &= u(t) \\ 2\dot{x}_2(t) + 2x_2(t) + x_1(t) &= u(t) \end{aligned}$$

We then rewrite this to

$$\dot{x}_2(t) = \frac{1}{2}\left(-2x_2(t) - x_1(t) + u(t)\right)$$

which will be the first of two equations. What will be the other equation? It'll simply by $\dot{x}_1(t) = x_2(t)$.

The third step then consists of rewriting it as a matrix equation:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{2} & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{2} \end{bmatrix} \begin{bmatrix} u(t) \end{bmatrix} = A\mathbf{x} + B\mathbf{u}$$

so that state-space matrices are

$$A = \begin{bmatrix} 0 & 1 \\ -0.5 & -1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$$

and the state-space vectors

$$\dot{\mathbf{u}}(t) = \begin{bmatrix} u(t) \end{bmatrix}, \qquad \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix}, \qquad \mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

REWRITING A LINEAR HIGHER ORDER ODE AS SYSTEM OF FIRST ORDER ODES

Starting with an nth order ordinary differential equation of the form

$$a_0 x + a_1 \frac{dx}{dt} + \cdots + a_n \frac{d^n x}{dt^n} = a_n a_0 \frac{d^n x}{dt^n} + a_1 \frac{d^{n-1} x}{dt^{n-1}} + \ldots + a_{n-1} \frac{dx}{dt} = a_n$$

1. Introduce the state vector
$$\mathbf{x} = \begin{bmatrix} x_1 = x \\ x_2 = \frac{dx}{dt} \\ \vdots \\ x_{n-1} = \frac{d^{n-2}x}{dt} \\ x_n = \frac{d^{n-1}x}{dt} \end{bmatrix}$$
2. Introduce the vector
$$\dot{\mathbf{x}} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_{n-1}}{dt} \\ \frac{dx_n}{dt} \end{bmatrix}$$
3. A linear system would be of the form
$$\dot{x} = A\mathbf{x} + B\mathbf{u}$$
For the first $n-1$ rows of the system (i.e. every row but the last), the equation will simply be $\dot{x}_i = x_{i+1}$. For the last row, you need to introduce the state variables into the differential equation:
$$a_0 x + a_1 \frac{dx}{dt} + \cdots + a_{n-1} \frac{d^{n-1}x}{dt} + a_n \frac{d^n x}{dt^n} = u$$
becomes, as $x_1 = x$, $x_2 = dx/dt$, ..., $x_n = d^{n-1}/dt^{n-1}$,
$$\begin{aligned} a_0 x_1 + a_1 x_2 + \cdots + a_{n-1} x_n + a_n \dot{x}_n &= u \\ \dot{x}_n &= -\frac{a_0}{a_n} x_1 - \frac{a_1}{a_n} x_2 - \cdots - \frac{a_{n-1}}{a_n} x_n + \frac{1}{a_n} u \end{aligned}$$
4. Write out your matrices, which will become
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & -\frac{a_3}{a_n} & \cdots & -\frac{a_{n-1}}{a_n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix} u \tag{4.2}$$

Note that it may be the case that you get two ODEs at once, e.g.

$$\begin{aligned} m\ddot{h} + K_h h &= 0 \\ I_\theta \ddot{\theta} + K_\theta \theta &= 0 \end{aligned}$$

In that case, our state vector will simply be

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \begin{bmatrix} h \\ \dot{h} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

What exactly do we then? Well, note that we'd again have $\dot{x}_1(t) = x_2(t)$, and also $\dot{x}_3(t) = x_4(t)$: the equations associated with the not-highest order derivatives are easy. For the highest order derivatives, i.e. the ones associated with $\dot{x}_2(t)$ and $\dot{x}_4(t)$, the equations follow again by plugging stuff into the original ODEs:

$$\begin{aligned} m\ddot{h} + K_h h &= 0 \\ m\dot{x}_2(t) + K_h x_1(t) &= 0 \\ \dot{x}_2(t) &= -\frac{K_h}{m} x_1(t) \end{aligned}$$

and similarly

$$\begin{aligned} I_\theta \ddot{\theta} + K_\theta \theta &= 0 \\ I_\theta \dot{x}_4(t) + K_\theta x_3(t) &= 0 \\ \dot{x}_4 &= -\frac{K_\theta}{I_\theta} x_3(t) \end{aligned}$$

Thus, we have the systems of equations

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{K_h}{m} x_1(t) \\ \dot{x}_3(t) &= x_4(t) \\ \dot{x}_4(t) &= -\frac{K_\theta}{I_\theta} x_3(t) \end{aligned}$$

or, in matrix notation,

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K_h}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{K_\theta}{I_\theta} & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$

4.1.1 *The output equation*

Now, the bad news is that we actually don't know the state of the aircraft directly: we use sensors to measure the state. So, instead, we have the output equation:

OUTPUT EQUATION

The **output equation** is given by

$$y(t) = C\mathbf{x}(t) + D\mathbf{u}(t) \tag{4.3}$$

where

- $\mathbf{x}(t)$ is the state vector;
- $\mathbf{u}(t)$ is the input vector;
- C is the **output matrix**;
- D is the **feedthrough or feedforward matrix**.

In above equation, C depends heavily in reality on which sensors we have available and how they behave; D depends on how the input is directly transmitted to the output. Often, D is equal to 0 (as that would mean that what you measure is a directly affected by an input, which does not seem likely). Note that these matrices are very different from A and B, as those heavily depend on the physical dynamics of the system.

Now, let me be clear: the output and and state equations are totally different equations: the state equation is used to calculate the change in state variables, $\dot{\mathbf{x}}(t)$, whereas the output equation is used to compute the output of the system, as measured by signals. However, *both* equations use exactly the same state variables in their vector $\mathbf{x}(t)$ and the same input vector (so $\mathbf{x}(t)$ and $\mathbf{u}(t)$ re exactly the same equations; you can't be like, nahhh fack it I'll just use a different vector for the output equation).

Furthermore, note that the equation

$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

can be written as the block diagram shown in figure 4.4.

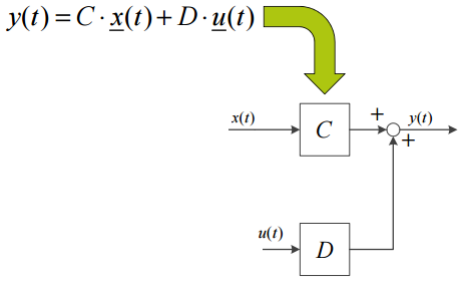


Figure 4.4: Block diagram.

Example 1: continued

Formulate the output equation of the state-space model if we only have a sensor for measuring $\dot{q}$ (a sensor for the velocity of the mass):

$$\mathbf{y}(t) = \begin{bmatrix} \dot{q}(t) \end{bmatrix}$$

Now, note that we can directly relate this to a state, namely $\dot{q}(t) = x_2$. Thus, the output equation is then

$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + 0 \cdot \mathbf{u}(t)$$

Now, suppose we'd have a sensor for measuring both $\dot{q}$ and q (in this order). What would the output equation be then?

Note that we must write

$$\mathbf{y}(t) = \begin{bmatrix} \dot{q}(t) \\ q(t) \end{bmatrix}$$

and not $\mathbf{y}(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}$ as the order is fixed! Now, we have (again) $\dot{q}(t) = x_2$ and $q(t) = x_1$, thus we have

$$\begin{aligned} \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) \\ \begin{bmatrix} \dot{q}(t) \\ q(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} \end{aligned}$$

Suppose we have a sensor that measures a linear combination of $\dot{y}$ and y (e.g. some kind of energy sensor):

$$\mathbf{y}(t) = \left[m\dot{q}(t) + kq(t)\right]$$

What would the output equation be then?

The output equation is then simply (since $\dot{q}(t) = x_2(t)$ and $q(t) = x_1(t)$

$$\mathbf{y}(t) = \begin{bmatrix} k & m \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + 0 \cdot \mathbf{u}(t)$$

Now, a nice definition:

THE COMPLETE STATE-SPACE MODEL

The **complete state-space model** consists of the state equation describing the system dynamics, and the output equation describing which states are measured:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{4.4}$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) \tag{4.5}$$

which can be written as the block diagram of figure 4.5.

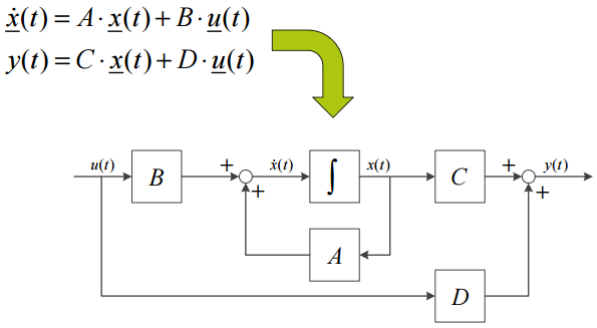
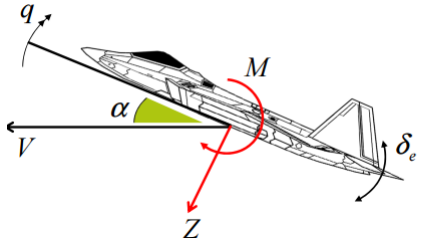


Figure 4.5: Block diagram.

Let's now do a slightly more complicated example, an actual aerospace example:

Example 2

Suppose we investigate the pitch dynamics of an aircraft in state-space form, and we have sensors for angle of attack α and pitch rate q (rate of change of pitch attitude, not the pitch angle itself). Look at figure 4.6 for reference.



Symbol	Description
q	pitch rate
α	angle of attack
V	velocity = **constant!**
δ_e	elevator deflection
Z	Aerodynamic force
M	Aerodynamic moment

Figure 4.6: Aircraft characteristics.

First of all, trust me that we have the following equation of motion (where α is the angle of attack, q the

pitch rate (so the change in pitch angle) and δ_e the deflection angle of the elevators).

$$\begin{aligned} z_1\frac{1}{V}\dot{\alpha} + z_2\alpha + z_3\frac{1}{V}q + z_4\delta_e &= 0 \\ m_2\alpha - m_3\frac{1}{V}\dot{q} + m_4\frac{1}{V}q + m_5\delta_e &= 0 \end{aligned}$$

Again, we first define the state vector, input vector and output vector:

$$\mathbf{x} = \begin{bmatrix}\alpha \\ q\end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix}\delta_e\end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix}\alpha \\ q\end{bmatrix}$$

Where does the state vector come from? The highest order derivative of α is first order, so we only include α. The highest order derivative of q is also only first order, so we only include q (if we'd had $\ddot{q}$ appear in one of the equations, we'd have included $\dot{q}$ as well, but not $\dot{\alpha}$ if there had been no $\ddot{\alpha}$).

Then, we rewrite the two equations of motion: the first one can be rewritten to

$$\begin{aligned} z_1\frac{1}{V}\dot{\alpha} + z_2\alpha + z_3\frac{1}{V}q + z_4\delta_e &= 0 \\ \dot{\alpha} &= -\frac{z_2}{z_1}V\alpha - \frac{z_3}{z_1}q - \frac{z_4}{z_1}V\delta_e \end{aligned}$$

and the second one can be rewritten to

$$\begin{aligned} m_2\alpha - m_3\frac{1}{V}\dot{q} + m_4\frac{1}{V}q + m_5\delta_e &= 0 \\ \dot{q} &= \frac{m_2}{m_3}\alpha + \frac{m_4}{m_3}q + \frac{m_5}{m_3}V\delta_e \end{aligned}$$

Thus, we have the system

$$\begin{aligned} \dot{\alpha} &= -\frac{z_2}{z_1}V\alpha - \frac{z_3}{z_1}q - \frac{z_4}{z_1}V\delta_e \\ \dot{q} &= \frac{m_2}{m_3}\alpha + \frac{m_4}{m_3}q + \frac{m_5}{m_3}V\delta_e \\ \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \begin{bmatrix}\dot{\alpha} \\ \dot{q}\end{bmatrix} &= \begin{bmatrix}-\frac{z_2}{z_1}V & -\frac{z_3}{z_1} \\ \frac{m_2}{m_3}V & \frac{m_4}{m_3}\end{bmatrix}\begin{bmatrix}\alpha \\ q\end{bmatrix} + \begin{bmatrix}-\frac{z_4}{z_1}V \\ \frac{m_5}{m_3}V\end{bmatrix}\delta_e \end{aligned}$$

Suppose we'd have $z_1 = -1.5$, $z_2 = -1.25$, $z_3 = -4$, $z_4 = -0.15$, $m_2 = 2$, $m_3 = -15$, $m_4 = -0.1$ and $m_5 = -0.025$, then this equation becomes

$$\dot{\mathbf{x}} = \begin{bmatrix}\dot{\alpha} \\ \dot{q}\end{bmatrix} = \begin{bmatrix}-83.33 & -2.67 \\ -5 & -7.5\end{bmatrix}\begin{bmatrix}\alpha \\ q\end{bmatrix} + \begin{bmatrix}-10 \\ -1.25\end{bmatrix}\delta_e$$

For the output equation, we have $\mathbf{y} = \begin{bmatrix}\alpha \\ q\end{bmatrix} = \begin{bmatrix}\alpha \\ q\end{bmatrix}$, thus we simply have

$$\begin{aligned} \mathbf{y} &= C\mathbf{x}(t) + D\mathbf{u}(t) \\ \begin{bmatrix}\alpha \\ q\end{bmatrix} &= \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha \\ q\end{bmatrix} + \begin{bmatrix}0 \\ 0\end{bmatrix}\delta_e \end{aligned}$$

The full state-space model is thus given by:

$$\begin{aligned} \begin{bmatrix}\dot{\alpha} \\ \dot{q}\end{bmatrix} &= \begin{bmatrix}-83.33 & -2.67 \\ -5 & -7.5\end{bmatrix}\begin{bmatrix}\alpha \\ q\end{bmatrix} + \begin{bmatrix}-10 \\ -1.25\end{bmatrix}\delta_e \\ \begin{bmatrix}\alpha \\ q\end{bmatrix} &= \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha \\ q\end{bmatrix} + \begin{bmatrix}0 \\ 0\end{bmatrix}\delta_e \end{aligned}$$

4.2 Extending state-space systems

4.2.1 Adding states: effect on state equation

Now, there's an important deficiency in above results: we could make α very large, but we'd keep flying at the same altitude as we never do anything with the altitude. However, we can rather easily include the altitude (and while we're at it, let's also include the velocity). This is very easily done, actually (this is one of the main advantages of state-space systems: they can easily be extended to include more variables). We must find two extra equations; these will be $\dot{\theta} = q$ (as the change in attitude (pitch angle) is equal to the pitch rate) and $\dot{h} = V \cdot \gamma = V \cdot (\theta - \alpha)$ (see also figure 4.7)[1]. Thus, the state equation becomes

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -83.33 & -2.67 & 0 & 0 \\ -5 & -7.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -V & 0 & V & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \\ h \end{bmatrix} + \begin{bmatrix} -10.00 \\ -1.25 \\ 0 \\ 0 \end{bmatrix} \delta_e$$

Simple as that. Note that there the extra columns in the first and second row of the matrix are simply zero as θ and h do not appear in the equation for $\dot{\alpha}$.

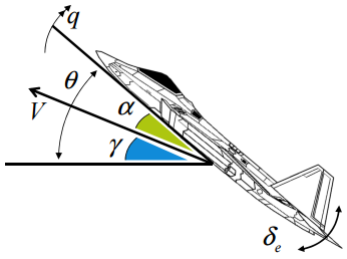


Figure 4.7: Aircraft attitude and pitch.

4.2.2 Adding outputs: effect on output equation

Now, you may ever get into a situation where your states are fine (there's no need to add extra states I mean), but you for example add an extra sensor to your system, which will produce a new output. How can we incorporate this extra sensor in our output equation? Let's look at an example.

Example 3

Suppose we have the new state-space system given as (note that this is basically the equation I gave just before, but without the equation for V)

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -83.33 & -2.67 & 0 \\ -5 & -7.5 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} -10.00 \\ -1.25 \\ 0 \end{bmatrix} \delta_e$$

and with the original output equation

$$\begin{bmatrix} \alpha \\ q \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Suppose we want to know the climb rate as well. Add the climb rate $\dot{h}$ to the output equation of the given state-space system.

In other words, we must leave the state equation unchanged, but we must modify the output equation so that there is another row describing $\dot{h}$. However, it is absolutely required that you keep the following in

[1] To be clear, on the exam it'll be very clear which equations you have to add; the main point is here how to *add* these equations to the system, not how to derive these extra equations yourself.

mind: for $\mathbf{y}$, you need to use the same $\mathbf{x}$ as given in the state-equation, i.e.

$$\mathbf{x} = \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

and thus our equation of $\mathbf{y}$ must be of the form

$$\mathbf{y} = C\mathbf{x} = \begin{bmatrix} \alpha \\ q \\ h \end{bmatrix} = C \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix}$$

Now, the first two equations of this system will remain the exact same: our first two equations will stay

$$\begin{aligned} \alpha &= \alpha \\ q &= \theta \end{aligned}$$

Now, how can we write $\dot{h}$ as a function α, q and θ? Well, basically, we simply get $\dot{h} = V(\theta - \alpha)$ as before (look at figure 4.7). Thus, we now get the system of equations

$$\begin{aligned} \alpha &= \alpha \\ q &= q \\ \dot{h} &= -V\alpha + V\theta \end{aligned}$$

and thus we have our output equation

$$\begin{aligned} \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \\ \begin{bmatrix} \alpha \\ q \\ \dot{h} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -V & 0 & \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} \end{aligned}$$

Example 4

Suppose we have the state equation

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -83.33 & -2.67 & 0 & 0 \\ -5 & -7.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -V & 0 & V & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \\ h \end{bmatrix} + \begin{bmatrix} -10.00 \\ -1.25 \\ 0 \\ 0 \end{bmatrix} \delta_e$$

and you want to have an output equation that outputs the vector

$$\mathbf{y}(t) = \begin{bmatrix} \alpha \\ q \\ \dot{h} \\ h \end{bmatrix}$$

What would be the output equation?

Note that we'd still have $\alpha = \alpha$, $q = q$, $\dot{h} = -V\alpha + V\alpha$, but we'd need to come up with an equation for h. However, note that it is already included in $\mathbf{x}$, thus, we simply have $h = h$ and the equation becomes

$$\mathbf{y} = \begin{bmatrix} \alpha \\ q \\ \dot{h} \\ h \end{bmatrix} = C\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -V & 0 & V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \\ h \end{bmatrix}$$

4.2.3 Adding inputs

Now, can we also extend our state-space system to include an extra input, for example to include the thrust vectoring (i.e. we can control the direction of the thrust)? Well, if we include the thrust vector in the equations of motion:

$$\begin{aligned} z_1\frac{1}{V}\dot{\alpha} + z_2\alpha\alpha + z_3\frac{1}{V}q + z_4\delta_e + z_{tv}\frac{1}{V}\delta_{tv} &= 0 \\ m_2\alpha - m_3\frac{1}{V}\dot{q} + m_4\frac{1}{V}q + m_5\delta_e + m_{tv}\frac{1}{V}\delta_{tv} &= 0 \end{aligned}$$

leading to the matrix equations

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} -\frac{z_2V}{z_1} & -\frac{z_3}{z_1} & 0 & 0 \\ \frac{m_2V}{m_3} & \frac{m_4}{m_3} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -V & 0 & V & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ q \\ \theta \\ h \end{bmatrix} + \begin{bmatrix} -\frac{z_4V}{z_1} & \frac{z_{tv}}{z_1} \\ \frac{m_5V}{m_3} & \frac{m_{tv}}{m_3} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \delta_e \\ \delta_{tv} \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -V & 0 & V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \alpha \\ q \\ \theta \\ h \end{bmatrix} \end{aligned}$$

Note that the thrust vector angle δ_{tv} is not a state but an input.

4.3 Controllers for state-space systems

For state-space systems, the feedback gain is actually a matrix, called the **gain matrix**. Why? Take, for example, a 3 state system with 2 inputs and 1 output, as shown in figure 4.8. To make the output $\mathbf{y}$ 'compatible' with $\mathbf{r}$, we must have that K is a 2×1 matrix. To be precise:

SIZE OF FEEDBACK GAIN MATRIX

> The feedback gain matrix needs to be of the number of rows equal to the number of inputs, and the number of columns equal to the number of outputs. *The number of states does not matter.*

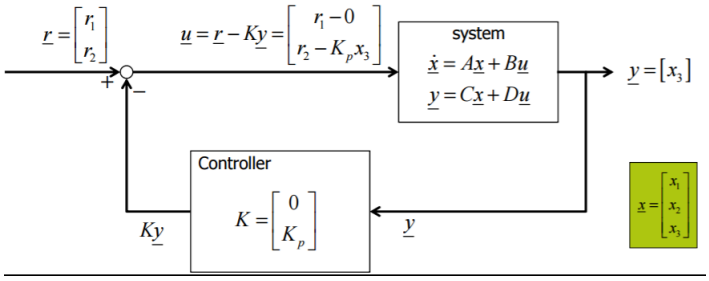


Figure 4.8: State-space system block diagram.

Suppose we have a 3 state system, with 2 inputs and 3 outputs. K then needs to be a 2×3 matrix, as shown in figure 4.9.

Now, what are the entries of K? Well, you can set it up manually: in figure 4.9, we apparently wanted to compare the output x_3 with reference r_2; all other entries are zero because we simply don't compare x_2 with r_1 for example.

Consider the pitch rate control for the unstable aircraft of before, if we include thrust vectoring, as depicted in figure 4.10. In that case, we will have two reference values: we will compare the output with some reference pitch rate for the elevator, and a reference pitch rate for the thrust vectoring, i.e. $\mathbf{q}_{ref} = \begin{bmatrix} q_{ref,ele} \\ q_{ref,tv} \end{bmatrix}$. To be precise,

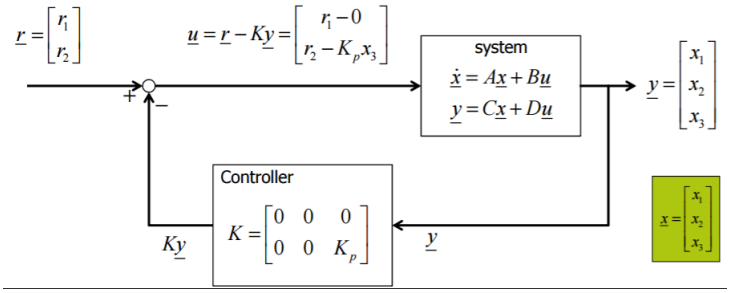


Figure 4.9: State-space system block diagram.

this vector should be compared with the vector $\begin{bmatrix} K_{de}q(t) \\ K_{tv}q(t) \end{bmatrix}$ where K_{de} is the gain corresponding to the elevator angle and K_{tv} to the angle of the thrust vectoring. To get this vector from **y** (look carefully at the output **y** in figure 4.10, you can check yourself that you need the matrix

$$K = \begin{bmatrix} 0 & K_{de} & 0 & 0 \\ 0 & K_{tv} & 0 & 0 \end{bmatrix}$$

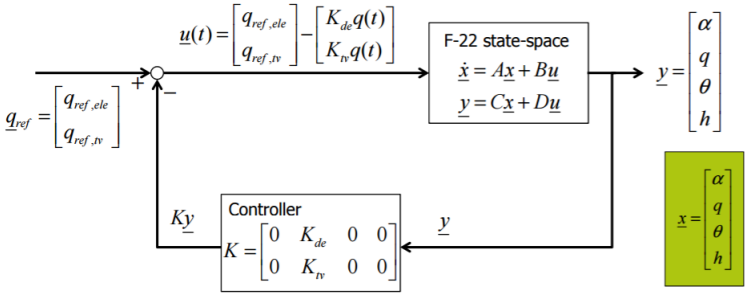


Figure 4.10: State-space system block diagram.

4.3.1 *Disturbances*

Now, consider again the pitch rate controller. Can we have disturbances in there as well? Yes, we can, as shown in figure 4.11. The disturbance shown is turbulence, which affects the angle of attack directly, but not the pitch rate, attitude of altitude (these are affected by turbulence eventually, but no directly when there is a gust of wind (whereas the angle of attack is directly affected, as the direction of the local wind changes)).

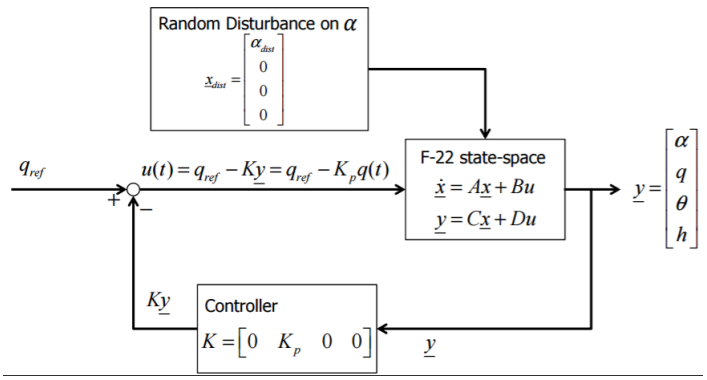


Figure 4.11: State-space system block diagram, including disturbance.

4.4 Converting transfer functions to state-space models

Suppose you were stupid and came up with the transfer functions in the Laplace domain but it turned out that you're stupid and couldn't solve it: can we then still save you and convert it to a state-space model?

Yes, you can, fortunately (even if you're stupid):

CONVERTING TRANSFER FUNCTION TO STATE-SPACE MODELS

Suppose we have the **controller canonical form**

$$H(s) = \frac{b_0 + b_1 s + \cdots + b_m s^m}{a_0 + a_1 s + \cdots + a_n s^n}$$

with $m < n$ (if $m \geq n$, please see example 6):

1. Split the transfer function into two parts, as shown in figure 4.12.

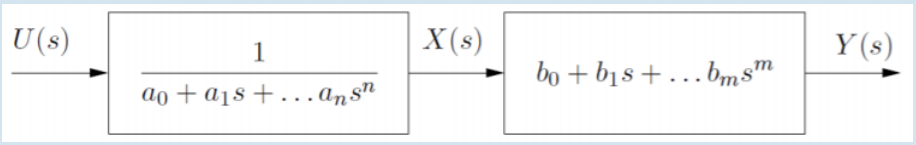


Figure 4.12: Splitting the transfer function.

2. To obtain the state equation:
   (a) $X(s)/U(s)$ then leads to the differential equation for the input:

   $$U(s) = X(s)\left(a_0 + a_1 s + \cdots + a_n s^n\right)$$

   which can straightforwardly be inverse transformed to

   $$u = a_0 x + a_1 \frac{dx}{dt} + \cdots + a_{n-1}\frac{d^{n-1}x}{dt^{n-1}} + a_n \frac{d^n x}{dt^n}$$

   (b) Apply the problem solving guide "Rewriting a linear higher order ODE as system of first order ODEs" to determine the state equation; in other words, your result will be

   $$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & -\frac{a_3}{a_n} & \cdots & -\frac{a_{n-1}}{a_n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix} u \tag{4.6}$$

3. To obtain the output equation:
   (a) $Y(s)/X(s)$ leads to

   $$Y(s) = X(s)\left(b_0 + b_1 s + \cdots + b_m s^m\right)$$

   which can straightforwardly be transformed to

   $$y = b_0 x + b_1 \frac{dx}{dt} + \cdots + b_{m-1}\frac{d^{m-1}x}{dt^{m-1}} + b_m \frac{d^m x}{dt^m}$$

   (b) Introduce $x_1 = x$, $x_2 = \frac{dx}{dt}$, ..., $x_m = \frac{d^{m-1}x}{dt^{m-1}}$ in this equation to get the output equation

   $$y = \begin{bmatrix} b_0 & b_1 & \cdots & b_m & 0_{1\times(n-m-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u$$

   where $0_{1\times(n-m-1)}$ means that there are $n - m - 1$ zeros there.

Let's see this in practice.

Example 5

Convert the equivalent transfer function (TF) of our controlled taxiing aircraft into state space form using the 4 step procedure. Use the input of the TF as input to the state-space system, and the output of the TF as the output of the state-space model. The equivalent transfer function of the controlled taxiing aircraft is:

$$H(s) = \frac{K_y K_\psi V^2 l}{s^2 + \left(K_\psi V/l\right) s + K_y K_\psi V^2/l}$$

See also figure 4.13.

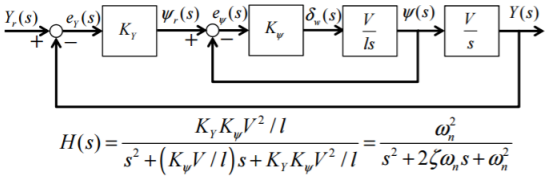


Figure 4.13: Block diagram.

The first step is to split the transfer function in two parts: we can write it as shown in figure 4.14.

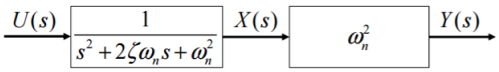


Figure 4.14: Splitting the transfer function.

Thus, we have for step 2a that

$$U(s) = X(s)\left(\omega_n^2 + 2\zeta\omega_n s + s^2\right)$$

which simply becomes

$$u = \omega_n^2 x + 2\zeta\omega_n \frac{dx}{dt} + \frac{d^2x}{dt^2}$$

Thus, for step 2b, the state equation becomes (since $a_n = a_2 = 1$, $a_0 = \omega_n^2$ and $a_1 = 2\zeta\omega_n$)

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \end{aligned}$$

Then, for step 3a, we get

$$Y(s) = X(s)\left(\omega_n^2\right)$$

and thus

$$y = \omega_n^2 x$$

For step 3b, this then becomes

$$y = \omega_n^2 x_1$$

so that

$$\begin{aligned} \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \\ y &= \begin{bmatrix} \omega_n^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\, u \end{aligned}$$

Now, I indicated before that if you have

$$H(s) = \frac{b_0 + b_1 s + \cdots + b_m s^m}{a_0 + a_1 s + \cdots + a_n s^n}$$

with $m \geq n$, then there's something different you have to do. The reason for this the output equation for y: we'd the zeroes at the end of it wouldn't make sense any more (suppose $n = m$, then we'd need -1 zeroes at the end of the matrix, how would that work? Not). So, let me amend a bit what you then have to do: :

Example 6

Suppose we have the transfer function

$$H(s) = \frac{2s^2}{s^2 + 2s + 1}$$

which clearly has $m = n = 2$. We can, however, rewrite it as follows:

$$\begin{aligned} H(s) &= \frac{2s^2}{s^2+2s+1} = \frac{2s^2}{s^2+2s+1} - 2 + 2 = \frac{2s^2}{s^2+2s+1} - 2\frac{s^2+2s+1}{s^2+2s+1} + 2 \\ &= \frac{2s^2 - 2s^2 - 4s - 2}{s^2 + 2s_1 + 1} + 2 = \frac{-4s-2}{s^2+2s+1} + 2 \end{aligned}$$

where we do have $m < n$, and thus we can use the procedure outlined before. Note that we now have the system shown in figure 4.15.

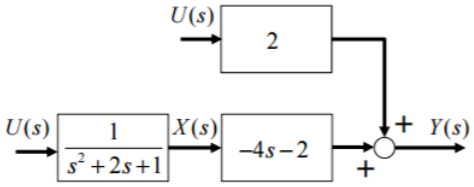


Figure 4.15: Splitting the transfer function.

We can then continue with the procedure: we get for step 2a that

$$U(s) = X(s)\left(1 + 2s + s^2\right)$$

and thus

$$u = x + 2\frac{dx}{dt} + \frac{d^2x}{dt}$$

For step 2b, this leads to (since $a_n = a_2 = 1$, $a_0 = 1$ and $a_1 = 2$):

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \end{aligned}$$

For step 3a, we get

$$Y(s) = X(s)(-2 - 4s) + 2U(s)$$

so that

$$y = -2x - 4\frac{dx}{dt} + 2u$$

For step 3b, the equation becomes slightly different: we don't have the zeroes in the left matrix any more, and in front of u, it now simply becomes 2 (should make a lot of sense, honestly):

$$\begin{aligned} \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \\ y &= \begin{bmatrix} -2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 \end{bmatrix} u \end{aligned}$$

Note that when $m \geq n$, the feed forward matrix D is not equal to zero!

From these two examples and the problem solving guide, I hope you realize that it's actually really easy to set up a state-space system if the differential equation is known: it's just a matter of remembering what should go into what matrix, basically.

4.5 Cramer's rule

Now, can we also go the opposite way? If we already have our state-space model, can we deduce our transfer functions? Yes, we can, using some linear algebra (the derivation is unimportant but I'll show it anyway): we start with

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

which becomes in the Laplace domain

$$\begin{aligned} sX(s) &= AX(s) + BU(s) \\ sX(s) - AX(s) &= BU(s) \\ (sI - A)X(s) &= BU(s) \end{aligned}$$

so that

TRANSFER FUNCTIONS FROM STATE EQUATION

The system of transfer functions from a state-space system is given by

$$\frac{X(s)}{U(s)} = (sI - A)^{-1} B \tag{4.7}$$

which will look like

$$\frac{X(s)}{U(s)} = \begin{bmatrix} \frac{x_1(s)}{u_1(s)} & \frac{x_1(s)}{u_2(s)} & \dots & \frac{x_1(s)}{u_n(s)} \\ \frac{x_2(s)}{u_1(s)} & \frac{x_2(s)}{u_2(s)} & \dots & \frac{x_2(s)}{u_n(s)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_m(s)}{u_1(s)} & \frac{x_m(s)}{u_2(s)} & \dots & \frac{x_m(s)}{u_n(s)} \end{bmatrix} \tag{4.8}$$

which has as many rows as there are states, and as many columns as there are inputs.

The output equation can similarly be transformed:

$$\begin{aligned} y(t) &= Cx(t) + Du(t) \\ Y(s) &= CX(s) + DU(s) = C(sI - A)^{-1} BU(s) + DU(s) \end{aligned}$$

where we simply substitute the expression for $X(s)$. We then divide by $U(s)$ to arrive at

TRANSFER FUNCTIONS FROM STATE EQUATION

The system of transfer functions from a state-space system is given by

$$\frac{Y(s)}{U(s)} = C(sI - A)^{-1} B + D \tag{4.9}$$

which will look like

$$\frac{Y(s)}{U(s)} = \begin{bmatrix} \frac{y_1(s)}{u_1(s)} & \frac{y_1(s)}{u_2(s)} & \dots & \frac{y_1(s)}{u_n(s)} \\ \frac{y_2(s)}{u_1(s)} & \frac{y_2(s)}{u_2(s)} & \dots & \frac{y_2(s)}{u_n(s)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{y_m(s)}{u_1(s)} & \frac{y_m(s)}{u_2(s)} & \dots & \frac{y_m(s)}{u_n(s)} \end{bmatrix} \tag{4.10}$$

which has as many rows as there are outputs, and as many columns as there are inputs.

Now, the thing is, it can be rather laborious to calculate the entire matrix $X(s)/U(s)$ if we are only interested in certain elements of it (e.g. looking at equation (4.8), we may only be interested in $x_2(s)/u_2(s)$ and not all the other elements). How can we do that quickly? We have Cramer's rule for that (which you may vaguely recall from last), which went something like this: suppose we have

$$\frac{X(s)}{U(s)} = (sI - A)B = \begin{bmatrix} s-a_{11} & -a_{12} & -a_{13} \\ -a_{21} & s-a_{22} & -a_{23} \\ -a_{31} & -a_{32} & s-a_{33} \end{bmatrix}^{-1} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} \frac{x_1(s)}{u_1(s)} & \frac{x_1(s)}{u_2(s)} \\ \frac{x_2(s)}{u_1(s)} & \frac{x_2(s)}{u_2(s)} \\ \frac{x_3(s)}{u_1(s)} & \frac{x_3(s)}{u_2(s)} \end{bmatrix}$$

then we could find $x_3(s)/u_2(s)$ as follows:

1. Replace the third column (due to x_3) of $sI - A$ by the second column (due to u_2) of B;
2. Divide the determinant of this matrix by the determinant of the original matrix $sI - A$.

This means that we'd get

$$\frac{x_3(s)}{u_2(s)} = \frac{\begin{vmatrix} s-a_{11} & -a_{12} & b_{12} \\ -a_{21} & s-a_{22} & b_{22} \\ -a_{31} & -a_{32} & b_{32} \end{vmatrix}}{\begin{vmatrix} s-a_{11} & -a_{12} & -a_{13} \\ -a_{21} & s-a_{22} & -a_{23} \\ -a_{31} & -a_{32} & s-a_{33} \end{vmatrix}}$$

This allows you to straightforwardly calculate the desired transfer function (this would be the transfer function to go from input 2 to state 3, btw) without calculating the inverse of the entire matrix.

Don't worry too much about doing it by hand, we'll see in the E-lectures how to do it with Matlab.

Finally, the teacher said literally during the lecture that this was 100% guaranteed going to be in the exam, that you have to be able to select a transfer function starting from a state-space system.

4.6 *Converting block diagrams to state-space models*

To close this chapter off, we can also convert block diagrams to state-space models. Consider the block diagram of figure 4.16 (I couldn't get rid of the red symbols unfortunately). It describe the roll-attitude hold autopilot mode. To convert this to a state-space model, basically the only thing you have to remember is that the result of every integration in the block diagram should be considered a state for your state-space system. That means that in figure 4.16, your states will be $x_3 = \delta_a$ (as this is the result of an integration, see the $1/s$ in front of it); $x_2 = \dot{\phi}$ (as again there is an integration in front of it, the $K_a/(\tau_a s + 1)$); and finally $x_1 = \phi$.

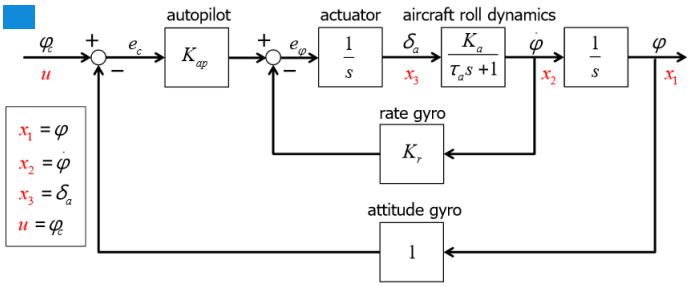


Figure 4.16: Block diagram.

Furthermore, the input is $u = \phi_c$. Now, since we have three states, we have to come up with three equations, each of the form $\dot{x}_1 = ...$, $\dot{x}_2 = ...$ and $\dot{x}_3 = ...$ The first two are relatively easy: for the first one, we see the relation

$$x_1 = \frac{x_2}{s} \qquad \rightarrow \qquad x_1 s = x_2$$

which, upon transforming to the time domain becomes

$$\dot{x}_1 = x_2$$

Similarly, we see that for the roll dynamics, we have figure 4.17, so that we have

$$\begin{aligned} x_3 \frac{K_a}{\tau_a s + 1} &= x_2 \\ K_a x_3 &= x_2 \tau_a s + x_2 \\ x_2 \tau_a s &= K_a x_3 - x_2 \\ \dot{x}_2 \tau_a &= K_a x_3 - x_2 \\ \dot{x}_2 &= -\frac{1}{\tau_a} x_2 + \frac{K_a}{\tau_a} x_3 \end{aligned}$$

and that'll be our second equation.

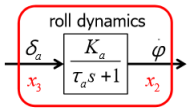


Figure 4.17: Roll dynamics transformation.

Now, the third one will be relatively harder, but it's just a matter of working stuff out. First, we see that

$$x_3 = \frac{e_\phi}{s}$$

or $x_3 s = e_\phi$, where

$$e_\phi = K_{ap} e_c - K_r \dot{\phi} = K_{ap} e_c - K_r x_2$$

since $x_2 = \dot{\phi}$. K_{ap} and K_r are simply constants, but we need to get rid of e_c, but for that one, the following relation holds:

$$e_c = \phi_c - \phi = u - x_1$$

since $u = \phi_c$ and $\phi = x_1$. Thus, we get

$$x_3 s = e_\phi = K_{ap} e_c - K_r x_2 = K_{ap} \left(u - x_1 \right) - K_r x_2 = -K_{ap} x_1 - K_r x_2 + K_{ap} u$$

which becomes

$$\dot{x}_3 = -K_{ap} x_1 - K_r x_2 + K_{ap} u$$

Thus, we have the following system of equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{1}{\tau_a} x_2 + \frac{K_a}{\tau_a} x_3 \\ \dot{x}_3 &= -K_{ap} x_1 - K_r x_2 + K_{ap} u \end{aligned}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{\tau_a} & \frac{K_a}{\tau_a} \\ -K_{ap} & -K_r & 0 \end{bmatrix}$$

5 Dynamic analysis

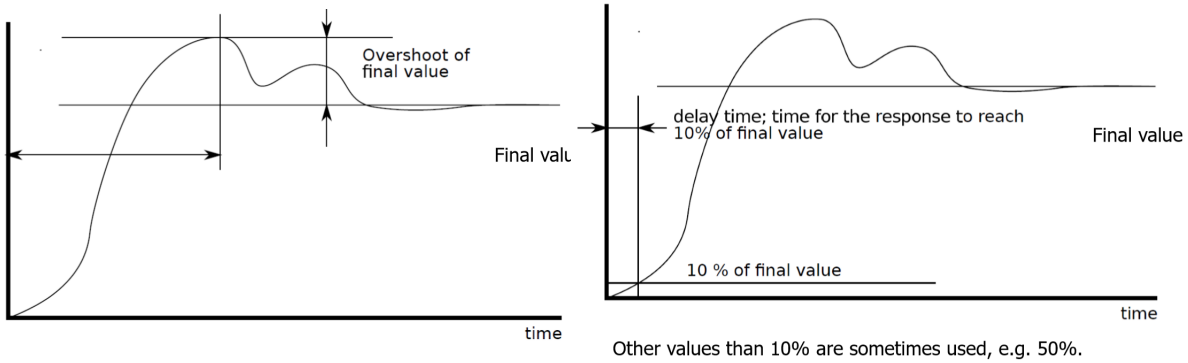
5.1 Dynamic properties

A number of response criteria are normally applied for dynamic systems:

- Maximum overshoot
- Delay time
- Rise time
- Peak time
- Settling time

These are shown in figures 5.1a to 5.2b:

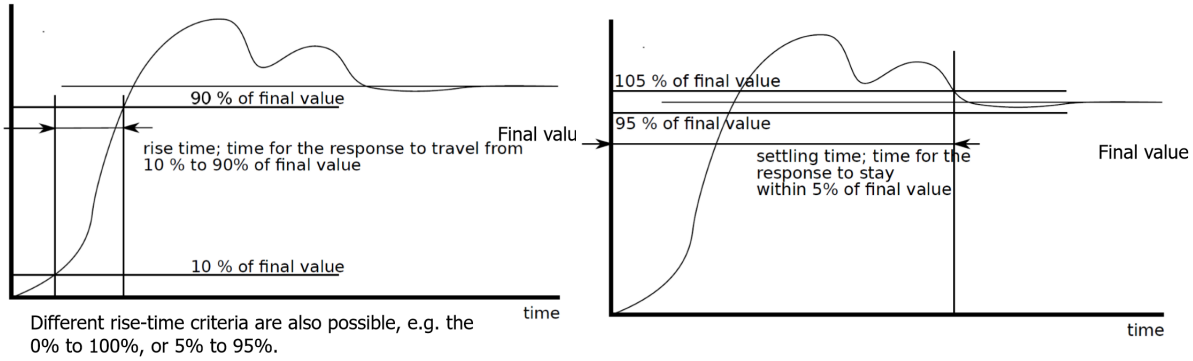
- Overshoot is how much the the output overshoots the final value, as shown in figure 5.1a.
- Delay time is the time for the response to reach 10% of the final value, as shown in figure 5.1b.
- Rise time is the time for the response to travel from 10% to 90% of the final value,a shown in figure 5.2a.
- Peak time is the time it takes for the response to reach its peak.
- Settling time is the time for the response to stay within 5% of the final value.



(a) Overshoot.

(b) Delay time.

Figure 5.1: Overshoot and delay time.



(a) Rise time.

(b) Settling time.

Figure 5.2: Rise and settling time.

This can be summarized as shown in figure 5.3.

Fortunately, there are special Matlab commands which can output these quantities easily.

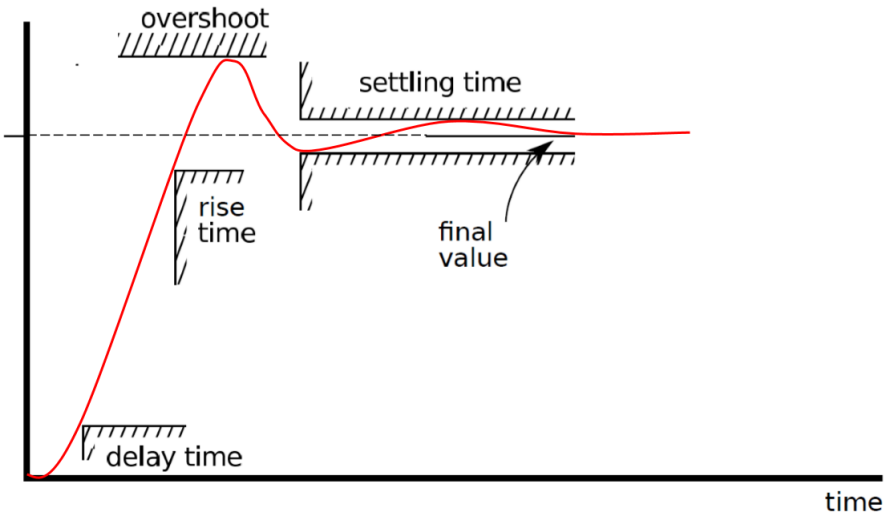


Figure 5.3: Criteria for assessing respons to step input.

5.2 *Initial and Final value theorems*

Now, so far, one way to determine the final value of an output would be to simply plot the output, and look what the value is at a very large value of t. However, this is not always precise enough (especially if it takes a long time to reach a constant value), and instead, we can use the following theorem to calculate it exactly:

FINAL VALUE THEOREM

The **Laplace final value theorem** states that

$$\lim_{T\to\infty} f(t) = \lim_{s\to 0} \{sF(s)\} \tag{5.1}$$

On the other hand, the initial value can be calculated using the initial value theorem:

INITIAL VALUE THEOREM

The **Laplace initial value theorem** states that

$$\lim_{T\to 0} f(t) = \lim_{s\to\infty} \{sF(s)\} \tag{5.2}$$

If you have no idea what this means: suppose we $Y(s) = \frac{1}{s}H(s)$, with transfer function

$$H(s) = \frac{625}{25s^3 + 1.25s^2 + 625s + 25}$$

The final value theorem then tells us that we can predict the value of $f(t)$ as t goes towards infinity by simply plugging in $s = 0$ in the transfer function:

$$\lim_{T\to\infty} f(t) = \lim_{s\to 0} \{sF(s)\} = \lim_{s\to 0} \left\{ s\frac{1}{s}H(s) \right\} = \lim_{s\to 0} \left\{ \frac{625}{25s^3 + 1.25s^2 + 625s + 25} \right\} = \frac{625}{25} = 25$$

In other words, $f(t)$ will go towards 25 for $t \to \infty$. Pretty easy right?

The values of the first, second, third etc. derivative can also be easily calculated by computing the limits of $s^2F(s)$, $s^3F(s)$, etc. Shouldn't be too hard.

Note that you have to learn these formulas by heart! They won't be given to you on the exam, unfortunately.

5.3 Poles and zeroes

Consider a transfer function of the form:

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_0 + b_1 s + b_2 s^2 + \cdots + b_m s^m}{a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n}$$

We can rewrite this transfer function as follows:

$$H(s) = K \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)}$$

Then:

ZEROES AND POLES

The z_i are the **zeroes** of the transfer function; if $s = z_i$, then $N(s) = 0$ and $H(s) = 0$.

The p_j are the **poles** of the the transfer function; if $s = p_j$, then $D(s) = 0$ and $H(s) \to \infty$.

Example 1

Suppose we have the following transfer function:

$$H(s) = \frac{(s+1)(s-2)}{(s+2)(s-i)(s+i)(s+5+2i)(s+5-2i)}$$

What are the poles and zeroes of this transfer function?

The poles are $s = -2$, $s = i$, $s = -i$, $s = -5-2i$, $s = -5+2i$. The zeroes are $s = -1$ and $s = 2$. These can be plotted in figure 5.4; note that we uses crosses for poles and circles for zeroes; this is the notation we use. Furthermore, note that complex poles will always appear in complex conjugates.

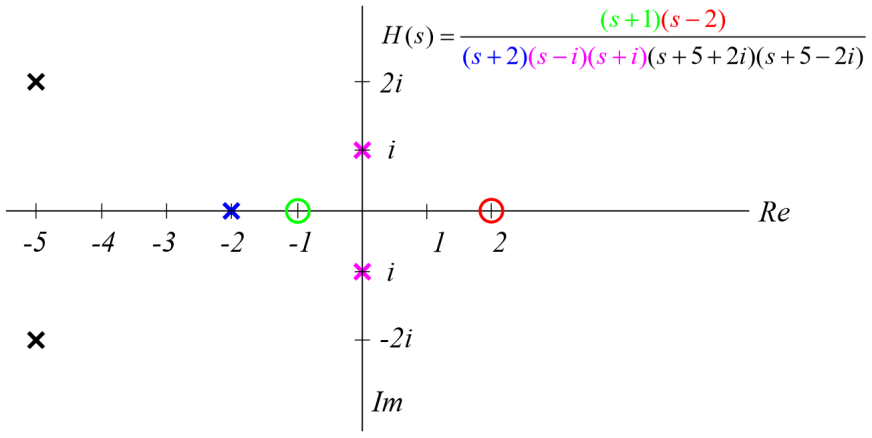


Figure 5.4: Poles and zeros.

Of course, we can also go the other way:

Example 2

Write down the transfer function you see here:

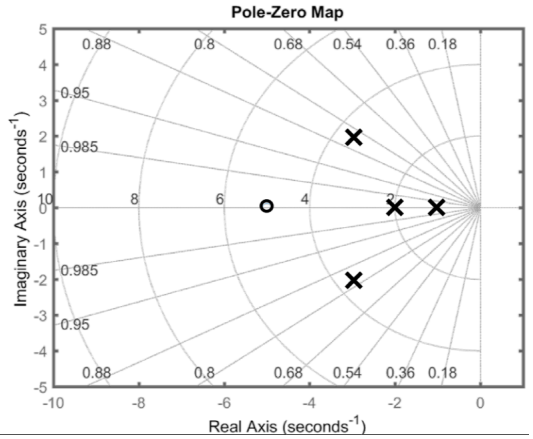


Figure 5.5: Poles and zeros of some transfer function.

We have a zero at $s = -0.5$, and poles at $s = 0$, $s = 0.5$, $s = -2 + 2i$ and $s = -2 - 2i$, meaning our original transfer function was

$$H(s) = \frac{s + 0.5}{s(s - 0.5)(s + 2 - 2i)(s + 2 + 2i)}$$

Now, these poles and zeros have a physical meaning: note that we can use partial fraction expansion to write

$$\begin{aligned} H(s) &= K\frac{(s - z_1)(s - z_2)\cdots(s - z_m)}{(s - p_1)(s - p_2)\cdots(s - p_n)} = K\left((s - z_1)(s - z_2)\cdots(s - z_m)\right)\frac{1}{(s - p_1)(s - p_2)\cdots(s - p_n)} \\ &= K\left((s - z_1)(s - z_2)\cdots(s - z_m)\right)\left(\frac{A_1}{s - p_1} + \frac{A_2}{s - p_2} + \cdots + \frac{A_n}{s - p_n}\right) \end{aligned}$$

So $Y(s)$ is given by

$$Y(s) = U(s)K\left((s - z_1)(s - z_2)\cdots(s - z_m)\right)\left(\frac{A_1}{s - p_1} + \frac{A_2}{s - p_2} + \cdots + \frac{A_n}{s - p_n}\right)$$

Now, note that $K\left((s - z_1)(s - z_2)\cdots(s - z_m)\right)$ just differentiates/scales $U(s)$; for example

$$\mathcal{L}^{-1}\left\{5s^2U(s)\right\} = 5\frac{d^2u(t)}{dt^2}$$

The other stuff is much more important:

$$\left(\frac{A_1}{s - p_1} + \frac{A_2}{s - p_2} + \cdots + \frac{A_n}{s - p_n}\right)$$

has inverse transforms

$$\mathcal{L}^{-1}\left\{\frac{1}{s - p_i}\right\} = e^{p_i t}$$

Now, this means a few things:

EFFECT OF POLES ON STABILITY

If $p < 0$, e^{pt} will be continuously decreasing, so the system will be stable. For example, if we have $p = -3$, then $\mathcal{L}^{-1}\left\{\frac{1}{s--3}\right\} = e^{-3t}$, which is clearly stable.

If $p = 0$, e^{pt} is constant: $\mathcal{L}^{-1}\left\{\frac{1}{s-0}\right\} = e^{0t} = 1$.

If $p > 0$, e^{pt} is increasing, and thus unstable. For example, if $p = 5$, then $\mathcal{L}\left\{\frac{1}{s-5}\right\} = e^{5t}$ which is continuously increasing and thus stable.

If p is complex, e^{pt} may be stable or unstable, but it is always oscillatory. For example, for $p = -2 + 4j$, we get

$$\mathcal{L}^{-1}\left\{\frac{1}{s-(-2+4j)}\right\} = e^{(-2+4j)t} = e^{-2t}e^{4jt} = e^{-2t}(\cos 4t + j\sin 4t)$$

Clearly, the value of the real part of p determines whether it is stable or unstable (using the same criteria as previously listed). The value of the imaginary part determines the frequency of the oscillations.

Regarding stability, I should specify a bit more what I mean with stability:

STABILITY

A system is **stable** if a bounded input leads to a bounded output.

A system is **unstable** if a bounded input leads to an unbounded output.

A system is **marginally stable** if an input leads to a pure oscillation.

This is all depicted in figure 5.6, which should clarify it a lot. Note that if $p = 0$, this means that the response is *marginally stable*. Furthermore, note that if one of the poles is unstable, then the entire system is unstable, no matter how many stable poles there are (an unstable component is always dominant compared to stable components of the response).

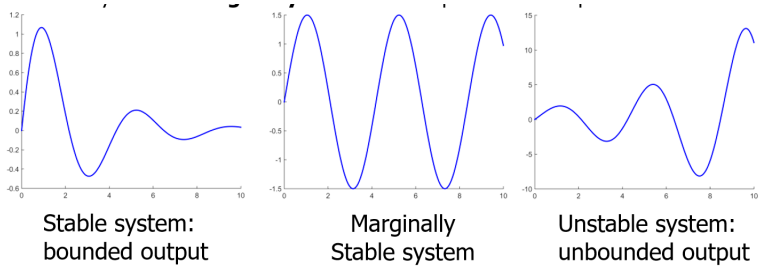


Figure 5.6: Stability of systems.

5.4 Properties of 1st order systems

Consider we have the following first order differential equation for the dynamics of a piston with cross-sectional area A and some constant k_v:

$$\dot{x}_p = \frac{k_v}{A}x_v$$

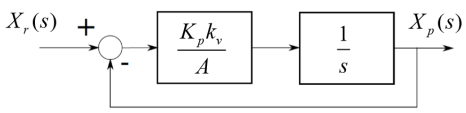
where x_v is some reference input, and $\dot{x}_p$ is the velocity of the piston (don't worry too much about how this piston exactly works, it won't help. Just remember that we relate the velocity of the piston to how much the valve is opened).

This leads to the Laplace transform

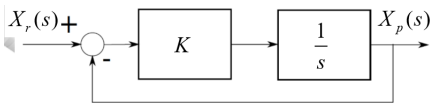
$$sX_p(s) = \frac{k_v}{A}X_v(s)$$

so that the transfer function is $H_{pv} = \frac{X_p(s)}{X_v(s)} = \frac{k_v}{As}$.

If we include a controller as well, we get the closed-loop system of figure 5.7a, which can be rewritten to the system of figure 5.7b, where $K = \frac{K_p k_v}{A}$.



(a) Closed-loop system of hydraulic actuator.



(b) Closed-loop system of hydraulic actuator.

Figure 5.7: Closed-loop system of hydraulic actuator.

Now, if you remember correctly, our equivalent transfer function will now be

$$H_{eq}(s)\frac{X_p(s)}{X_r(s)} = \frac{\text{feed forward path}}{1+\text{feedback path}} = \frac{\frac{K}{s}}{1+\frac{K}{s}} = \frac{1}{s/K+1} = \frac{1}{\tau s+1}$$

where we define

TIME CONSTANT

The time constant is given by

$$\tau = \frac{1}{k} \tag{5.3}$$

How does this time constant affect stuff? Well, the poles are given by $s = -1/\tau$, which leads to the sketch of figure 5.8. From what we've discussed so far, we realize that for negative τ, the system is unstable. For positive τ, the system is stable. Furthermore, we can compute the $x_p(t)$ using the inverse Laplace transform: suppose we have the step input function, $U(s) = \frac{1}{s}$:

$$X_p(s) = U(s)\cdot H_{eq}(s) = \frac{1}{s}\frac{1}{\tau s+1} = \frac{1}{s} + \frac{-\tau}{\tau s+1} = \frac{1}{s} - \frac{1}{s+1/\tau}$$

Since $e^{-at} \leftrightarrow \frac{1}{s+a}$, this leads to

$$x_p(t) = 1 - e^{-t/\tau}$$

We plot this response for various values of τ as shown in figure 5.9. Note that for large values of τ, the pole is close to 0, and the result is a slow response. On the other hand, for smaller values of τ, the result is a very fast response.

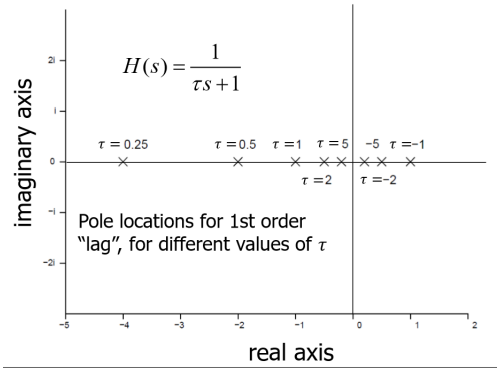


Figure 5.8: Location of the poles.

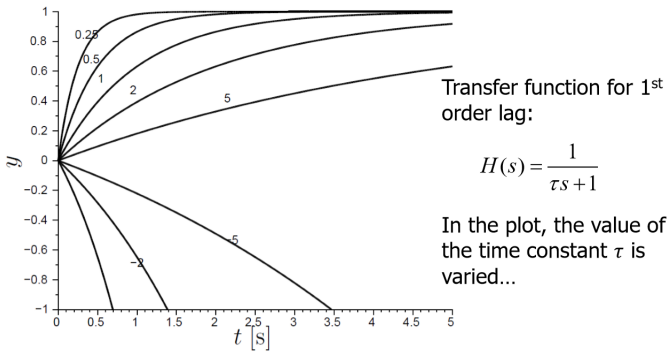


Figure 5.9: Response to input step function for various values of τ.

5.5 *Properties of 2nd order systems*

Consider the damped mass-spring system of figure 5.10. We straightforwardly get

$$\begin{aligned} m\ddot{y}(t) + B\dot{y}(t) + ky(t) &= u(t) \\ H(s) &= \frac{1}{ms^2 + Bs + k} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \end{aligned}$$

The poles occur when $s^2 + 2\zeta\omega_n s + \omega_n^2$, i.e.

$$p_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{(2\zeta\omega_n)^2 - 4\omega_n^2}}{2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$$

From the vibrations courses, you remember what happens for various values of ζ:

- If $0 < \zeta < 1$, then the system is underdamped: it oscillates around some value, but its damped.
- If $z = 1$, then the system is critically damped: it converges to some value, but does not oscillate around it.
- If $z > 1$, then the system is overdamped: it converges to some value and does not oscillate, but converges at a slower rate than the critically damped system.

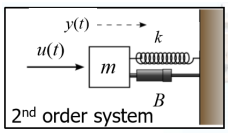


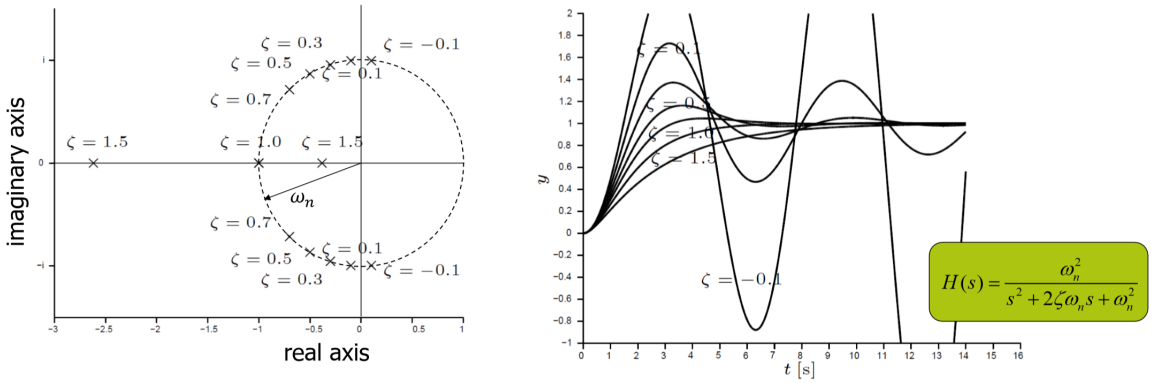
Figure 5.10: Mass-spring damper system.

Furthermore, look at figure 5.11a: we have plotted the poles for a fixed ω_n ($\omega_n = 1$) and various ζ. Note that for $\zeta \leq 1$, all poles are located on a circle with radius 1 in the left-half-plane; higher values of ζ get more and more real poles. Furthermore, once we go beyond $\zeta = 1$, the poles move away from the circle; one of them goes towards the origin, the other moves away. The effect of changing ζ is shown in figure 5.11b.

Similarly, looking at figure 5.12a, we see that if we keep ζ constant and increase ω_n, we move away from the origin on straight lines. The effect of changing ω_n is shown in figure 5.12b.

Above results can be summarized in figure 5.13.

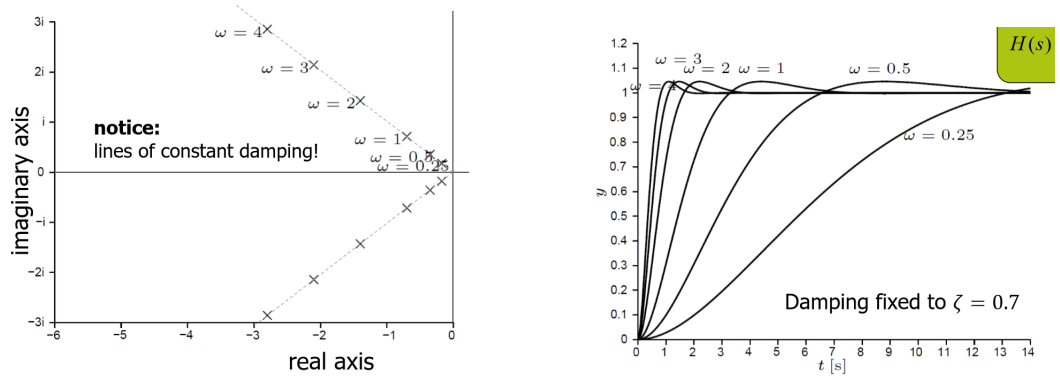
Finally, to summarize, look at figure 5.14: for various poles, the system response is shown. Note that for negative real parts, there is damping; if there are positive real parts, there is amplification. If there is an imaginary part, you get oscillations. You should be able to understand which graphs belong to which poles (had they been given in random order).



(a) Pole locations for $\omega_n = 1$, with varying damping coefficients ζ.

(b) Effect of varying ζ.

Figure 5.11: Fixed $\omega_n = 1$, but varying ζ.



(a) Pole locations for $\zeta = 0.7$, with varying natural frequencies ω_n.

(b) Effect of varying ω_n.

Figure 5.12: Fixed $\zeta = 0.7$, but varying ω_n.

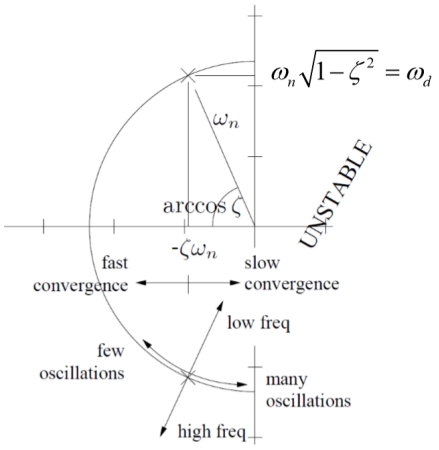


Figure 5.13: Properties of 2nd order systems.

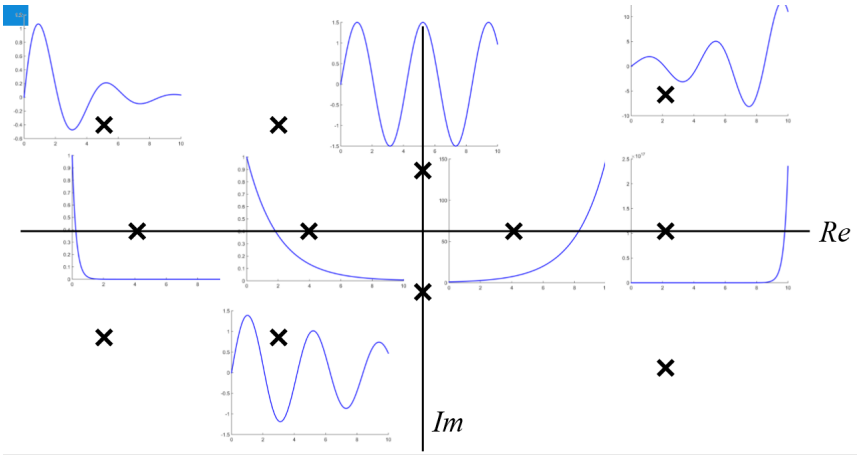


Figure 5.14: Characteristics of responses for different pole locations.

5.6 A closer look at poles and zeroes

5.6.1 Dominant poles

Higher order systems (like aircraft) often behave much like 1st ad 2nd order systems, due to the "dominant poles" of a system:

- For higher order systems, poles closest to the origin are "dominant".
- Distance from a pole p to the origin is measured as follows:

$$r_o = \sqrt{(\mathrm{Re}\,(p))^2 + (\mathrm{Im}\,(p))^2}$$

For example, consider the transfer function

$$\begin{aligned} H(s) &= H_1(s) H_2(s) \\ H_1(s) &= \frac{1}{s+1} \\ H_2(s) &= \frac{30^2}{s^2 + 2 \cdot 0.01 \cdot 30 s + 30^2} \end{aligned}$$

$H_1(s)$ has a pole at $s = -1$, which has a distance to the origin of $r_o = 1$. On the other hand, $H_2(s)$ has poles at $s = -0.3 \pm 30i$, which has a distance of $r_o = 29.8$ to the origin. We see the responses (to a unit step function) of both $H_1(s)$ (dotted red line below the black line), $H_2(s)$ (green line) and their product (black line) plotted in figure 5.15: clearly, $H_1(s)$ is much closer to the actual response of the product $H_1(s) H_2(s)$ than $H_2(s)$ is, so we say that $H_1(s)$ is dominant in the response.

5.6.2 Effect of zeros

Look at figure 5.16. Consider the two distinct control loops shown there: in the first one, we have (use $H(s)$ = feed forward path$/(1 +$ feedback path):

$$H_{eq,1}(s) = \frac{K_p}{s^2 + K_d s + K_p}$$

For the second one, we get

$$H_{eq,2}(s) = \frac{K_p + K_d s}{s^2 + K_d s + K_p}$$

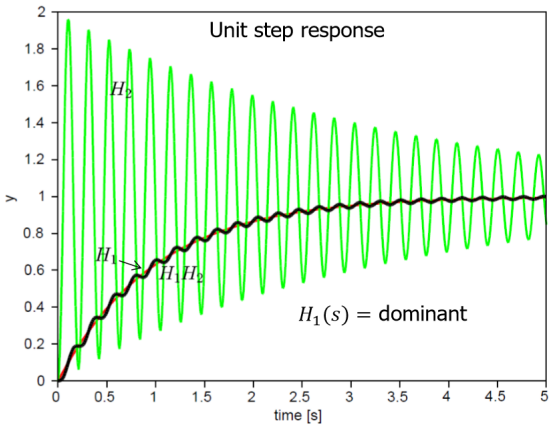


Figure 5.15: Effect of dominant poles.

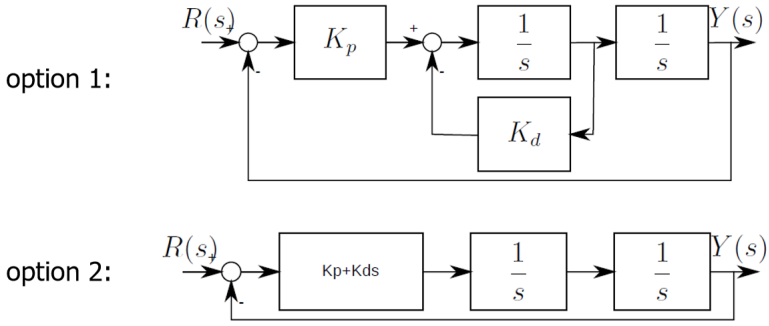


Figure 5.16: Two different control loops.

Note that the only difference is in the numerator. Note that the second one has an additional zero at $s = -K_p/K_d$. Indeed, we could write the second transfer function as

$$H_{eq,s}(s) = \frac{K_p}{s^2 + K_d s + K_p} + K_d s \cdot \frac{1}{s^2 + K_d s + K_p}$$

which looks like the block diagram shown in figure 5.17.

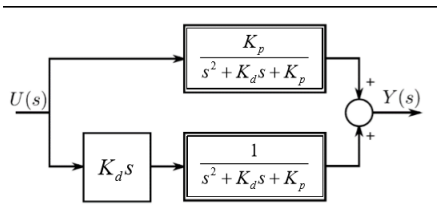


Figure 5.17: Second control loop redrawn.

What does this mean? The upper path of this system is the same as the first system of figure 5.16. However, the lower system is slightly different: remember that multiplication by s represents differentiation in the time domain: this means that the lower path basically can be seen as an additional term that "adds" the response of the derivative of an input signal. What do I mean? Suppose we have a step function as unit signal, then the response of this system will consist of two parts:

- The 'regular' response to a step function as you'd expect following the upper path in figure 5.17.
- The response to the derivative of the step function, i.e. the response to an impulse function, due to the lower branch of figure 5.17.

This is shown in figure 5.18. We see the effect of zeroes: it adds a "kick" to the original step response.

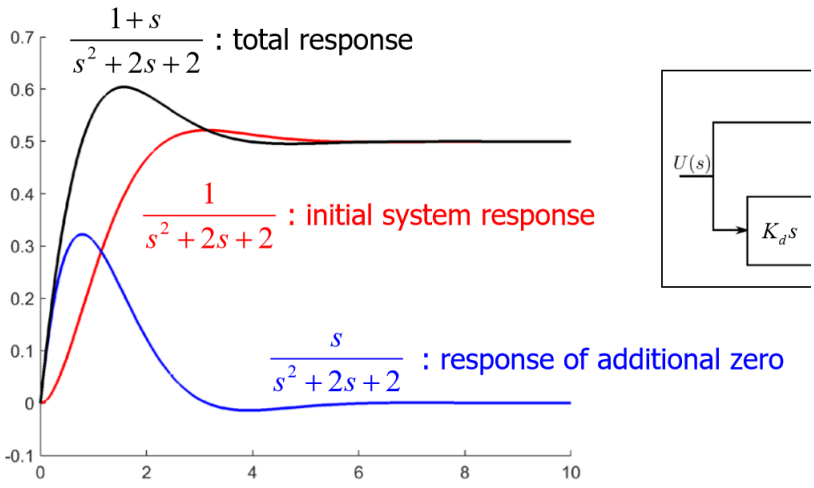


Figure 5.18: Response.

Now, what happens if we have a zero in the right half plane ($z_i > 0$)? Then the kick is reversed, as shown in figure 5.19: this is called a **non-minimum phase zero** (more on that in a later chapter).

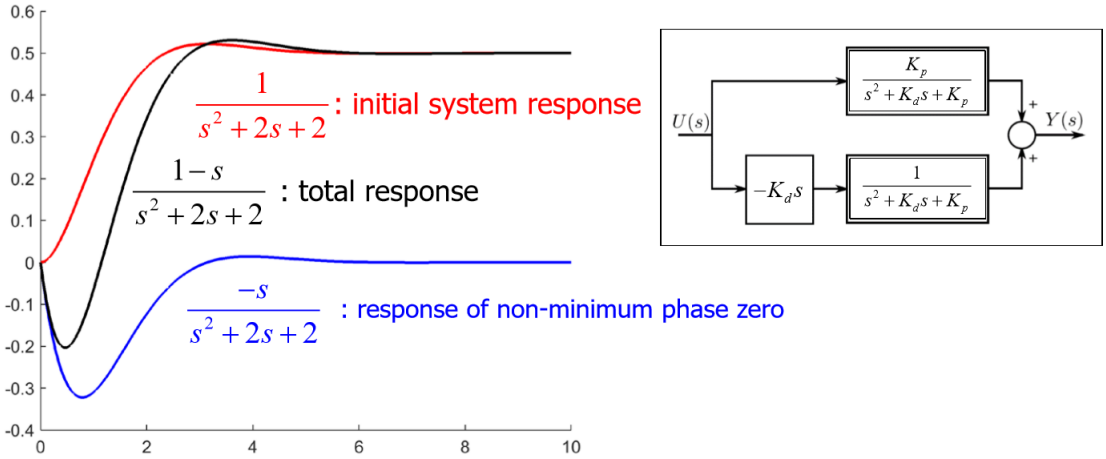


Figure 5.19: Response of non-minimum phase zero.

Furthermore, it is important to note: *zeroes that are placed close to poles neutralize the effect of those poles*.

5.6.3 *Summary stability*

For bounded input, a system produces a bounded response:

- If all poles are in the left half-plane, then the system is stable.
- If the poles are on the imaginary axis, then the systems are marginally stable.
- If the poles are in the right-half plane, then the systems are unstable.

There is one special case to consider: poles that lay in the origin: these are not marginally stable, but also produce an unbounded response. For example, consider the green input function of figure 5.20. You can calculate the for the transfer function

$$H(s) = \frac{1}{s^2}$$

the response will be like the blue line. Clearly, this becomes unbounded.

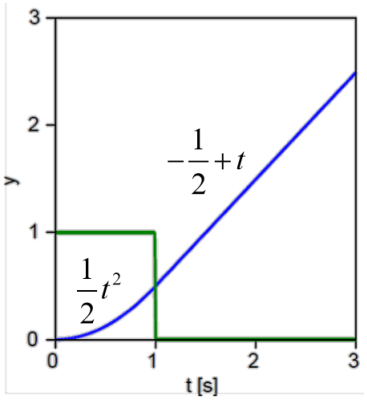


Figure 5.20: Response of system with double poles in origin to an unit pulse function.

5.7 *Type N systems*

Systems that have poles at the origin are even more interesting than that, however. Consider the simply closed loop system of figure 5.21.

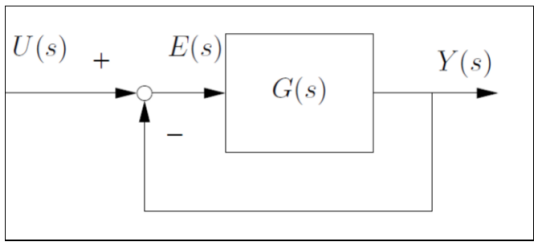


Figure 5.21: Simple closed loop control system.

Of course, over time, we want our control error, $e(t)$, to become zero over time. We can use Laplace final value theorem to determine what the **steady state error** will be for different 'basic' input functions. Consider the input functions shown in figure 5.22, with corresponding Laplace transform shown next to it.

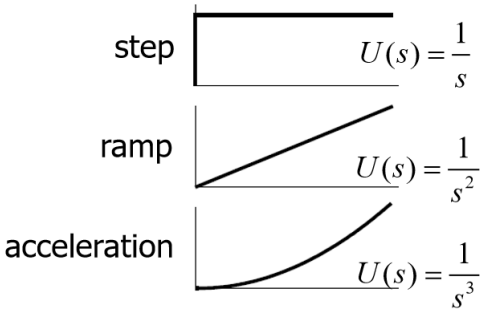


Figure 5.22: Some input functions. Note that the ramp function is the integral of the step function, and the acceleration function is the integral of the ramp function.

The control error is equal to $e(t) = y(t) - u(t)$. We will analyse three different systems:

$$\begin{aligned} G_1(s) &= \frac{1}{s+1} \\ G_2(s) &= \frac{1}{s} \\ G_3(s) &= \frac{1+1.5s}{s^2} \end{aligned}$$

Note that the first system has no poles in the origin, the second system has one pole in the origin and the third system has two poles in the origin. You can do the maths equations yourself, but we get the following results: in figure 5.23a (disregard type 0, type 1, type 2), we plot step input function $u(t)$ and the output function $y(t)$ for the three different systems as a function of time t. We see that:

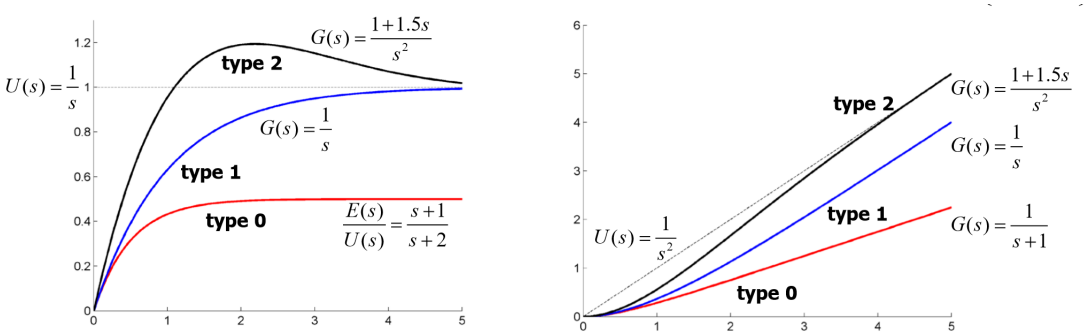
- $G_1(s)$ produces a constant steady-state error.
- $G_2(s)$ converges to the input function, meaning the steady-state error is zero.
- $G_3(s)$ converges to the input function, meaning the steady-state error is zero.

In figure 5.23b, we compare the ramp input function with the output functions for the three systems:

- $G_1(s)$ produces an ever-increasing steady state error.
- $G_2(s)$ produces a constant steady-state error.
- $G_3(s)$ converges to the input function, meaning the steady-state error is zero.

In figure 5.24, we compare the acceleration input function with the output functions for the three systems:

- $G_1(s)$ produces an ever-increasing steady state error.
- $G_2(s)$ produces an ever-increasing steady state error.
- $G_3(s)$ produces a constant steady-state error.



(a) Response to step function.

(b) Response to ramp function.

Figure 5.23: Response to step and ramp function. $G_3(s)$ in black, $G_2(s)$ in blue and $G_1(s)$ in red.

Clearly, we sense a pattern here: let me first define the following terms:

STEADY STATE ERRORS

We distinguish the following steady state errors:
- **Position error**: steady state error to a step input.
- **Velocity error**: steady state error to a ramp input.
- **Acceleration error**: steady state error of CL to an acceleration input.

TYPE N SYSTEM

A **type N system** has N poles in the origin, e.g. a type 2 system has 2 poles in the origin.

Then, from what we observed before, we can draw the following conclusion:

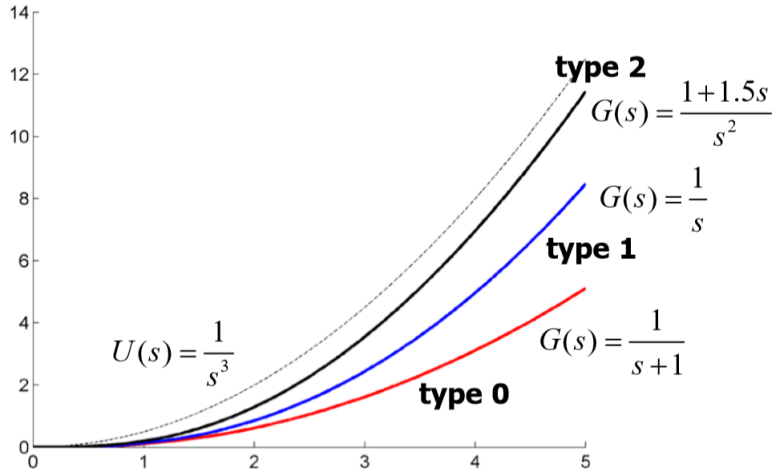


Figure 5.24: Response to acceleration function. $G_3(s)$ in black, $G_2(s)$ in blue and $G_1(s)$ in red.

STEADY STATE ERRORS FOR TYPE N SYSTEMS

A type 0 system has a nonzero steady state error.

A type 1 system has a zero steady state position error.

A type 2 system has a zero steady state position and velocity error.

A type 3 system would have a zero steady state position, velocity and acceleration error.

5.8 *Introducing PID Control*

PID control stands for **proportional + differential + integral control**. Remember that we saw in section 5.6.3 that the addition of a controller with gain $K_d s$ basically added another term in the total response of the system; it added the response to the derivative of an input signal. Indeed, we can use this to have a more sophisticated control loop. Consider figure 5.25.

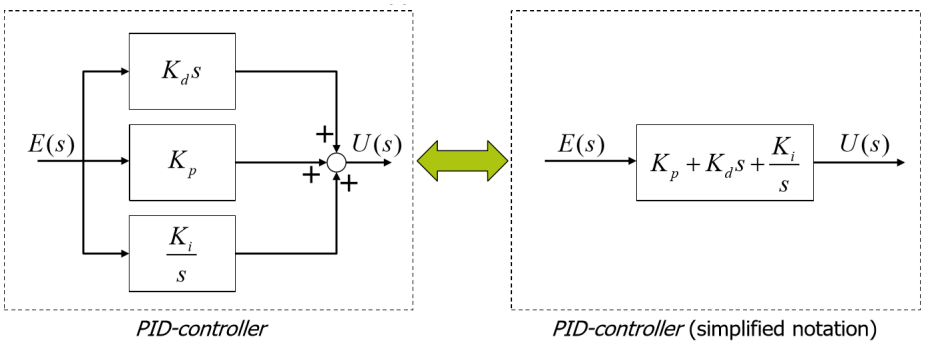


Figure 5.25: PID-controller.

We have a 'regular' **proportional controller**, with gain K_p. This only considers the actual value of the error at that time to determine what signal will be given to the system. We also have a **differential controller**, with gain $K_d s$. This considers the *rate of change of the error* to determine what signal will be given to the system. Finally, we have the **integral controller**, with gain K_i/s. This considers the integral (i.e. the history of the error) to determine what signal will be given to the signal. The effects of increasing each of the gains is shown in figure 5.26.

Gain (increasing)	Stability	Rise Time	Overshoot	Settling Time	Steady-State error
K_p	Degrades	Decreases	Increases	Small change	Decreases
K_i	Degrades	Decreases	Increases	Increases	Decreases significantly
K_d	Improves for small K_d	Minor change	Decreases	Decreases	No effect (theoretically)

Figure 5.26: Effects of altering gains of PID controller.

It should be obvious that this leads to a system where we have much more control over (although it comes at the cost of increased complexity).

6 Gain Tuning with Root-Locus

Now, so far, we haven't really discussed how you could find suitable values for controller gains in your system. However, as you can imagine, this is a rather vital part in designing your system. This chapter focusses on how to do that.

6.1 Gain tuning and the S-plane

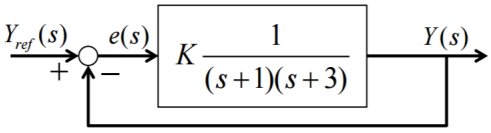


Figure 6.1: Feedback system.

Consider the feedback system shown in figure 6.1. The equivalent transfer function of this is

$$\frac{Y(s)}{Y_{ref}(s)} = \frac{\text{Feed forward path}}{1+\text{Feedback path}} = \frac{K\frac{1}{(s+1)(s+3)}}{1+K\frac{1}{(s+1)(s+3)}}$$
$$= \frac{K\frac{1}{(s+1)(s+3)}}{\frac{(s+1)(s+3)+K}{(s+1)(s+3)}} = \frac{K}{(s+1)(s+3)}$$

Thus, the poles are located at $p_{1,2}(K) = -2 \pm \sqrt{1-K}$: in other words, the pole locations are a function of the gain K. Indeed, we can plot the location of the poles for several values of K as shown in figure 6.2.

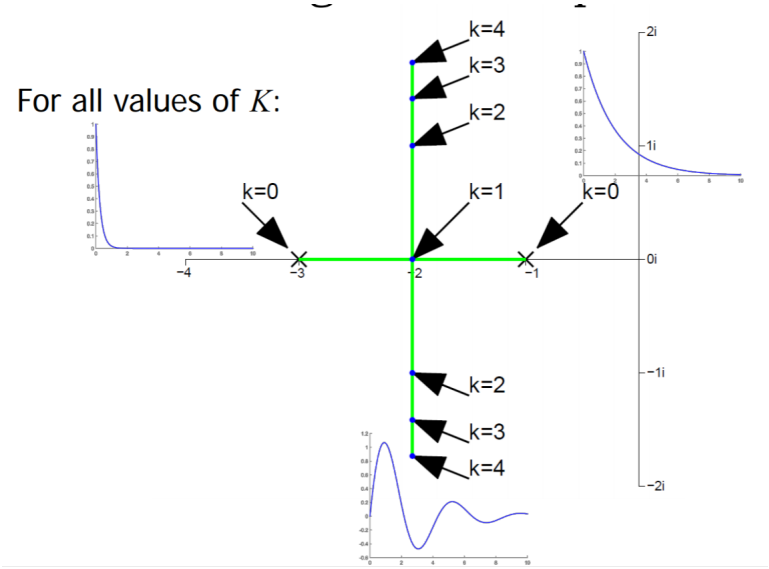


Figure 6.2: Root-locus plot.

Using figure 6.2, we can determine relatively easily which gain we want: the graphs can be easily constructed and from those you can choose your desired gain k (based on how fast decay you want and how much damping you want, etc.). Figure 6.2 is called a **root-locus plot**.

6.2 Evans root-locus plot

For second order systems (which have two poles), the pole locations can be calculated analytically. However, for more complex systems, there is a graphical method: **Evans' root-locus**. We can easily perform these with Matlab/Python, but you need to study these anyway, because you need to be able to predict how root-loci will look like (at least in the E-lectures).

First, consider the closed loop of figure 6.3.

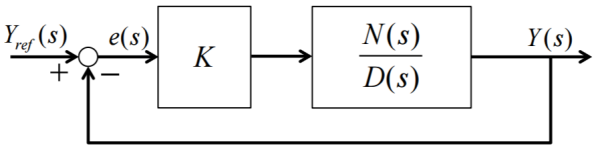


Figure 6.3: Feedback system: general case.

The equivalent transfer function of this closed loop system is clearly

$$H_{CL}(s) = \frac{Y(s)}{Y_{ref}(s)} = \frac{KN(s)/D(s)}{1 + KN(s)/D(s)} = \frac{KN(s)}{D(s) + KN(s)}$$

The poles are located at locations that satisfy $K\frac{N(s)}{D(s)} = -1$: this equation is called the **characteristic equation**. Solving this for a given K gives the poles of the closed loop system belonging to that gain. The paths followed by these roots for varying K are drawn in the complex plane; these paths are called **root-loci**. You then manually pick roots/poles in an acceptable location (based on arbitrary requirements), and you then calculate the gain K corresponding to these roots.

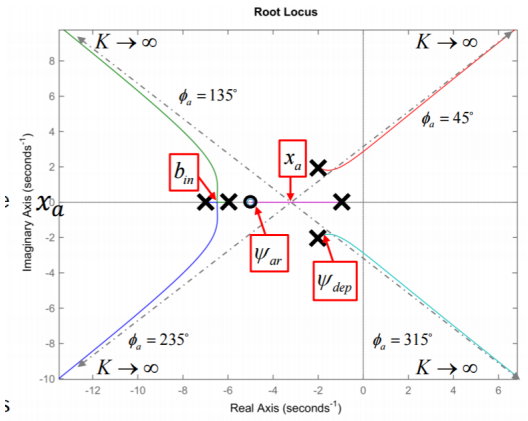


Figure 6.4: Root-locus plot of arbitrary system.

Consider, for example, the root-locus plot of a certain system, as shown in figure 6.4. It's a bit overwhelming at first, but let me guide you through it, step by step:

1. First of all, the indicated crosses and circles correspond to the poles and zeros respectively of the system when the gain would be $K = 0$.
2. Then, we steadily increase the gain K. The pole that was originally the most left pole then follows the path indicated by blue. The pole almost directly next to it follows the green path. The pole closest to the origin follow the purple path until it reaches the original zero. The other poles follow the red and light blue path respectively.
3. In programs like Matlab, you can then click on points on one of those lines (trust me, and we'll see it happening during the E-lectures), and then it tells you what gain corresponds to what point, and some of the properties (e.g. damping). So: constructing a root-locus plot does *not* immediately give you an exact solution for the gain K: using some trial and error, you have to click a few points, look what

their properties are, and determine yourself whether you find them desirable (for example, you may be interested in gains such that critical damping is achieved: you then click points and try to find a gain where critical damping is achieved).

Now, there are a grand total of seven rules that are important in constructing a root-locus plot (yes you don't have to construct one yourself, you have Matlab/Python for that, but it may be handy to read them at least once so that you know how it should look like):

1. Root loci only exist on the real axis (the horizontal axis) to the left of an odd number of real axis poles or zeroes. What do I mean with this exactly? Look at figure 6.4: on the real axis, the pole on the horizontal axis closest to the origin is the first pole/zero on the horizontal axis. The zero to the left of it is the second pole/zero on the horizontal axis. The pole to the left of that one is the third pole/zero on the horizontal axis and the pole to the left of that one is the fourth pole/zero on the horizontal axis. We see that there is a root locus to the left of the first pole/zero on the horizontal axis (namely the purple line), and to the left of the third pole/zero on this axis (namely the green and blue lines), but not to the left of the second pole/zero or fourth pole/zero. This is no coincidence: this is always the case: when counting from the right, there will only be root loci to the left of odd-numbered poles/zeroes (doesn't matter whether they are poles or zeroes).
2. For $K = 0$, the root-locus starts at the open loo poles.
3. For $K \to \infty$, if we have n poles and m zeroes, then m of the poles end up in the zeroes (so all zeroes get a pole, basically); the remaining $n - m$ poles follow the asymptotes to infinity.
4. The asymptotes intersect the real axis at the centroid x_a and leave at an angle ϕ_a, given by

$$x_a = \frac{\sum_{j=1}^{n} \mathrm{Re}\left(p_j\right) - \sum_{i=1}^{m} \mathrm{Re}\left(z_i\right)}{n-m} \tag{6.1}$$

$$\phi_a = \frac{2q+1}{n-m} \cdot 180°, \qquad q = 0, 1, 2, ..., n-m-1 \tag{6.2}$$

   where q is the asymptote counter (i.e. the how manyth asymptote you're at; we have $n - m$ asymptotes).
5. From the complex plane, the root-locus enters the real axis in break-away points (or break in points, different name for same thing), as denoted by b_{in} in figure 6.4. You can calculate these using Matlab/Python.
6. The root-locus leaves the poles with an angle of departure at the poles, and arrives at the zeros with an angle of arrival. You can calculate these using Matlab/Python.
7. The gain at each point of the root-locus is calculated with the magnitude condition: we must have

$$|K| \frac{|N(s)|}{|D(s)|} = |-1| \tag{6.3}$$

   Again, just trust Matlab/Python to do this for you. We'll see plenty enough in the E-lectures on this.

If you're interested, you can look at the slides on how to sketch your own root-locus plot in case you want to give your parents to be proud of. Otherwise, don't bother since you're gonna use Matlab/Python on the exam for this.

6.3 *Gain tuning with root-locus*

How exactly does one go about tuning gains with root-locus? Consider again our taxiing aircraft, as shown in figure 6.5. Our (only) aim is to critical or overdamping ($\zeta \geq 1$) for the entire system (as long as it's not underdamped or undamped).

Now, something important is that root-locus can only tune one gain at a time. Thus, we first need to tune the inner loop; afterwards we'll bother with the outer loop.

We first focus on the control loop of figure 6.6. What is absolutely quintessential to remember is that both for the Matlab command `rltool()` and for the Python command `rlocus()`, you have to input the *open-loop* transfer function. Thus, our input will be the transfer function

$$H_{OL}(s) = K \cdot \frac{V}{l} \cdot \frac{1}{s} \cdot \frac{1}{0.1s+1} = K \cdot \frac{V}{0.1ls^2 + ls}$$

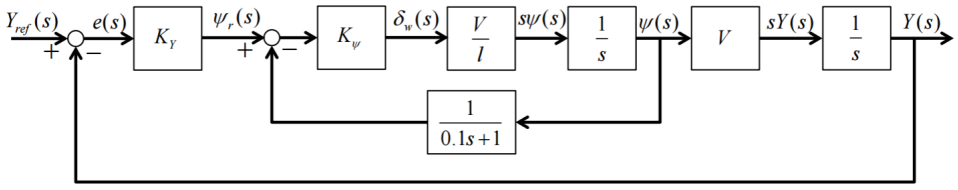


Figure 6.5: Feedback system of taxiing aircraft.

Please, please input the open loop transfer function and not the closed loop, otherwise you'll look like a complete idiot on the exam and your parents will want to disown you afterwards.

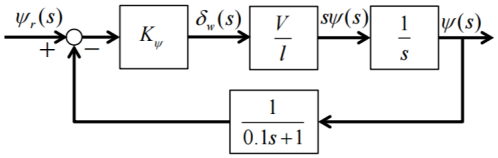


Figure 6.6: Inner loop.

Let's give values to V and l, so say $V = 20$ and $l = 5$. Finally, when performing root-locus stuff, you have to set $K = 1$ in the transfer function. In other words, you would first define

$$H_{OL}(s) = \frac{20}{0.5s^2 + 5s}$$

in Matlab/Python (shouldn't be difficult at all); then you write `rltool(H_OL)` (Matlab) or `rlocus(H_OL)` (Python).

Now, you have to trust me a bit (or you should program it yourself, but not really necessary in my opinion): when you do this, then it'll produce a plot like figure 6.7 (in Python). Then, you can use your mouse to click on points on the root-loci (the blue lines); you'll then see the associated gain and damping ratio ζ appear somewhere (I think in the command window, at least somewhere where you could easily read it, he showed it during the lecture). Again, to reiterate: we'll see plenty enough of examples in the E-lectures of root-locus stuff, so just agree with me for now that this works.

Clicking multiple points will point to the conclusion that with $K = K_\psi = 0.6$, we achieve overdamping for the inner loop (achieved by clicking on any point on the real axis: remember section 5.5, where we discovered that for a 2nd order system, if the poles lay on the real axis, then the damping ratio was $\zeta \geq 1$). We clicked on one of the neonpink dots in figure 6.7.

Now that we have $K_\psi = 0.6$, we can modify the system as shown in figure 6.8a and 6.8b.

Again, don't be that despicable person who uses the closed loop transfer function, but use the open loop function; again, we use $V = 20$, $l = 5$ and we set $K = 1$:

$$H_{OL}(s) = K \cdot \frac{1.2s + 0.6V}{0.1ls^2 + ls + 0.6V} \cdot V \cdot \frac{1}{s} = K \cdot \frac{1.2Vs + 0.6V^2}{0.1ls^3 + ls^2 + 0.6Vs} = 1 \cdot \frac{24s + 240}{0.5s^3 + 5s^2 + 12s}$$

Again, we get the nice plot of figure 6.9 (you have to code a little extra to get the lines of constant damping and circles of natural frequency, but we'll see that in the E-lecture). By clicking points on the root loci, you'll come to the conclusion that $K_Y = 0.04$ is a good gain. Again, don't worry about writing the program script for this yourself, we'll see it in the E-lectures.

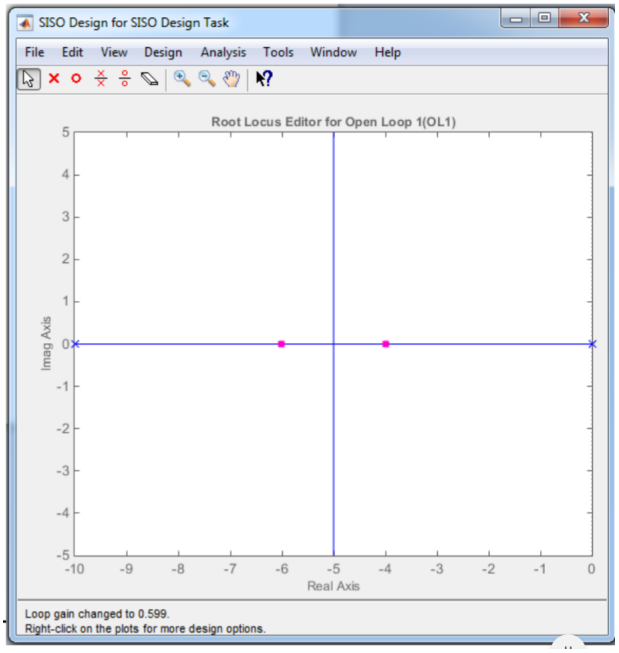
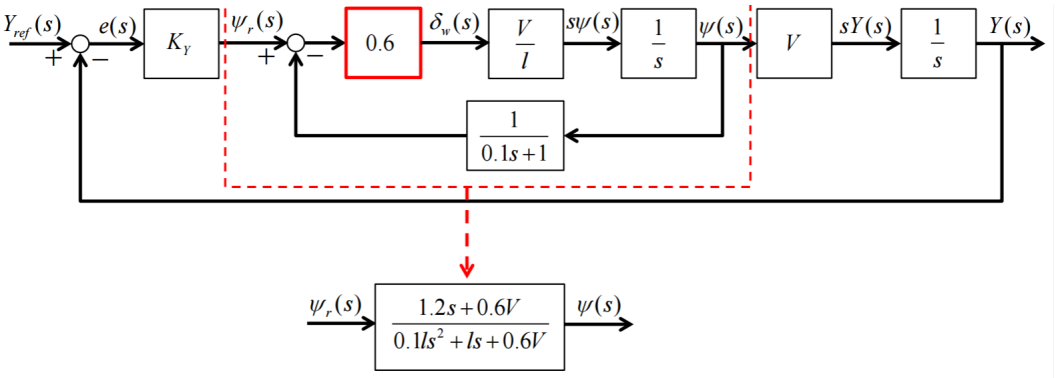
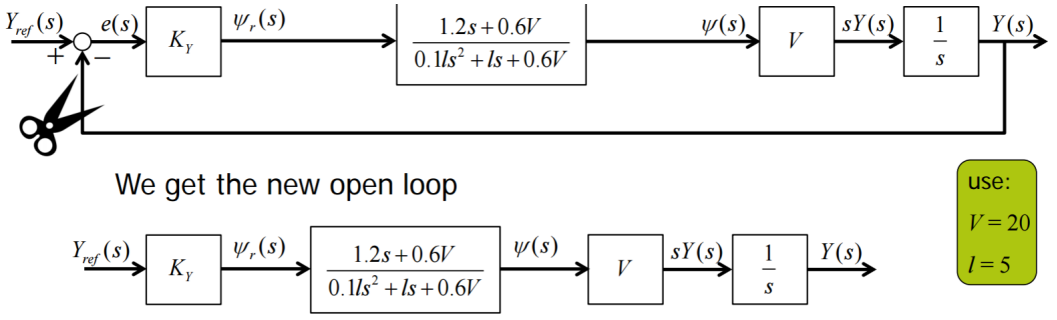


Figure 6.7: Root-locus: you can click on any point on the blue lines; that point will then turn neonpink. You then see below what the gain is that is associated with this point.



(a) Rewriting the transfer loop.



(b) Rewriting the transfer loop.

Figure 6.8: System of coupled equations.

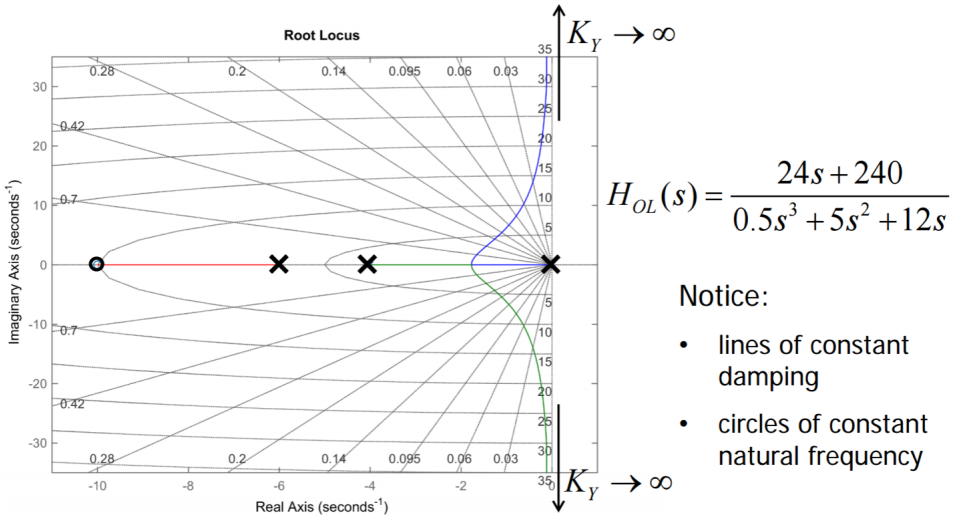


Figure 6.9: Root-locus: the straight lines are lines of constant damping. The ellipses are circles of constant frequency. These can be introduced to the plot by simple Matlab/Python functions.

6.4 *Zeros "pull" the root-locus*

Yeah something totally interesting happens with zeros in the vicinity of root-loci. Consider the transfer functions

$$H_1(s) = \frac{s+0.5}{s^4+4s^3+6.25s^2}$$
$$H_2(s) = \frac{s+0.4}{s^4+4s^3+6.25s^2}$$
$$H_3(s) = \frac{s+0.3}{s^4+4s^3+6.25s^2}$$
$$H_4(s) = \frac{s+0.2}{s^4+4s^3+6.25s^2}$$

H_1 is plotted in figure 6.10a, H_2 in figure 6.10b, H_3 in figure 6.11a and H_4 in figure 6.11b. Clearly, we see the zeroes moving to the right. Meanwhile, the left root loci is pulled towards the right as well, and the right root loci is also pulled downwards a bit. In figure 6.11b, everything breaks down.

this kind of behaviour is very hard to predict by hand, so we just use Matlab/Python for this stuff.

6.5 *Tuning 'nasty' loops*

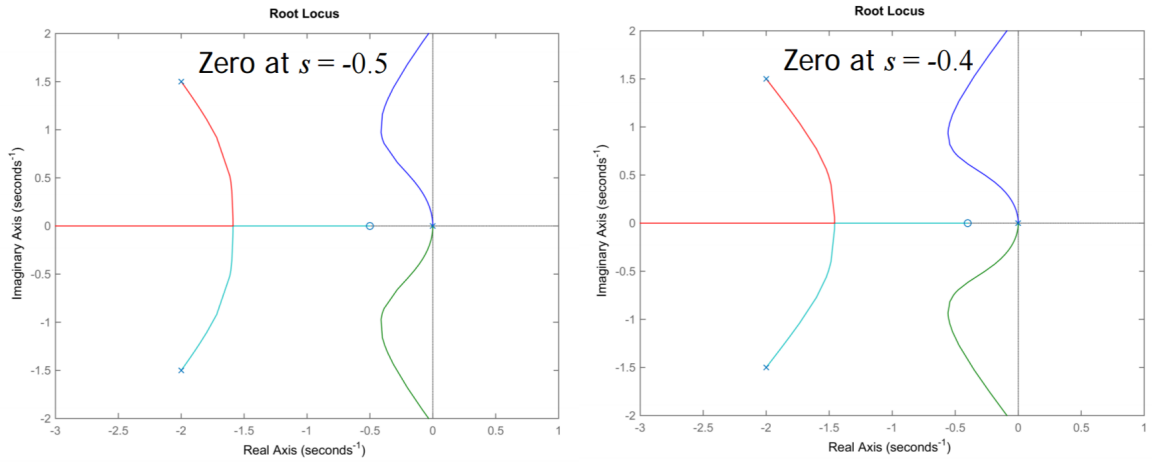
We previously saw what we had to do if we had two controller gains, but in that case, we still had a clearly distinguishable inner and outer loop. But what if that isn't the case?

Consider the 'nasty' system shown in figure 6.12a. Clearly, we can't separate an inner and outer loop this time. Instead, what we'll do is first assume a fixed value for K_r (say $K_r = 0.9$), and then we'll combine the K_r and $1/s$ blocks to what is shown in figure 6.12b.

The open-loop (not closed-loop) for tuning K will then be

$$H_{OL}(s) = \frac{20}{(s+1)(s+4)}\left(K_r + \frac{1}{s}\right)$$

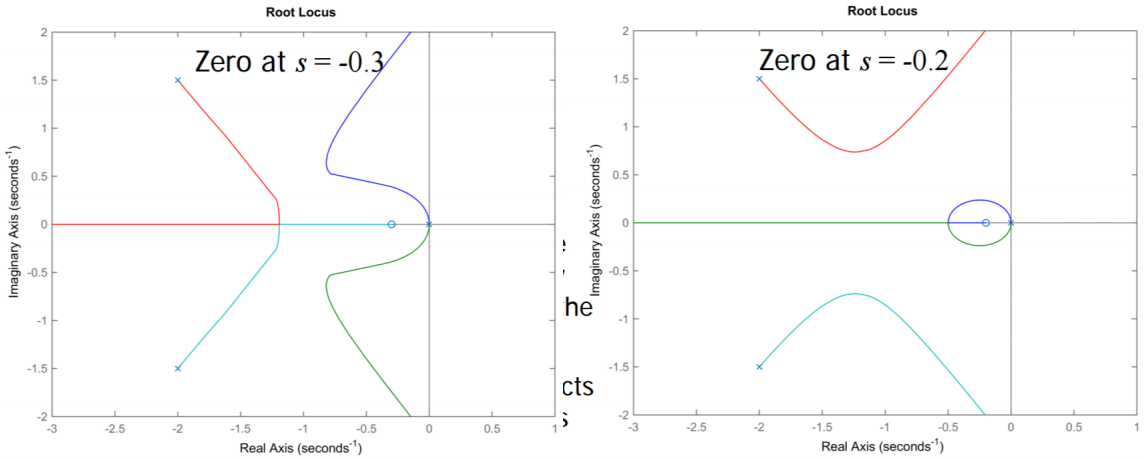
where K_r is simply a constant, user-defined value (in our case $K_r = 0.9$). The associated root-locus plot is then shown in figure 6.13.



(a) Zero at $s = -0.4$.

(b) Zero at $s = -0.4$.

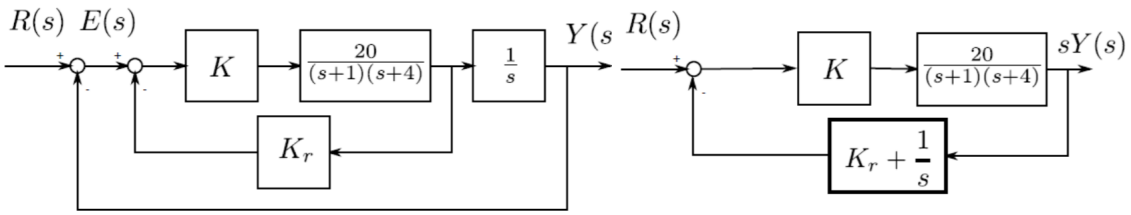
Figure 6.10: System of coupled equations.



(a) Zero at $s = -0.3$.

(b) Zero at $s = -0.2$.

Figure 6.11: Zeros pulling the root-locus.



(a) Nasty loop.

(b) Rewriting the nasty loop.

Figure 6.12: System of coupled equations.

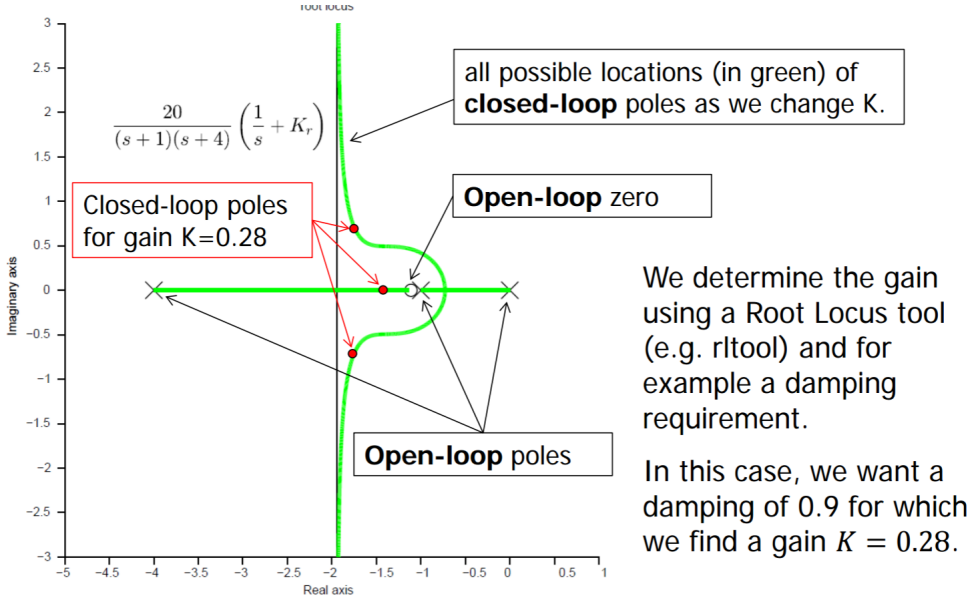


Figure 6.13: Root-locus.

If we then want a damping of 0.9, then we find a gain of $K = 0.28$ (again, just click some points until ou get close to this damping value).

We then redraw the system as shown in figure **??**: the equivalent transfer function we will use is then

$$H_{OL}(s) = \frac{20Ks}{s(s+1)(s+4)+20K}$$

Note that for the internal closed loop, you of course *do* take the closed loop transfer function. This gives the root locus of figure 6.14, and to get a damping of 0.9, we get a gain of $K_r = 0.90$.[1]

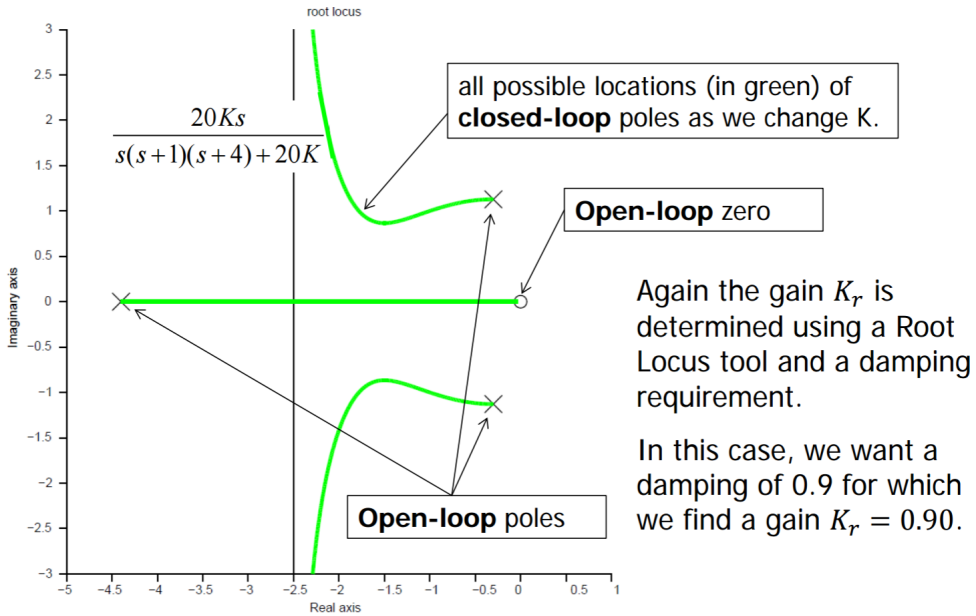


Figure 6.14: Root-locus.

[1] Honestly, I don't understand what the fucking point is of this step. We previously assumed $K_r = 0.90$ and set K such that we got a damping of 0.9. Of course, if we then check which value K_r should be to get a damping of 0.9 for that value of K, then of course it'll return $K_r = 0.90$. Maybe someone else can enlighten me on why this wouldn't be a huge waste of time.

6.6 *Controllers for State-Space Systems*

If you remember correctly from a few chapters back, for state-space systems, we had a controller gain*matrix*, so how do we tune that? Well, actually pretty easy: consider the block diagram of figure 6.15. We see that all gains are zero, except for the gain that compares the third output with the second input. In other words, we should take the transfer function

$$\frac{Y_3(s)}{U_2(s)}$$

Using the following straightforward piece of code, we can do it using the code shown in figure 6.16. Selecting a transfer function from a state-space system should really be new to you; you already did this in E-lecture 6.2.

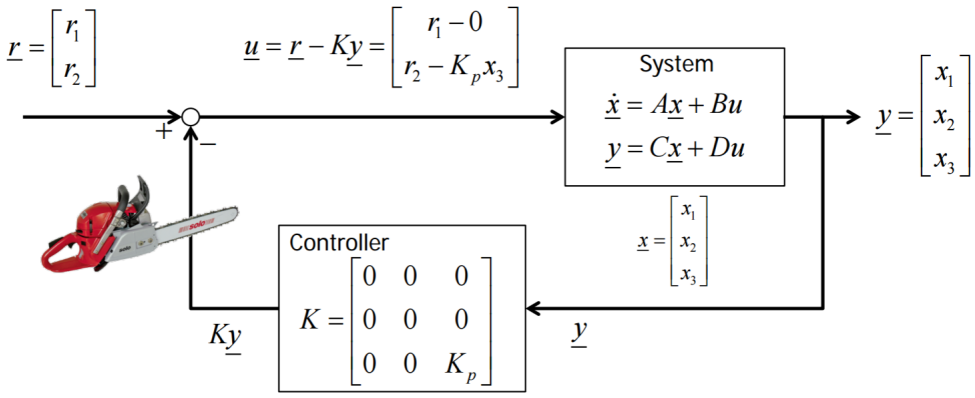


Figure 6.15: State-space system.

Python	Matlab

`sys=ml.ss(A,B,C,D)`
`Hol=ml.tf(np.matrix([[0,0,1]])*sys*np.matrix([[0],[1]]))`
`ml.rlocus(Hol)`
	`sys=ss(A,B,C,D);`
	`TFset=tf(sys);`
	`Hol=TFset(3,2);`
	`rltool(Hol);`

Figure 6.16: Code.

Index