ECEN415
Advanced Control Engineering

Christopher Hollitt
School of Engineering and Computer Science
Victoria University of Wellington

Trimester Two, 2021
Revised September 1, 2021

Contents

1 Classical Control in Continuous Time

1.1 Modes and the s-plane

We typically describe the operation of a continuous time, linear time invariant (LTI) system in the Laplace domain. The *transfer function* describes how the input is changed into the output.

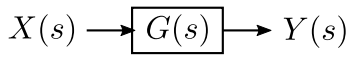


Figure 1.1: A transfer function representation of a system.

$$\text{where, } \begin{cases} X(s) := \mathcal{L}\{x(t)\} & \text{where } x(t) \text{ is the input signal} \\ Y(s) := \mathcal{L}\{y(t)\} & \text{where } y(t) \text{ is the output signal} \\ G(s) := \frac{Y(s)}{X(s)} \end{cases}$$

We have dealt exclusively with transfer functions that are rational functions of s. That is, they are fractions where the numerator and denominator are each polynomials in s.

$$G(s) = \frac{n(s)}{d(s)} = \frac{n_m s^m + \ldots + n_2 s^2 + n_1 s + n_0}{d_n s^n + \ldots + d_2 s^2 + d_1 s + d_0}$$

In real systems we typically have *strictly proper* ($m < n$) transfer functions. However, we will later see that is sometimes useful to think of transfer functions as having an equal number of poles and zeros, with the excess roots located at (complex) infinity. You may have encountered this idea before when dealing with root locus diagrams. In that context it is common to describe branches of the root locus as terminating on finite *or infinite* zeros. The poles of the transfer functions are extremely useful in understanding the behaviour of the system. To find them we equate the characteristic polynomial (the denominator polynomial) to zero.

$$\{\lambda_i\} = \{s | \overbrace{\underbrace{d_n s^n + \ldots + d_2 s^2 + d_1 s + d_0}_{\text{Characteristic polynomial}} = 0}^{\text{Characteristic equation}}\}$$

Remember that there will be exactly n values of s that satisfy this equation, though they need not be distinct.

In principle, we can decompose the transfer function into first order terms, each of which describes the effect of a single pole.

$$G(s) = \sum_{i=1}^{n} \frac{\rho_i}{s + \lambda_i} \qquad \lambda_i \in \mathbb{C}$$

where $\rho_i \in \mathbb{C}$ here are the residuals found by partial fractions expansion. Typically we don't completely decompose transfer functions in this way, but are content with leaving second order terms,

$$\frac{\rho_1 s + \rho_0}{(s+\sigma)^2 + \omega_{\mathrm{d}}^2} \equiv \frac{\rho_1 s + \rho_0}{s^2 + 2\zeta\omega_{\mathrm{n}} s + \omega_{\mathrm{n}}^2}$$

Remember also that a pole repeated k times led to a cascade of terms in the expansion, such as

$$\frac{N(s)}{(s+\lambda)^k} \rightsquigarrow \frac{\rho_k}{(s+\lambda)^k} + \frac{\rho_{k-1}}{(s+\lambda)^{k-1}} + \cdots + \frac{\rho_2}{(s+\lambda)^2} + \frac{\rho_1}{s+\lambda}$$

The three main types of poles each lead to a characteristic form of system time response, called a mode.

$$\begin{aligned}
\frac{1}{s+\lambda}, \quad & \lambda \in \mathbb{R} && \overset{\mathcal{L}}{\Longleftrightarrow} && \mathrm{e}^{-\lambda t} \\
\frac{1}{(s+\lambda)(s+\lambda^*)} \quad & \lambda = \sigma + \mathrm{j}\omega \in \mathbb{C} && \overset{\mathcal{L}}{\Longleftrightarrow} && \mathrm{e}^{-\sigma t}\cos(\omega t + \phi) \\
\frac{1}{(s+\lambda)^k}, \quad & \lambda \in \mathbb{R} && \overset{\mathcal{L}}{\Longleftrightarrow} && t^{(k-1)}\mathrm{e}^{-\lambda t}
\end{aligned}$$

The general properties of the modes can be inferred from the position of the corresponding position of the poles in the s-plane. Figure 1.2 illustrates the qualitative effect of moving a complex pair of poles in the s-plane. Note in particular, that the real part of the pole location determines the rate with which the mode decays and that the imaginary component determines the frequency at which the damped mode will oscillate.

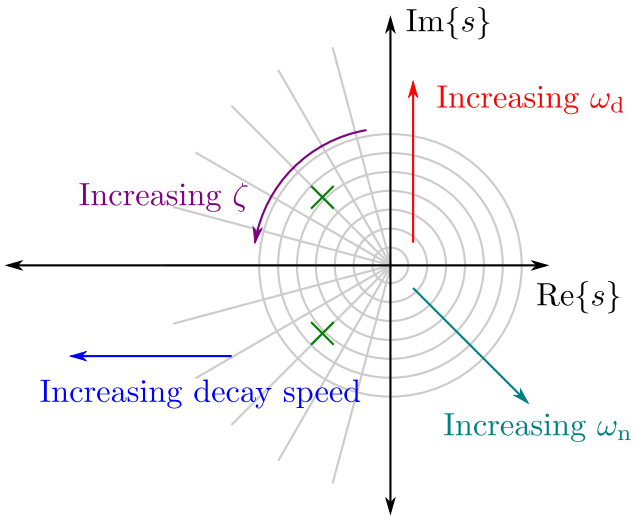


Figure 1.2: Qualitative behaviour of modes corresponding to a pair of complex conjugate poles in the s-plane.

We are often given specifications on both the settling time and damping that a system must exhibit (for example). We can therefore designate a region of the s-plane within which we need to place the dominant system poles. A typical example of this is shown in figure 1.3.

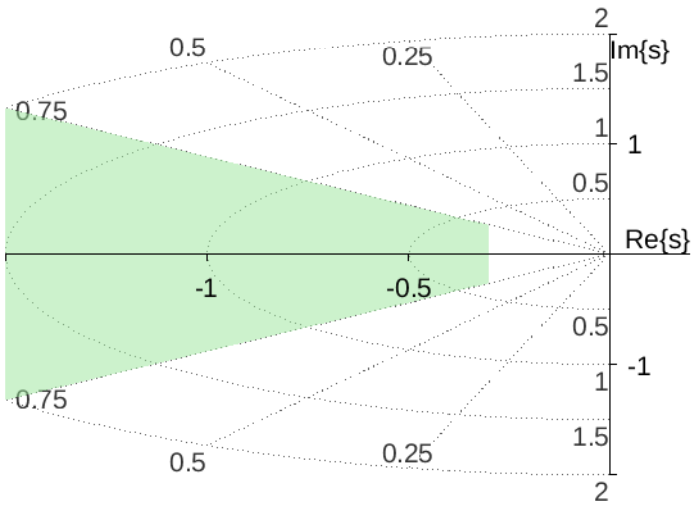


Figure 1.3: Acceptable dominant pole locations for a system having both a settling time and a damping constraint.

Remember that this region will only be accurate for second order systems for no zeros. The presence of more poles or of zeros changes the time response of the system away from the simple decaying exponential form used in deriving the equations that we use to find the correspondence between s plane pole locations and the time response.

We can be comfortable using an approximation to a second order system if there are additional poles, but those poles are sufficiently fast compared to the dominant pair. That is, we want the mode(s) produced by additional poles to decay at least ten times faster than the dominant mode.

The mechanism we use to alter the position of the system poles is *feedback*. Figure 1.4 shows a general arrangement for a feedback control system. We can use this to calculate the loop transfer function of the closed loop system, $T(s)$.

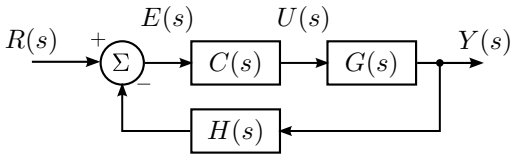


Figure 1.4: A simple feedback system, using both a forward compensator with transfer function $C(s)$ and feedback compensator with transfer function $H(s)$.

$$\begin{aligned} Y(s) &= C(s)G(s)E(s) \\ &= C(s)G(s)\big[R(s) - H(s)Y(s)\big] \\ &= C(s)G(s)R(s) - C(s)G(s)H(s)Y(s) \\ Y(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)H(s)}X(s) \\ \implies T(s) := \frac{Y(s)}{R(s)} &= \frac{C(s)G(s)}{1 + C(s)G(s)H(s)} \end{aligned}$$

Notice that feedback moves the poles of the system, so we can alter the system response.

That is, the values of s that make $1 + C(s)G(s)H(s)$ zero will not, in general, be the same as the values of s that make the denominator of $G(s)$ zero. By changing either $C(s)$ and/or $H(s)$ we can move the closed loop poles away from their open loop locations.

For example, we can move the poles of a dc motor having $G(s) = \dfrac{1}{s(s+1)}$ and proportional control, $C(s) = K$, as shown in figure 1.5. The path taken by the closed loop pole locations as we vary the

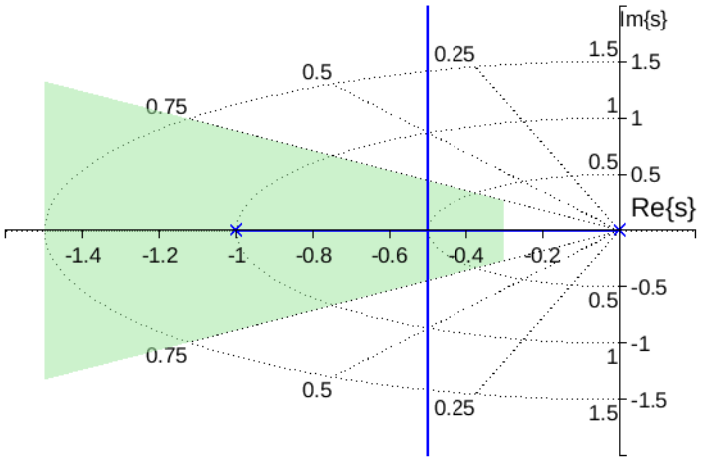


Figure 1.5: Root Locus diagram for a system $G(s) = \dfrac{1}{s(s+1)}$

feedback gain is called the *root locus*.

In this example, it is possible for us to choose a feedback gain that places the closed loop system poles within the desired region. It is therefore possible to satisfy the requirements on the system. (Note that this is a somewhat simple-minded example, as in a real design problem we typically also have constraints on the allowable gain so that we can achieve certain steady state error, or disturbance rejection requirements.) There are occasions when we would not be able to move the poles where we want. In such a case we would need to move the branches of the root locus around by introducing extra poles and/or zeros. This is why we often use a compensator more complex than a simple proportional gain).

1.2 Stability

One of the critical factors of control system design is ensuring that our controllers ensure that the final system remains stable at all times. We must therefore be very careful that the closed loop poles remain in the left half of the s-plane, as all modes corresponding to these pole locations will eventually decay. We also saw that the Bode plot could often be used to assess the stability of a system by looking at the gain and phase margins. These margins tell us (respectively) how much the gain can be increased, or the the phase decreased before the system becomes unstable. However, using a Bode system can be misleading for systems having roots in the right half of the s-plane, or for systems that have multiple crossings of unity gain, and/or -180° of phase.

1.3 Nyquist Diagrams

The Nyquist diagram provides a simple, universal, graphical method for assessing the stability of SISO systems. It works for simple systems that are managed with a Bode plot, but also for more complicated systems. It can be used when we have a known transfer function, but also if we only have experimental data, although we also need to know how many roots the open loop system has in the right half of the s-plane (though not where they are necessarily). The root locus and Routh-Hurwitz techniques also provide methods for determining stability, but the Nyquist diagram has the advantage of being applicable when you don't have an a mathematical description of your system (you can use it on experimental data).

1.3.1 Stability from the Nyquist Plot

Recall that a system is stable if and only if it has no poles in the right half of the s-plane. We seek a method that will tell us whether a system will be stable once we enclose it in a feedback loop - we want to know about the stability of a closed loop system. The Nyquist plot is a graphical technique that enables us to determine whether a system has *closed loop* poles in the the right half of the s-plane by examining its *open loop* poles. (In fact it does more - it counts the *number* of closed loop poles in the right half plane.)

1.3.2 The Nyquist Plot

A Nyquist plot is a plot of the real part vs the imaginary part of an open loop transfer function. Equivalently, you can think of it as a polar plot of a transfer function.

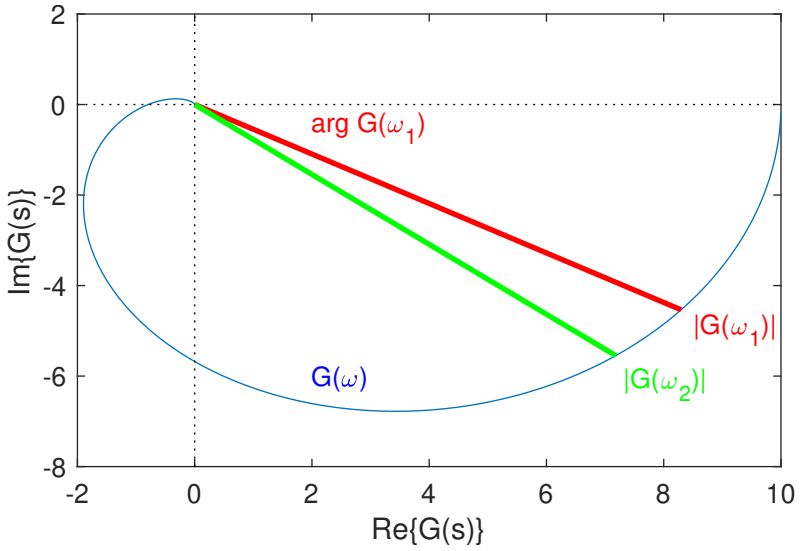


Figure 1.6: An example of the complex gain of a system plotted with polar coordinates. The frequency at which the gain is calculated is varies along the blue contour, with the red and green vectors showing the gain at two specific frequencies, ω_1 and ω_2.

The graph's axes are linear rather than logarithmic. This makes the plot awkward for visualising the entire behaviour of a system that has high gain regions. The Bode or Nichols plots are more useful for that task. The Nyquist plot is specialised for considering system stability, as it focuses on the low gain region around unity gain. We will examine two ways to construct a Nyquist plot

1. Using a Bode plot or frequency response,

2. Directly from a pole-zero map.

1.3.3 From the Bode to the Nyquist Plot

There are two commonly used approaches to hand plotting a Nyquist curve. If you have access to a Bode plot then it is reasonably straightforward to transfer the gain and phase information to a Nyquist plot. Alternatively, it is possible to plot the Nyquist diagram directly by considering the geometry of the pole zero diagram.

The curve on a Nyquist plot can be determined by choosing gain-phase points from a Bode plot at multiple frequencies. The Nyquist plot is the locus traced out by the transfer function as we vary frequency. Frequency thus varies along the Nyquist curve, but not in a regular way - there is no way to use a Nyquist plot to determine the frequency at which something happens. To construct the Nyquist plot, choose the points where something "interesting" happens on the Bode plot and transfer them to the Nyquist plot. Join the points with sensible curves. This produces a plot of gain vs. phase as the frequency varies from 0 to ∞.

Drawing a Nyquist diagram is slightly more complex than plotting the gain vs phase curve for $f = 0 \rightarrow \infty$. Nyquist's criterion (see later) actually requires that we evaluate the transfer function as we traverse a clockwise path that completely encloses the right half of the s-plane. Evaluating the transfer function for $f = 0 \rightarrow \infty$ corresponds to traversing the upper radius of the semicircle shown in the diagram. As the pole-zero diagram must be symmetric we know that the Nyquist plot must be symmetric in the section from $f = -\infty \rightarrow 0$.

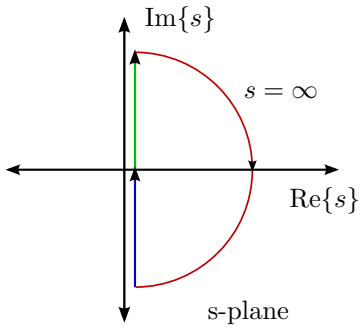


Figure 1.7: The Nyquist contour

Thus, to draw the complete Nyquist plot we need to add to the plot that you produced by transferring data from the Bode plot. The section from $-\infty$ to 0 is straightforward, as it is just the mirror image of the 0 to ∞ section that you have already drawn. Most transfer functions that you will meet have the degree of the denominator larger than the numerator (they are strictly proper). A consequence of this is that the gain is infinitesimally small at infinite frequency. Thus the gain for the circular part of the contour from $\omega = \infty$ to $\omega = -\infty$ is always zero. The whole sweep is therefore mapped to the origin of the Nyquist plot.

1.3.4 Examples

Consider the transfer function $G(s) = \dfrac{100}{s+10}$.

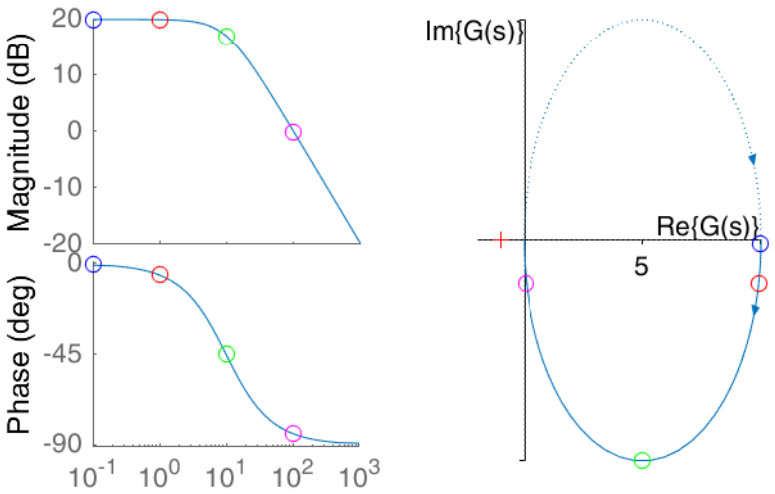


Figure 1.8: Example of converting the Bode plot for $G(s) = \dfrac{100}{s+10}$ to a Nyquist diagram.

Consider the transfer function $G(s) = \dfrac{100}{s^2+2s+26}$.

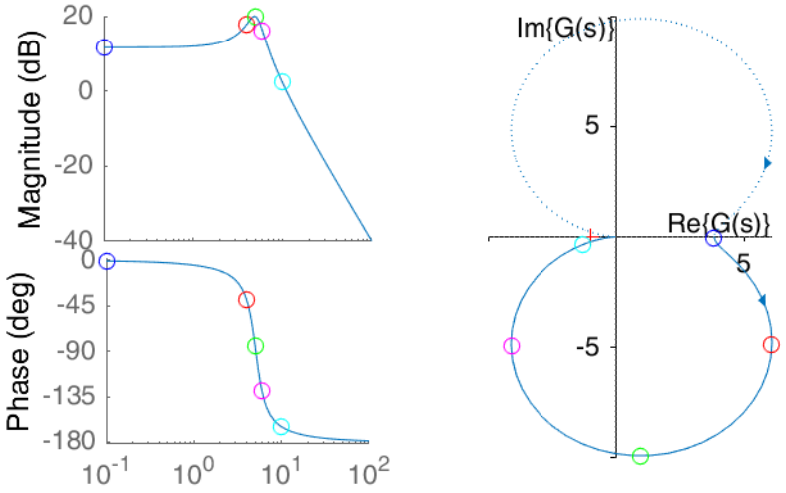


Figure 1.9: Example of converting the Bode plot for $G(s) = \dfrac{100}{s^2+2s+26}$ to a Nyquist diagram.

1.3.5 Construction of a Nyquist Plot

We can draw a Nyquist plot directly, without needing to draw a Bode plot first. We just consider the system response as we move around the required contour. Recall

$$|G(s)| = \frac{|K|\,|s+z_1|\,|s+z_2|\cdots|s+z_k|}{|s+p_1|\,|s+p_2|\cdots|s+p_n|}$$

$$\begin{aligned}\angle G(s) = \quad & \angle(s+z_1)+\angle(s+z_2)\cdots+\angle(s+z_k) \\ & -\angle(s+p_1)-\angle(s+p_2)\cdots-\angle(s+p_n)\end{aligned}$$

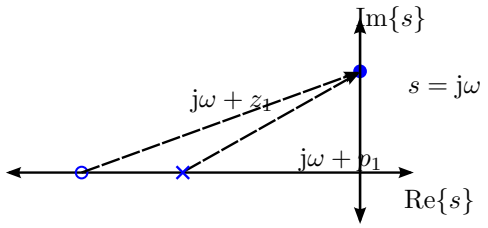


Figure 1.10: Diagram outlining the procedure for direct construction of a Nyquist plot. The point at $s = \mathrm{j}\omega$ is the test point that is carried through the Nyquist contour while the gain of the system is assessed by examination of the vectors from the test point to the system roots.

For example, consider the contribution of a single LHP pole. As we move from zero to infinite frequency the phase will move from zero to $-90°$. At the same time the gain will drop. We can therefore sketch the Nyquist plot that results. Example: $G(s) = 1/(s + 0.2)$.

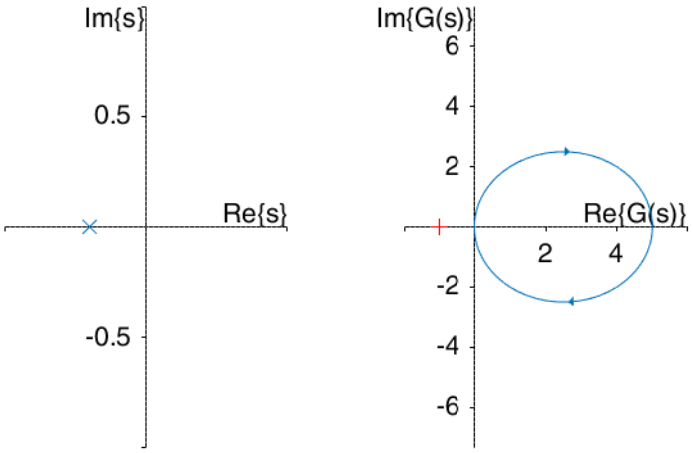


Figure 1.11: An example pole zero diagram of first order system is shown on the left, along with the corresponding Nyquist diagram on the right.

We can use the same procedure for drawing more complex examples. Simply start with a pole-zero map and consider what happens as we follow the Nyquist contour.

We need to be a little careful about plotting a Nyquist plot directly from a pole zero map, because the pole zero map doesn't tell us anything about the dc gain of the system. That is, the previous PZ map represents *any* transfer function of form $\frac{K}{(s+1+2\mathrm{j})(s+1-2\mathrm{j})} \equiv \frac{K}{s^2+2s+5}$. So, for example, if we knew that the transfer function was actually $\frac{10}{s^2+2s+5}$, which has a dc gain of 2, we would obtain the revised Nyquist plot shown in figure 1.13.

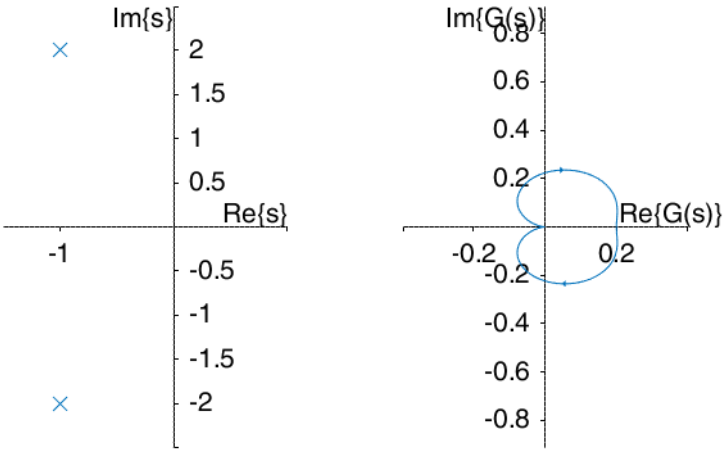


Figure 1.12: An example pole zero diagram of second order system is shown on the left, along with the corresponding Nyquist diagram on the right.

Following the contour we discussed before doesn't work if we have poles (or zeros) that would lie *on* the contour. Thus, if we have poles of zeros on the imaginary axis we modify the contour so that it takes an infinitesimally small "detour" into the positive real side of the s-plane to avoid the imaginary root. We then proceed as normal. The modified Nyquist contours can be seen in figure 1.14 for the cases of one or two poles located on the imaginary axis. Extension to higher numbers of poles should be self-evident, though this situation is rare in practice.

Note that there is no need to avoid zeros on the imaginary axis, as these simply cause the gain to go to zero. The phase moves by 180°as the contour moves through the zero, which results in the Nyquist diagram carrying on through the origin in a straight line. That is the Nyquist diagram approaches the origin from some direction and then leaves in the opposite direction.

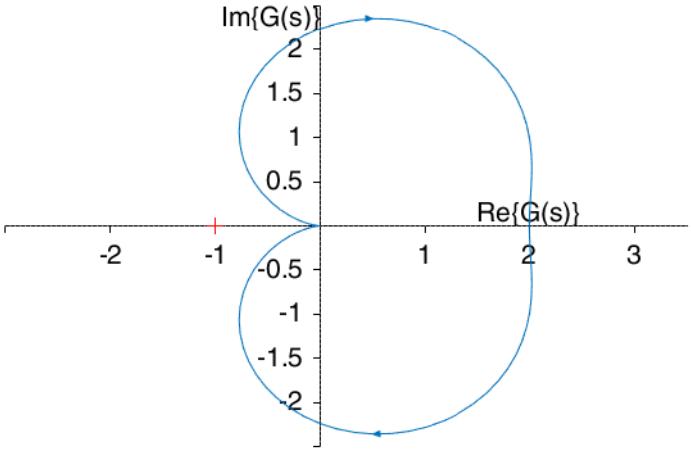


Figure 1.13: Nyquist plot of $G(s) = \frac{10}{s^2+2s+5}$.

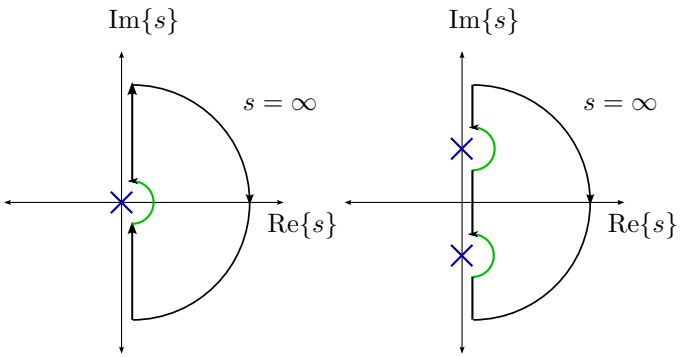


Figure 1.14: Modifications to the Nyquist contour made to avoid poles on the imaginary axis. In each case the semicircular detour (shown in green) has an infinitely small radius.

1.3.6 Examples

For example, consider $G(s) = \dfrac{4}{s(s+1)}$. Figure 1.15 shows an example of a system that has a pole on

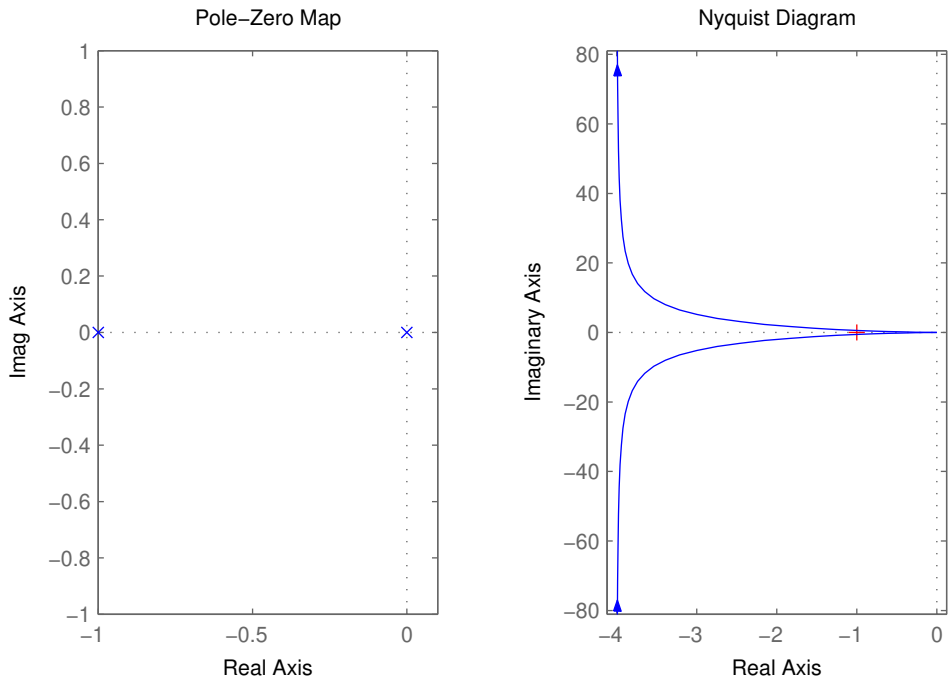


Figure 1.15: Pole-zero and Nyquist plots of the transfer function $G(s) = \dfrac{1}{s(s+1)}$.

the imaginary axis (at $s = 0$ in this case). The contour used is that shown on the left of figure 1.14. Notice that at the start of the contour, the test point is infintesmilly close to the pole at $s = 0$, but that there is an angle of $0°$ subtended by the vector. Thus, the Nyquist plot begins at an infinite radius and a phase of $0°$. This cannot be seen on the Nyquist diagram of course. As the test particle moves along the "detour" part of the Nyquist contour, the magnitude remains infinite, while the phase drops to $-90°$.

The test frequency then begins to move along the positive imaginary axis, which finally causes the gain to drop to a finite value, while the phase remains at $-90°$ for some time. This is what causes the Nyquist plot to appear in the seemingly strange position on the Nyquist diagram.

Figures 1.16 and 1.17 show some more examples of Nyquist diagrams for more complex transfer functions. Notice how it is permissible to draw the contours, even if they are located at infinity. In this sense the Nyquist diagram can be a cartoon that shows the overall behaviour of the system, without needing to be too quantitative. We will need to be a little careful with this when using the Nyquist diagram to assess stability in the next section.

Note that Matlab does *not* produce the "cartoon" Nyquist diagrams, so you sometimes need to take care in interpreting Matlab produced diagrams.

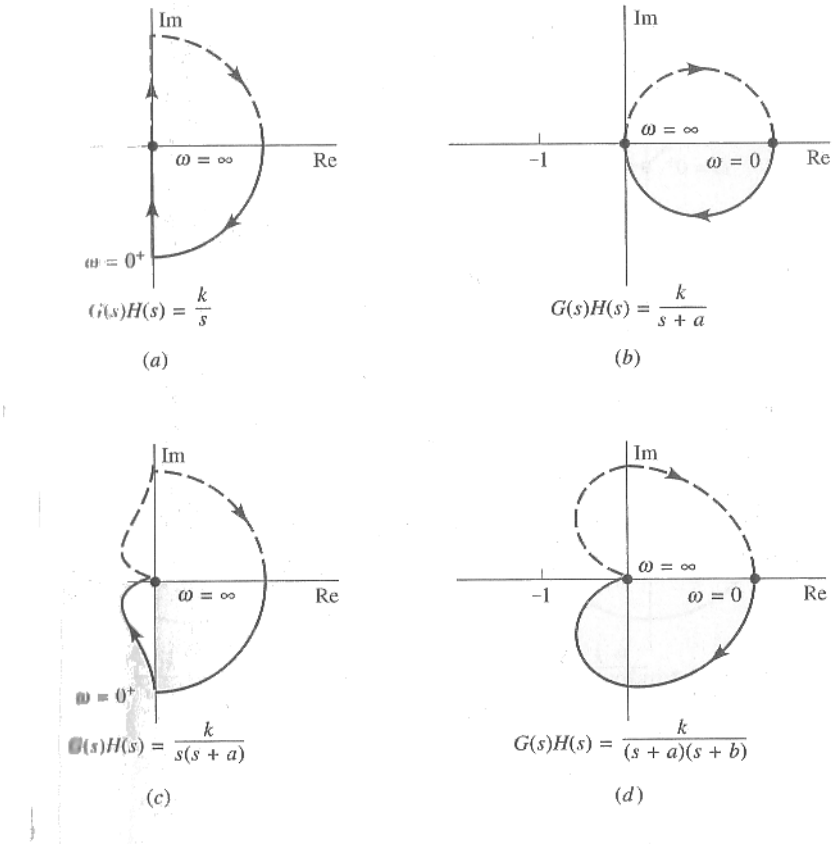


Figure 1.16: Example Nyquist diagrams for first and second order systems. This figure can be found on page 461 as table 6.9 in Stafani, Shahian, Savant and Hostetter "Design of Feedback Control Systems", 4th edition, Oxford University Press.

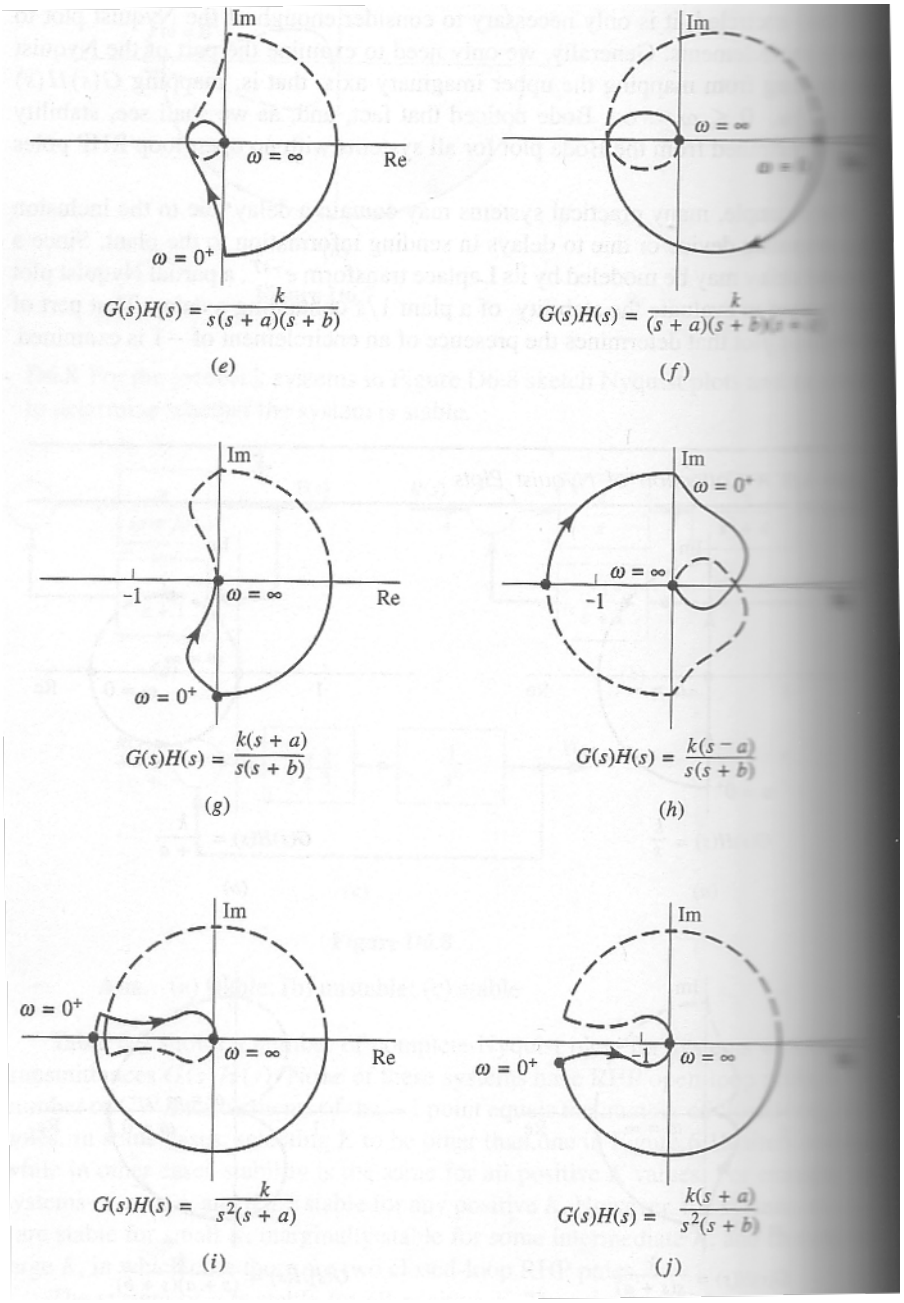


Figure 1.17: Examples of Nyquist diagrams for more complex systems. This figure can be found on page 462 as table 6.9 in Stafani, Shahian, Savant and Hostetter "Design of Feedback Control Systems", 4th edition, Oxford University Press.

1.3.7 Stability on the Nyquist plot

We can examine the Nyquist diagram to determine the gain and phase margins. The phase margin can simply be read as the difference between the phase $= -180°$ line and the point where the curve crosses the unit circle. The gain margin is the inverse of the distance to the point where the curve crosses the negative real axis. Note that if the curve crosses the negative x-axis, or the unit circle multiple

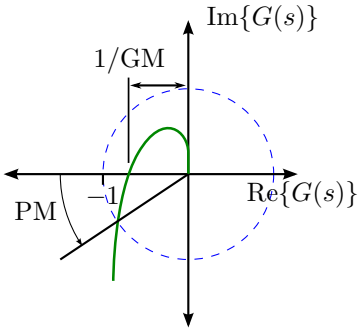


Figure 1.18: Graphical determination of phase and gain margins from the Nyquist diagram.

times then we choose the worst case for determining the gain and phase margin respectively.

Sometimes it is convenient to talk about a single number to characterise the stability. The so-called *stability margin* is a measure of how close the Nyquist plot comes to the critical point at $G(s) = -1$.

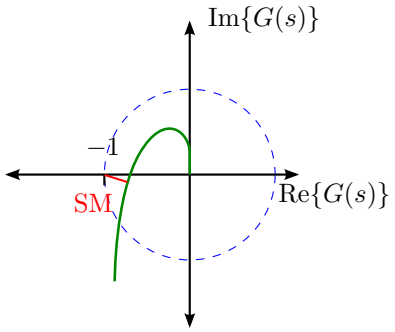


Figure 1.19: Graphical determination of the stability margin from the Nyquist diagram.

It is instructive to consider what happens to some rational function $f(s)$ as we traverse a closed clockwise contour in the s plane.

If we traverse a clockwise contour in the s-plane that encloses a *zero*, then the contour in the $f(s)$ plane encloses the origin in a *clockwise* encirclement. This can be seen in figure 1.22. If we traverse a clockwise contour in the s-plane that encloses a *pole*, then the contour in the $f(s)$ plane encloses the origin in an *anticlockwise* encirclement. This can be seen in figure 1.21. However, if we don't enclose a root, or the number of poles and zeros is matched, then we *do not* encircle the origin in the $f(s)$ plane. This can be seen in figures 1.20 and 1.23 respectively. This behaviour generalises:

Theorem 1 (Mapping Theorem)**.** *If a clockwise contour in the s plane encloses Z zeros and P poles, then the corresponding contour in the $f(s)$ plane will make Z-P clockwise encirclements of the origin.*

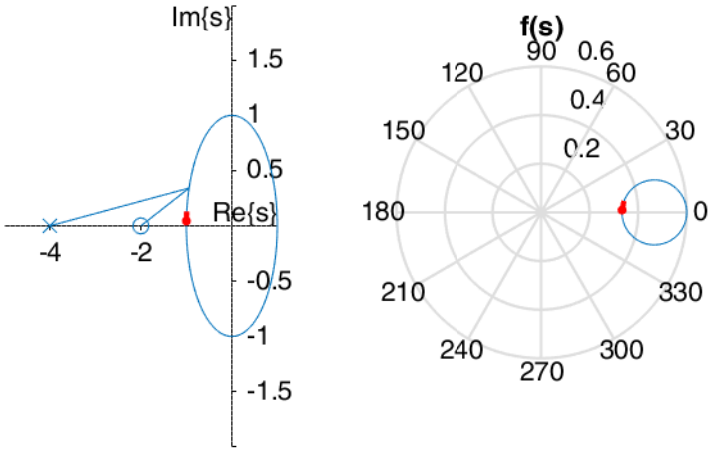


Figure 1.20: Traversing a clockwise contour in the s plane that does not enclose a root of $f(s)$ leads to no encirclement of the origin in the $f(s)$ plane.

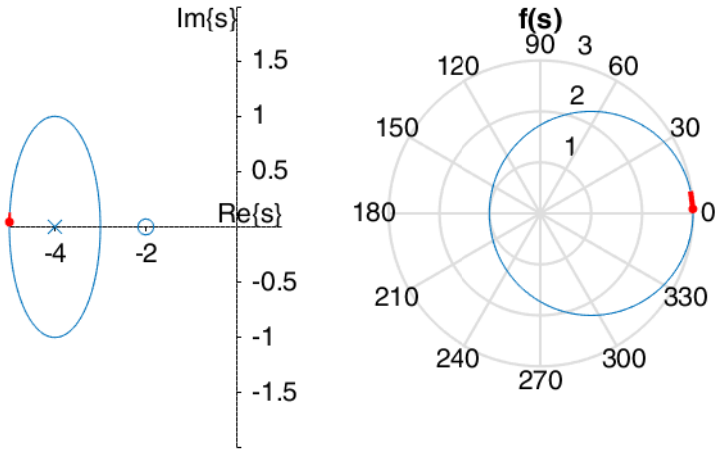


Figure 1.21: Traversing a clockwise contour in the s plane that encloses a pole of $f(s)$ leads to an anticlockwise encirclement of the origin in the $f(s)$ plane.

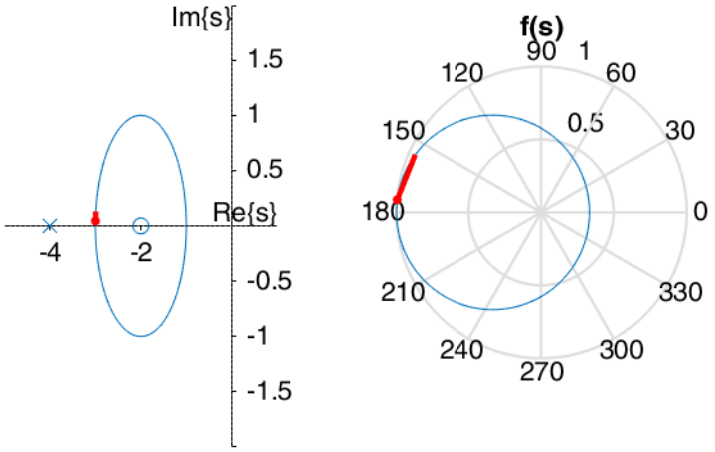


Figure 1.22: Traversing a clockwise contour in the s plane that encloses a zero of $f(s)$ leads to a clockwise encirclement of the origin in the $f(s)$ plane.

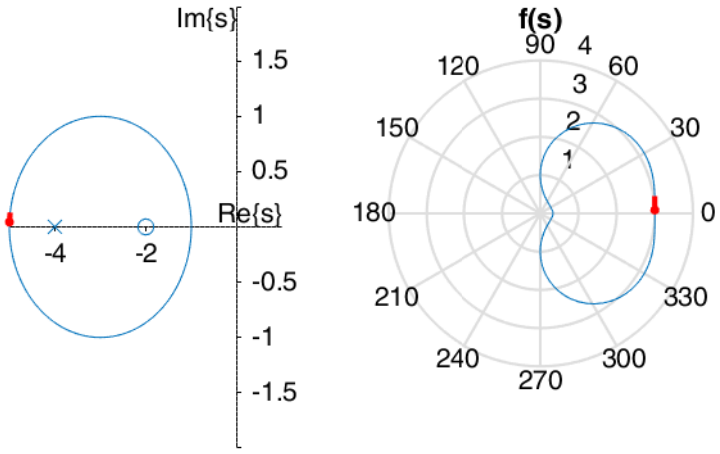


Figure 1.23: Traversing a clockwise contour in the s plane that encloses a both a pole and a zero of $f(s)$ leads to no encirclement of the origin in the $f(s)$ plane.

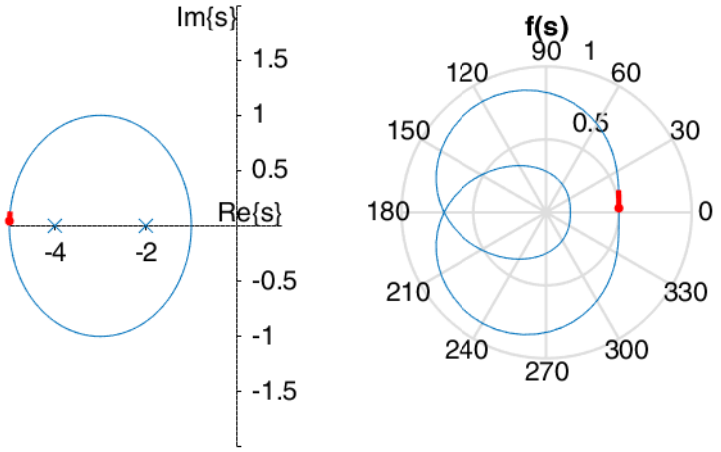


Figure 1.24: Traversing a clockwise contour in the s plane that encloses two poles of $f(s)$ leads to two anticlockwise encirclements of the origin in the $f(s)$ plane.

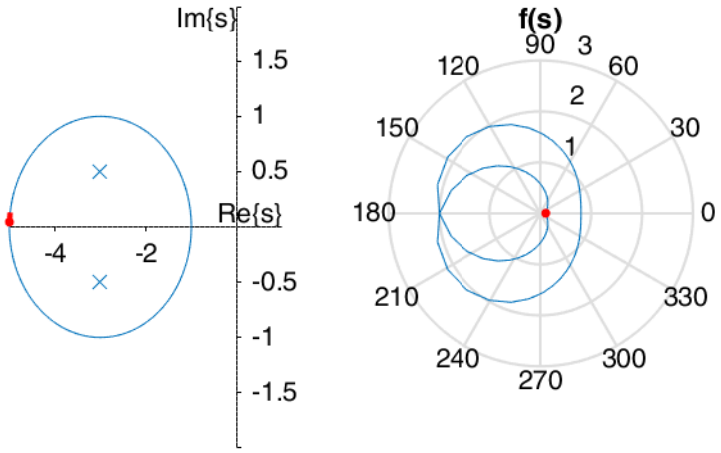


Figure 1.25: Traversing a clockwise contour in the s plane that encloses a pair of complex poles of $f(s)$ leads to two anticlockwise encirclements of the origin in the $f(s)$ plane.

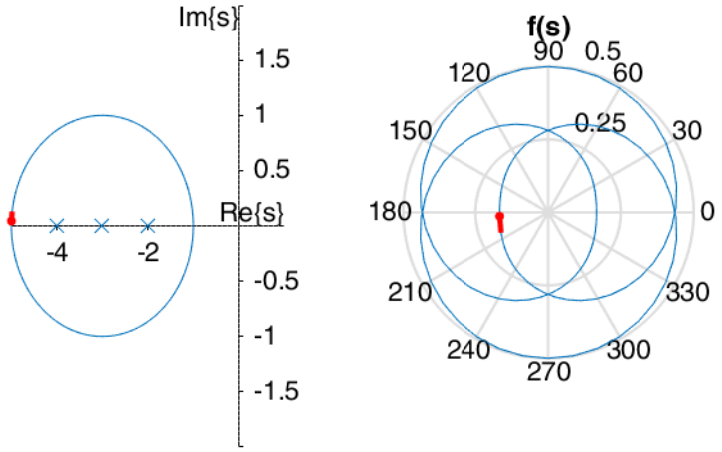


Figure 1.26: Traversing a clockwise contour in the s plane that encloses three poles of $f(s)$ leads to three anticlockwise encirclements of the origin in the $f(s)$ plane.

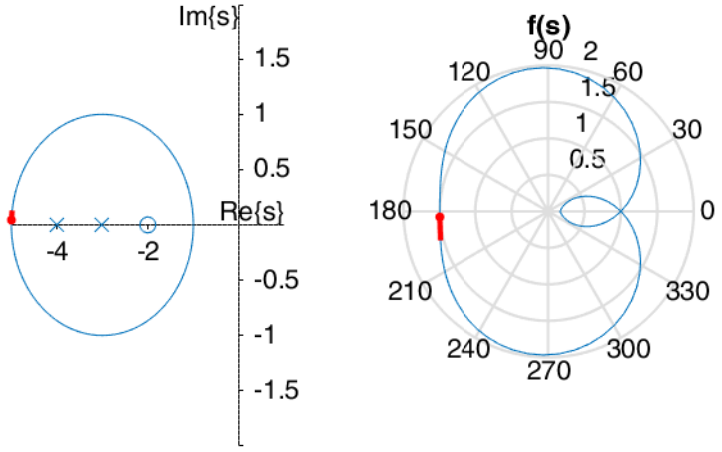


Figure 1.27: Traversing a clockwise contour in the s plane that encloses a pair of poles and a zero of $f(s)$ leads to a single anticlockwise encirclement of the origin in the $f(s)$ plane.

To determine the stability of a system we want to know whether the poles of the closed loop transfer function, $T = \frac{G(s)}{1+G(s)}$, lie in the right half of the s-plane. That is, we care whether $1+G(s) = 0$ anywhere in the right half plane. Consider $G(s) := \dfrac{n_{\text{ol}}(s)}{d_{\text{ol}}(s)} \rightsquigarrow 1 + G(s) = \dfrac{n_{\text{ol}}(s) + d_{\text{ol}}(s)}{d_{\text{ol}}(s)}$, $T = \dfrac{n_{\text{ol}}}{n_{\text{ol}} + d_{\text{ol}}}$. Notice that the poles of $1 + G(s)$ are the poles of the open loop transfer function. The zeros of $1 + G(s)$ are the poles of the closed loop transfer function. We will now look at using the mapping theorem on $1 + G(s)$. We know that the number of clockwise encirclements of the origin will be equal to the difference between the number of zeros and the number of poles.

This is the same as the difference in the number of poles of the closed loop transfer function, $T(s)$ and the number of poles of the open loop transfer function, $G(s)$.

We will examine what happens if the s-plane contour in the mapping theorem enclose the entire right half of the s-plane (the Nyquist contour) and consider the encirclements of the origin made by $1 + G(s)$. Equivalently, we will consider the encirclements of -1 made by $G(s)$. Applying the mapping theorem to this particular case, shows that the number of poles in the right half plane of the *closed loop* transfer function can be determined by examining the Nyquist plot of the open loop transfer function.

Theorem 2 (Nyquist Stability Theorem)**.** *The number of closed loop poles in right half of s-plane is equal to the number of open loop poles in right half of s-plane plus the number of clockwise encirclements of* -1*.*

Remember that for the system to be stable we must have *no* closed loop poles in the right half plane.

Notice that we can find the number of poles that the closed loop system has in the right half of the s plane, without ever finding the closed loop transfer function.

To count the clockwise encirclements:

1. Draw a straight line from $-1 + 0\text{j}$ to ∞ in *any* direction. Imagine sitting at -1, looking along your straight line.

2. Count how many times the Nyquist plot crosses from left to right over your chosen line. For each such crossing add one to the count of the number of encirclements.

3. Count how many times the Nyquist plot crosses from right to left over your chosen line. For each such crossing subtract one from the count of total encirclements.

If you have drawn a correctly constructed Nyquist plot then the final number that you get will be the same regardless of which orientation you chose for your original line. (So choose a direction that makes counting easy!)

1.3.8 Examples

Consider the system $G(s) = \dfrac{100}{(s+1)(s+2)(s+5)}$ This system will be closed loop stable, but if we were to increase the gain then it would eventually encircle $s = -1$ and become unstable.

Consider the conditionally stable system $G(s) = \dfrac{s^2 + 2s + 4}{s(s+4)(s+6)\left(s^2 + 1.4s + 1\right)}$, which has a Bode plot shown in figure 1.29 and a corresponding Nyquist diagram in figure 1.30.

For some gains this will be closed loop stable, for others unstable. You cannot easily determine this from a Bode plot.

Let's consdier the transfer function $G(s) = \dfrac{15000}{(s-10)(s+30)(s+100)}$, which includes a right half plane open loop pole at $s = 10$.

1.3.9 Compensators Effects on the Nyquist Diagram

A proportional compensator simply scales the Nyquist contour. Consider the following example of $G(s) = \dfrac{20}{(s+1)(s+2)(s+3)}$. Notice that too much proportional gain makes this system closed loop unstable as shown in figure 1.31.

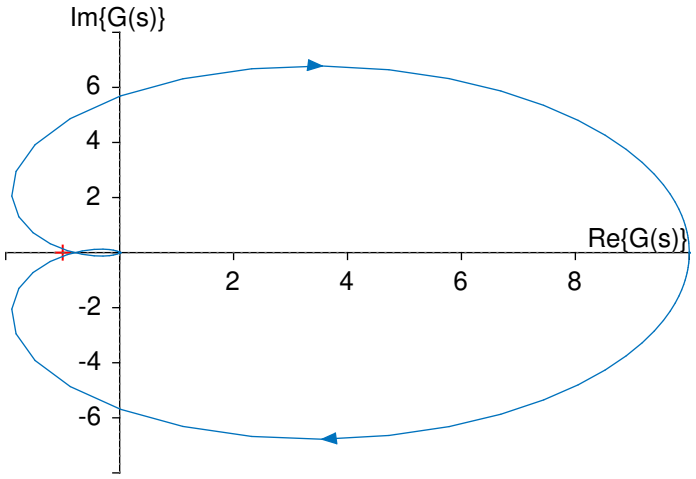


Figure 1.28: The Nyquist diagram of a third order system.

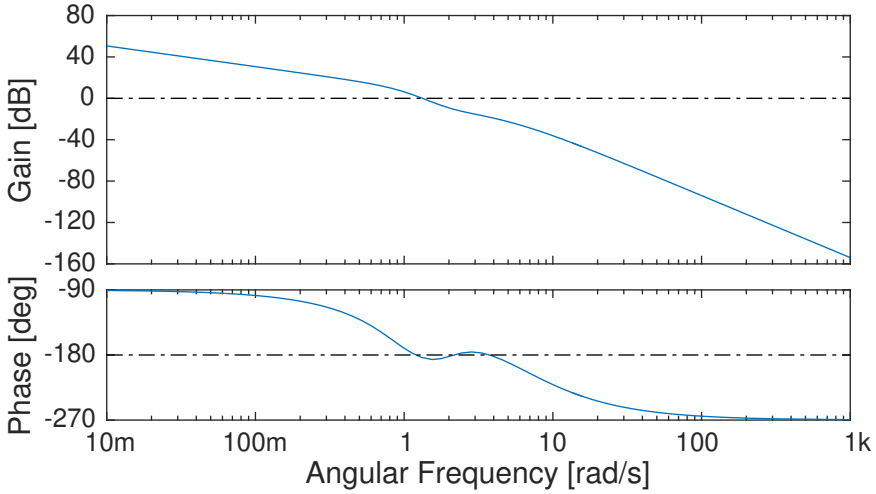


Figure 1.29: The Bode plot of a conditionally stable system.

Even if the system does not become unstable, the phase margin will typically still be reduced by increasing the gain. The example in figure 1.32 shows a second order system, for which the phase margin is reduced by a proportional compensator, $C(s) = 4$.

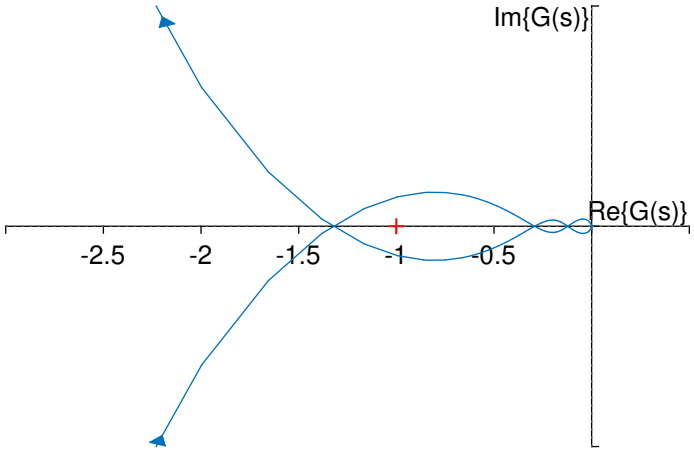


Figure 1.30: The portion of the Nyquist diagram of a third a conditionally stable system in the vicinity of $G(s) = -1$.

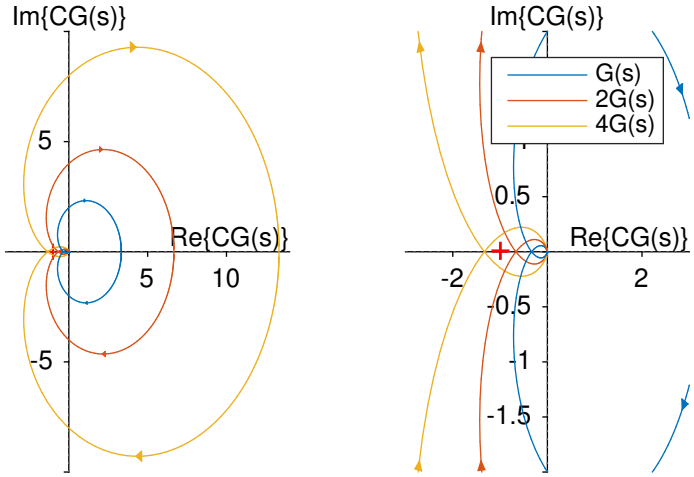


Figure 1.31: Nyquist diagrams for a plant and two proportionally compensated plants. The version on the right shows the portion of the diagrams in the vicinity of $G(s) = -1$ and exposes how varying the gain can affect the stability of the closed system.

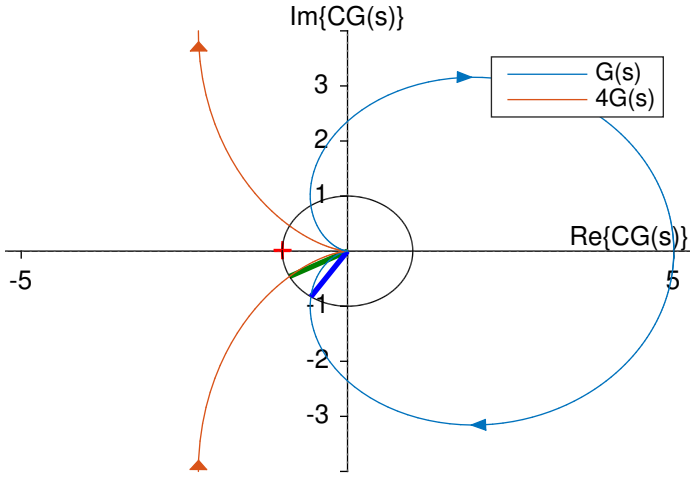


Figure 1.32: A second order system, showing reduced phase margin as the gain proportional compensator is increased.

A lag compensated system starts with a higher gain than the uncompensated system, but approaches the Nyquist contour of the uncompensated system before it nears $-1+\mathrm{j}0$. Consider our example again with $C(s) = \dfrac{s+0.4}{s+0.1}$. $(\omega_\mathrm{b} = 0.1, \alpha = 4)$

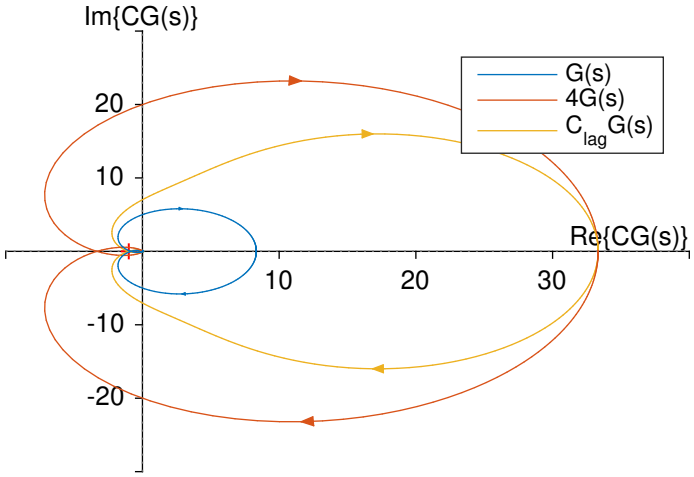


Figure 1.33: Comparison of the Nyquist diagrams for a system, a proportionally compensated system and a Lag compensated system chosen to have the same dc gain and the proportionally compensated system.

If ω_b is reduced then the compensated system approaches the uncompensated more "quickly".

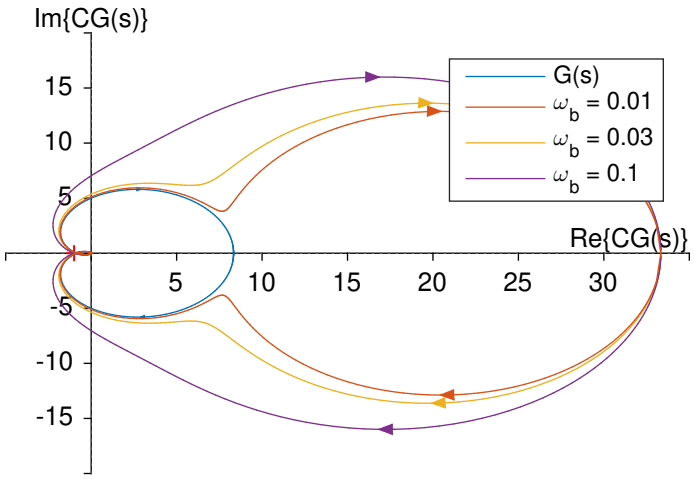


Figure 1.34: Nquist diagrams for a lag compenated system as the breakpoint of the compensator is adjusted.

A PI compensator changes the shape of the Nyquist plot at low frequencies. Again ω_b determines how quickly the system approaches the uncompensated system.

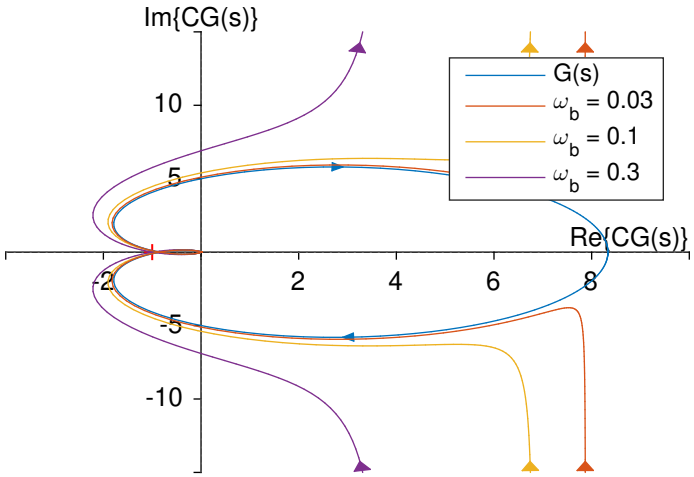


Figure 1.35: Nyquist diagram showing the effect of adding a PI compensator to a system. The plot also shows the changes in the plot produced by varying the breakpoint ω_b.

The lead compensator "rotates" the Nyquist contour *away* from -1, though the picture is confused by the extra gain at high frequency.

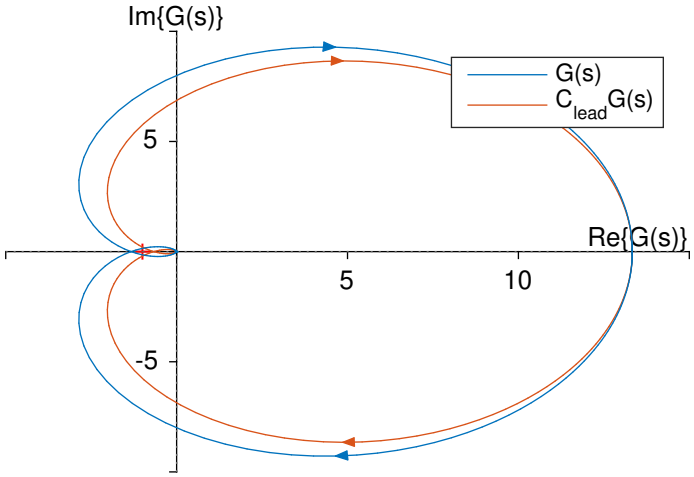


Figure 1.36: Nyquist plot of a lead compensated second order system.

2 Classical Control in Discrete Time

Classical control was first developed in an era when there were few computers do do sampled data processing. However, most control is now performed using computers, so a technique to extend control to the discrete domain is useful. Note though, that a continuous time approximation is often fine (if the sampling is fast enough), and we don't need to worry too much about the extra complexity introduced by sampling.

Sometimes we can just do our design assuming a continuous time system, and then convert the resulting controller to discrete time for implementation. However, sometimes we might choose to do the design directly in the discrete time domain. Most of the classical control techniques that we use for continuous time systems can also be carried over for discrete time systems. The big difference in the mathematical treatment of such systems is the use of the z-transform rather than the Laplace transform. We will also need to consider the choice of the sampling interval, which is not required in continuous time systems. Recall for example that a sampled data system has an upper frequency limit given by the Nyquist frequency, beyond which aliasing prevents the representation of signals.

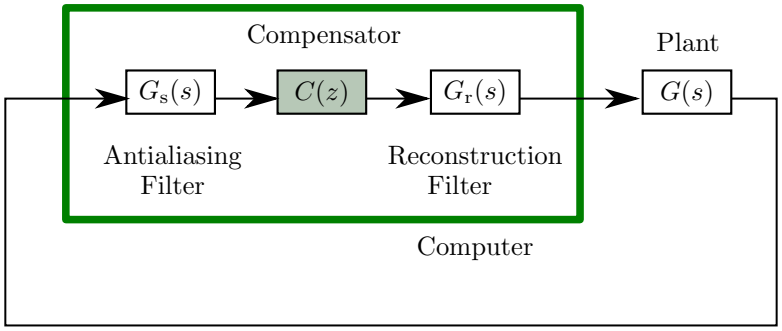


Figure 2.1: A schematic representation of a discrete time controller.

In a sampled data system we implement the controller in discrete time (typically digitally in a computer). If a plant is continuous time then it of course remains continuous time (though we frequently model it as discrete time so that we can analyse the whole loop using the same tools). We sometimes also include anti-aliasing and reconstructions filters, which remain in continuous time.

2.1 The z transform

Recall that we made extensive use of the Laplace transform in continuous time control system work because it allowed us to handle derivatives.

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = 2x(t) \overset{\mathcal{L}}{\Longleftrightarrow} sY(s) = 2X(s)$$
$$\implies G(s) := \frac{Y(s)}{X(s)} = \frac{2}{s}$$

In discrete time systems we replace differential equations with *difference* equations, where we look at how things change between two moments that are separated by one unit of time.

$$y(t+1) - y(t) = 2x(t) \qquad t \in \mathbb{Z}_+$$

We introduce the z-transform to deal with the time delay.

$$y(t+1) - y(t) = 2x(t) \overset{\mathcal{Z}}{\Longleftrightarrow} zY(z) - Y(z) = 2X(z)$$
$$G(z) := \frac{Y(z)}{X(z)} = \frac{2}{z-1}$$

The z-transform provides a general tool for working with difference equations, where the next value of a variable is described as a function of its current and previous values, as well as the current and previous

values of an input variable or variables. The example above exposes the slight mismatch between the s and z-domains, in that z domain transfer function has the extra "1" appearing in the denominator. This arises from the fact that differential equations describe only changes and use an initial condition to work out what the output is at any moment. By contrast difference equations keep track of the actual values internally. We will see that this difference has a wide range of minor consequences throughout the course. However, the remarkable thing is that there is a great deal of commonality and we will normally be able to develop continuous and discrete time formulations simultaneously.

Relationship between the Laplace and z transforms

If we were to take the Laplace transform of a discrete time signal, then we would obtain terms including $\mathrm{e}^{-skt_\mathrm{s}}$ because the various delays by amounts kt_s. If we simply rewrite $\mathrm{e}^{st_\mathrm{s}}$ as z then what we obtain is precisely the z transform.

2.2 The z plane

Just as plotting pole and zero locations on the s-plane was useful for understanding the behaviour of continuous time systems, so plotting them on the z-plane is useful for discrete time system. A pole zero plot is produced identically, but the locations for the roots require a little more thought, and the interpretation of the resulting map is also different. If you wish, you can also plot root locus diagrams in the z-plane, using the same rules as for the s-plane. Again the interpretation of the resulting curves is somewhat different.

The relationship between pole locations on the s and z planes is given by $z = \mathrm{e}^{st_\mathrm{s}}$, where t_s is the sample time. Consider a pair of poles at an arbitrary location $\lambda_s = \sigma \pm \mathrm{j}\omega_\mathrm{n}$. In the z-plane these poles would be at

$$\begin{aligned}\lambda_\mathrm{z} &= \mathrm{e}^{\lambda_\mathrm{s} t_\mathrm{s}} \\ &= \mathrm{e}^{(\sigma \pm \mathrm{j}\omega_\mathrm{n})t_\mathrm{s}} \\ &= \mathrm{e}^{\sigma t_\mathrm{s}} \mathrm{e}^{\pm \mathrm{j}\omega_\mathrm{n} t_\mathrm{s}} \\ &= \mathrm{e}^{\sigma t_\mathrm{s}} \angle \pm \omega_\mathrm{n} t_\mathrm{s}\end{aligned}$$

This is shown schematically in figure 2.2. We know that CT systems are stable iff all of their poles

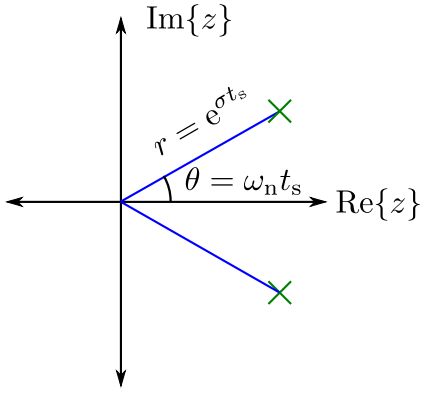


Figure 2.2: Pole locations on the z-plane.

are in the left half of the s-plane ($\mathrm{Re}\{\sigma\} < 0$). On the z-plane this corresponds to pole locations with $|\lambda_z| < 1$. Conversely, unstable poles have $\mathrm{Re}\{\lambda_s\} > 0$ and $|\lambda_z| > 1$ respectively. The correspondence between the regions of stability (and otherwise) in the s and z planes are shown in figure 2.3.

Examples for the step responses of both stable and unstable first order discrete time systems can be seen in figures 2.4 and 2.5. The transfer functions of the systems shown here are $G(z) = \dfrac{1-a}{z-a}$, where the particular value of $a \in \mathbb{R}$ for each curve is shown in the legend of the figures. The poles lie within the unit circle for 2.4, resulting in step responses that settle exponentially to one. In contrast, in figure 2.5 the poles lie outside the unit circle, so the resulting modes grow exponentially.

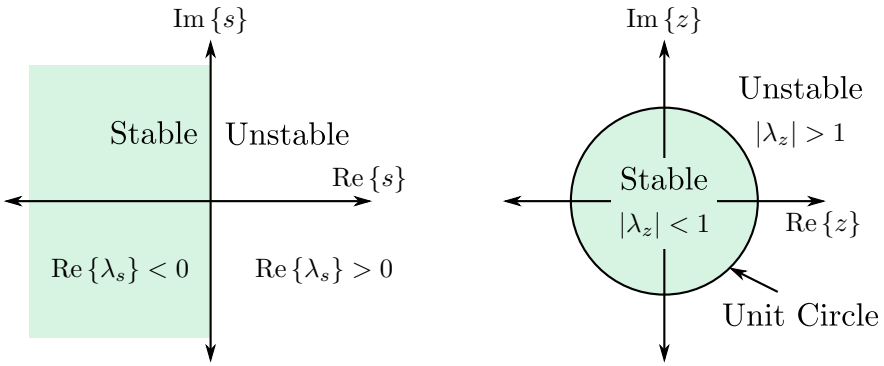


Figure 2.3: Regions of stability on the s and z planes.

A pole at $z = a$ in the z-plane is associated with a time domain mode a^t.

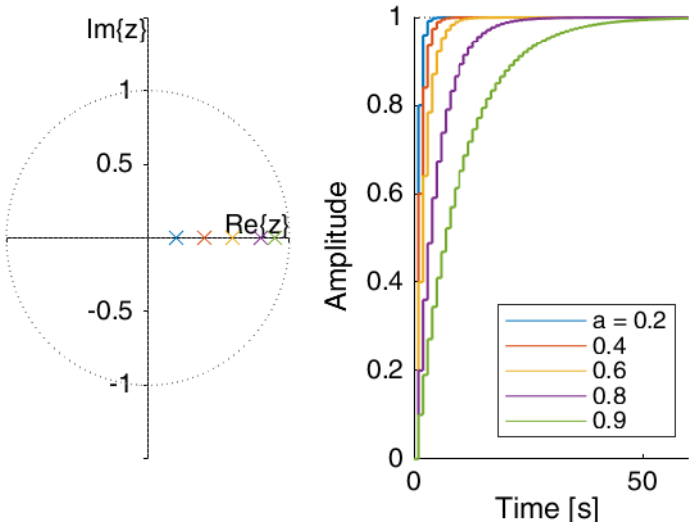


Figure 2.4: Stable modes corresponding to real positive pole locations in the z-plane.

When compared to the response of the system having its pole symmetrically on the right side of the z plane, notice that the oscillates between positive and negative values on successive samples. This can be seen in figure 2.6 and is a general property of poles in the left side of the z-plane.

We will not often encounter signals like this in control engineering, but you should recognise the behaviour because it will help you find errors in your work.

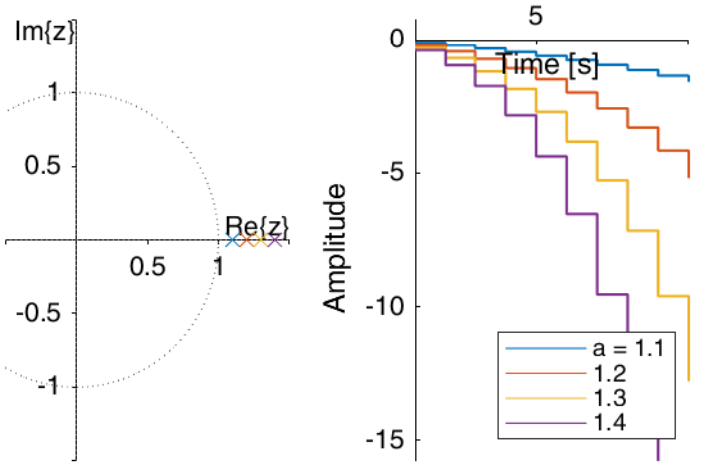


Figure 2.5: Unstable modes corresponding to real positive pole locations in the z-plane.

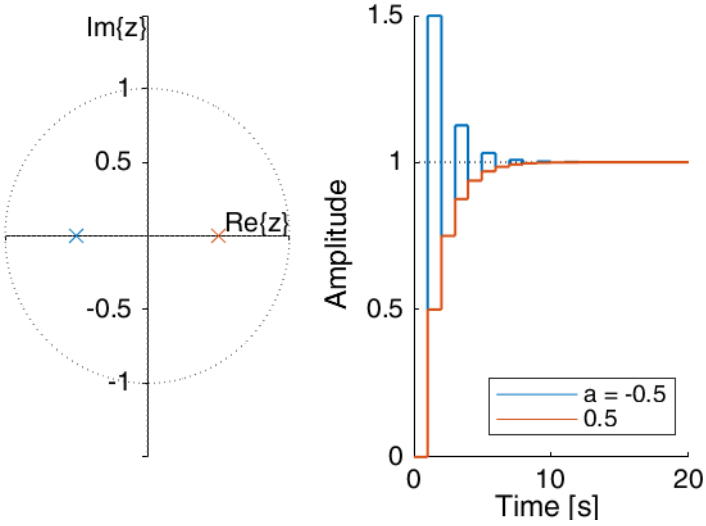


Figure 2.6: Stable modes corresponding to real negative pole locations in the z-plane.

We can also consider the effect of moving complex pairs of poles on the z-plane. Increasing the natural frequency of a complex pole pair increases their angle from the real axis of the z-plane , as shown in figure 2.7.

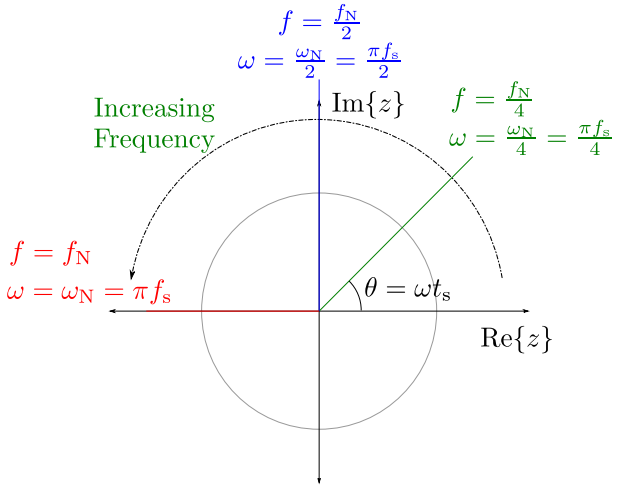


Figure 2.7: Frequency on the z-plane.

Notice that zero frequency (dc) in the z-plane occurs at $z = 1$. That is, $s = 0$ maps to $z = e^{0t_s} = 1$. This will be useful, as to calculate the dc gain of a discrete time system, we will substitute $z = 1$ into its transfer function.

Figure 2.8 shows the effect on system modes of moving the poles around the unit circle. In this case the poles have been placed slightly inside the unit circle, so that the modes are stable and hence decay to some steady state level. The transfer function used in these examples is $G(z) = \dfrac{1}{(z-a)(z-a^{\mathrm{T}})}$ with $a \in \mathbb{C}$ as given in the figure legend. In all cases $|a| = 0.9$ to ensure stability.

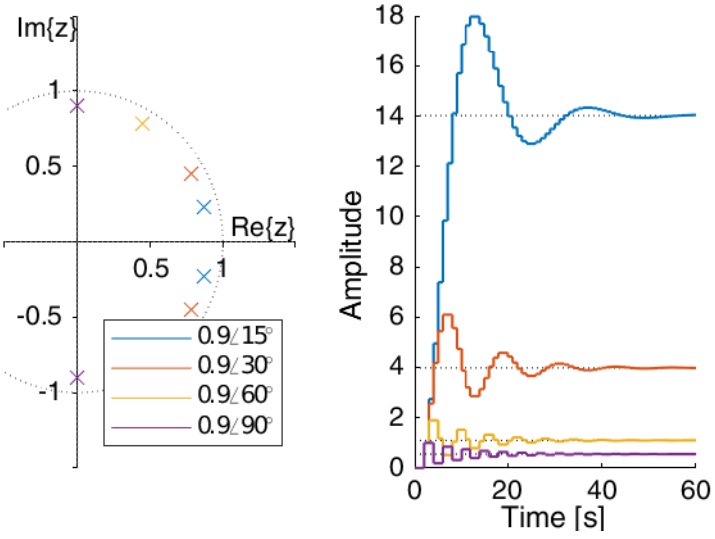


Figure 2.8: Stable modes corresponding to complex pole locations in the z-plane.

In a closed-loop discrete-time control system we need to be careful about the delay introduced by the sampling. Sampling too slowly leads to degradation in the phase margin of a system and can lead to instability. As a rule of thumb, most closed loop systems are designed so that the Nyquist frequency is at least ten times faster than the desired unity gain bandwidth of the control system. It is not uncommon

for the sampling rate to be chosen significantly higher than that, particularly for hard to control systems, such as non-minimum phase or open loop unstable systems.

As a result, in control we generally find most (dominant) poles live in a narrow wedge around the positive real axis of the z-plane as shown in figure 2.9.

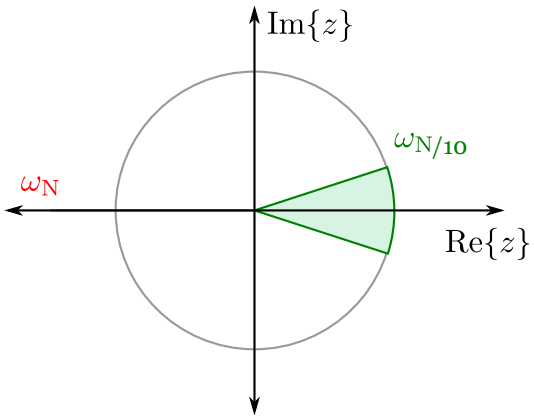


Figure 2.9: Region of the z-plane likely to contain most poles in a closed loop control system.

2.3 Aliasing

As should be evident from figure 2.7, there is a limit to the highest frequency that can be handled by the z-plane representation. The highest frequency that can be represented corresponds to a pole pair that has moved to an angle of π in the z-plane.

$$\begin{aligned} \pi &= \omega_{\max} t_s \\ \omega_{\max} &= \frac{\pi}{t_s} \\ \implies 2\pi f_{\max} &= \pi f_s \\ f_{\max} &= \frac{f_s}{2} \end{aligned}$$

So, the highest frequency that can be unambiguously represented on the z-plane is the Nyquist frequency.

If we try to move a pole pair to higher frequency then we should expect aliasing. As an example, consider a mode that has natural frequency of $\frac{5}{4} f_N$. Let's see where the corresponding poles would lie in the z-plane.

$$\begin{aligned} \lambda_s &= 0 \pm j\frac{5}{4} f_N \\ \rightsquigarrow \lambda_z &= e^{\pm j(\frac{5}{4} f_N) t_s} \\ &= e^{\pm j(\frac{5}{4} \frac{1}{2t_s}) t_s} \\ &= e^{\pm j\frac{5}{8}} \end{aligned}$$

This is indistinguishable from a pair of poles lying at $e^{\pm j\frac{3}{8}}$. Frequencies higher than the Nyquist frequency "wrap around" the complex plane and look like lower frequencies, which is the familiar aliasing behaviour of sampled systems. This is illustrated in figure 2.10 which illustrates that the mapping from the s to the z-plane is multivalued. That is, there are many frequencies in the s-plane that map to each point on the z-plane.

The possibility of aliasing may lead us to instinctively add antialiasing filters to a discrete time control system. In general this is a bad idea, because the filter will introduce phase delay (particularly if you use an aggressive higher order filter). While it is true that aliasing can compromise the performance of a control system, sometimes it is less troublesome that the stability reduction produced by a filter. Each case needs to be considered on its own merits.

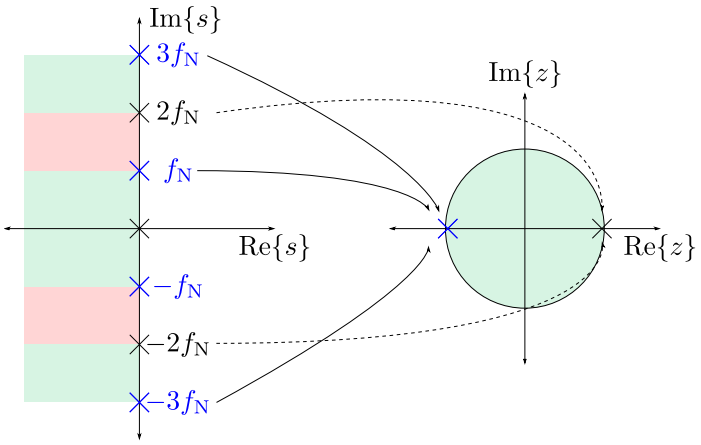


Figure 2.10: Aliasing of the mapping from the s to the z-plane.

2.4 Characteristic lines in the z-plane

2.4.1 Damping in the z-plane

Finally, we can consider where poles having the same damping ratio lie in the z-plane. A transfer function $G(s) = \dfrac{\omega_\mathrm{n}^2}{s^2 + 2\zeta\omega_\mathrm{n}s + \omega_\mathrm{n}^2}$ has poles at $\lambda_\mathrm{s} = -\zeta\omega_\mathrm{n} \pm \mathrm{j}\omega_\mathrm{n}\sqrt{1-\zeta^2}$, or equivalently at $\lambda_\mathrm{s} = -\zeta\omega_\mathrm{n} \pm \mathrm{j}\omega_\mathrm{d}$ We can again find the corresponding locations of the poles in the z-plane.

$$\begin{aligned}\lambda_\mathrm{z} &= \mathrm{e}^{\lambda_\mathrm{s}t_\mathrm{s}} \\ &= \mathrm{e}^{-\zeta\omega_\mathrm{n}t_\mathrm{s} \pm \mathrm{j}\omega_\mathrm{n}\sqrt{1-\zeta^2}t_\mathrm{s}} \\ &= \mathrm{e}^{-\zeta\omega_\mathrm{n}t_\mathrm{s}}\mathrm{e}^{\pm\mathrm{j}\omega_\mathrm{n}\sqrt{1-\zeta^2}t_\mathrm{s}}\end{aligned}$$

Notice that the second term has magnitude of one, so only serves to rotate the angle of the poles in the z-plane.

$$\begin{aligned}\lambda_\mathrm{z} &= \mathrm{e}^{-\zeta\omega_\mathrm{n}t_\mathrm{s}}\angle{\pm\omega_\mathrm{n}\sqrt{1-\zeta^2}t_\mathrm{s}} \\ &= \mathrm{e}^{-\zeta\omega_\mathrm{n}t_\mathrm{s}}\angle{\pm\omega_\mathrm{d}t_\mathrm{s}}\end{aligned}$$

We would like to know the shape of the line in the z-plane where we can find poles corresponding to a given value of damping ratio. This turns out to be easier if we switch to polar coordinates, and write the pole location as $r\angle\theta$ then we have

$$\begin{aligned}r = \mathrm{e}^{-\zeta\omega_\mathrm{n}t_\mathrm{s}} \implies \ln r &= -\zeta\omega_\mathrm{n}t_\mathrm{s} \\ \text{and } \theta &= \omega_\mathrm{n}\sqrt{1-\zeta^2}t_\mathrm{s}\end{aligned}$$

If we wanted we could choose a value of ζ along with a set of values for $\omega_\mathrm{n}t_\mathrm{s}$ and extract the corresponding values for r and θ. This would let us plot the locus of points having a particular ζ.

Alternately we can rearrange the two equations above to find expressions for $\omega_\mathrm{n}t_\mathrm{s}$.

$$\text{That is, } \omega_\mathrm{n}t_\mathrm{s} = \frac{-\ln r}{\zeta} \text{ and } \omega_\mathrm{n}t_\mathrm{s} = \frac{\theta}{\sqrt{1-\zeta^2}}$$

Equating the two, we find a direct expression for the constant damping locus

$$\begin{aligned}
\frac{-\ln r}{\zeta} &= \frac{\theta}{\sqrt{1-\zeta^2}} \\
-\ln r\sqrt{1-\zeta^2} &= \theta\zeta \\
\ln^2 r\left(1-\zeta^2\right) &= \theta^2\zeta^2 \\
\ln^2 r &= \left(\ln^2 r + \theta\right)\zeta^2 \\
\implies \zeta &= \frac{\ln r}{\sqrt{\ln^2 r + \theta^2}}
\end{aligned}$$

Figure 2.11 shows the curves traced by lines of constant damping ratio. That is, any two pairs of poles lying on one of these dashed lines will have the same damping (but different frequency).

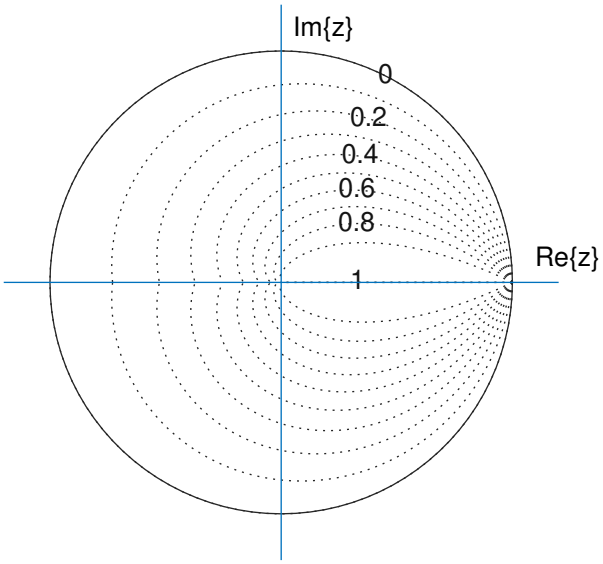


Figure 2.11: Lines of constant damping ratio.

Figure 2.12 shows the an example of such a region. Notice that these regions are generally more complex than in the case of the s-plane.

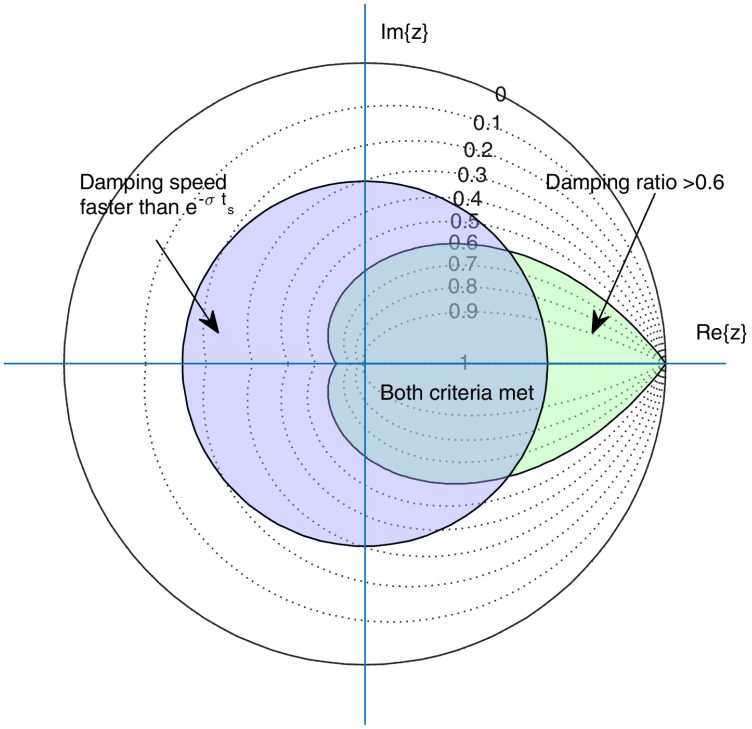


Figure 2.12: A hypothetical example of an acceptable region for the locations of closed loop poles.

2.4.2 Special locations in the z plane

As was the case in continuous time, the presence of damping perturbs the frequency of a mode so that the damped frequency is lower than the natural frequency. Figure 2.13 shows the resulting lines of constant damped frequency.

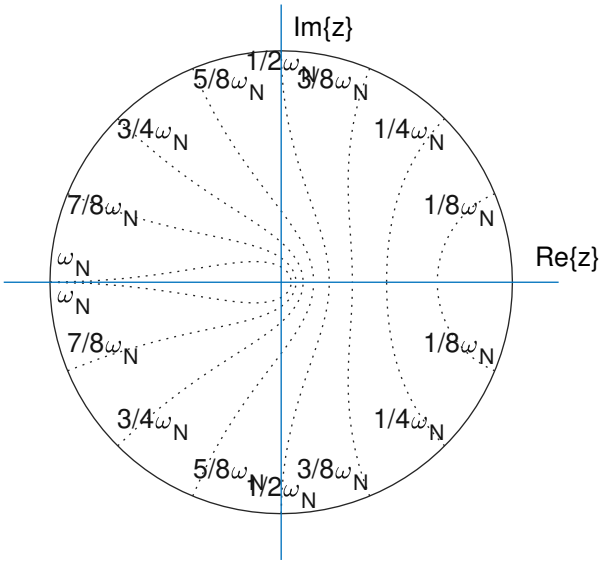


Figure 2.13: Lines of constant natural frequency are shown dotted on the diagram.

The mapping from the s-plane to the z-plane is conformal, so the lines of constant damping ratio and constant damped frequency remain perpendicular everywhere. Figure 2.14 shows the relationship between the two sets of curves.

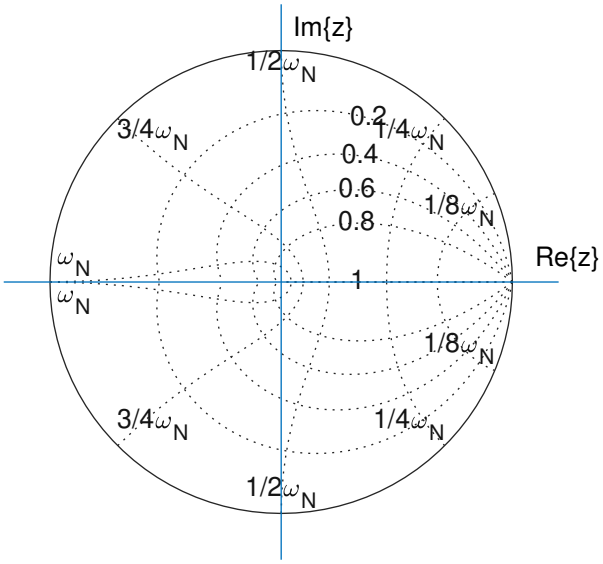


Figure 2.14: Lines of constant damping ratio and constant natural frequency are shown dotted on the diagram.

All of these features are difficult to produce manually, so make use of Matlab's `zgrid` function to draw them during your design work. Having access to a high resolution diagram of the z-plane will allow you to find appropriate locations in the z-plane. An example is shown in figure 2.15.

The z-location corresponding to $s = 0$ is $z = 1$. That is, a mode that is a constant (dc) has a pole

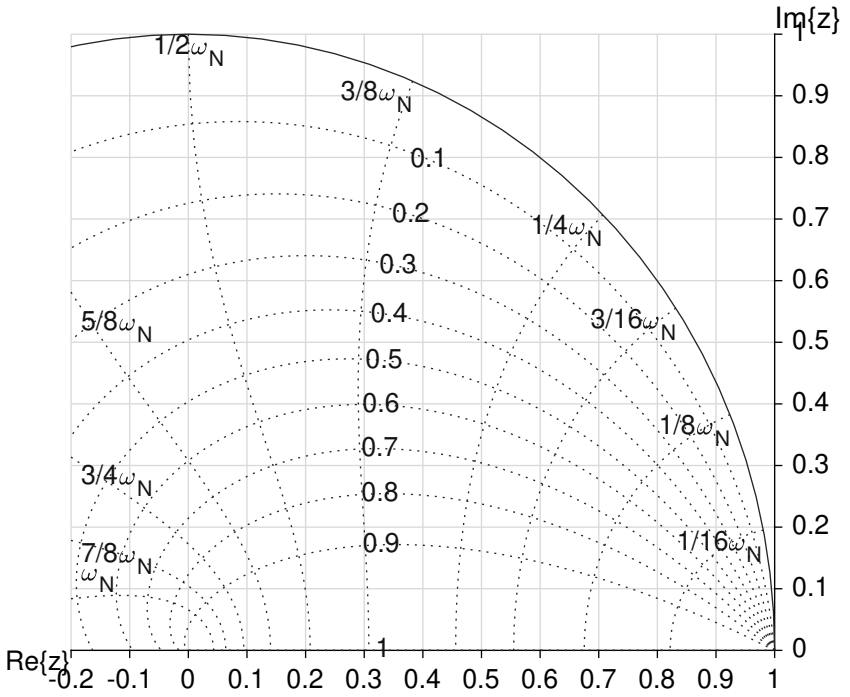


Figure 2.15: Closeup of lines of constant damping ratio and constant natural frequency are shown dotted on the diagram. The solid lines show the real and imaginary components of the z-plane locations.

at $z = 1$. The mode having a pole at $\lambda_z = 0$ corresponds to the transfer function $G(z) = \frac{1}{z}$. You may recall from earlier z-transform work that z^{-1} is the transfer function of a unit delay. That is, if we pass any signal through a system having transfer function $G(z) = \frac{1}{z}$ then we delay the signal by one sampling interval.

2.5 Matlab for discrete time systems

Many of Matlab's control commands can handle discrete time systems if you provide them with a sampling time. eg. `G=zpk([], [0.2], 3, 0.1);` $\rightsquigarrow G(z) = \frac{3}{z - 0.2}$, $t_s = 0.1$. Sometimes you want to work with a discrete time system with an unspecified sampling time. In such cases just enter `-1` as the sampling interval. Be warned that this sometimes leads to strange axis labels, as Matlab will assume $t_s = 1$ s when plotting. eg. `G=zpk([], [0.2], 3, -1);` $\rightsquigarrow G(z) = \frac{3}{z - 0.2}$, $t_s =$? If you omit the sampling time specification then Matlab will build a continuous time system and life will be *unpleasant* until you realise.

Matlab will conventionally show the response of discrete time systems using a stair plot, as shown in figure 2.16. The plot produced with `stairs` gives the impression that the signal exists (is defined) for all $t \in \mathbb{R}$. This is not true for a discrete time system, which are only defined at integer multiples of the sampling interval ($t \in \mathbb{Z}$). The stair plots *do* make sense when dealing with sampled data systems that include a zero order hold. In such systems the signals are defined for all $t \in \mathbb{R}$, but only *change* at the sampling interval. Happily it is this type of system that we more often encounter in engineering, so we can use stairs with a clear conscience.

2.6 Sampled time controllers

We commonly choose to build a controller for a continuous time system using a digital computer. One of the consequences of this choice is that we must sample the continuous time signals emitted by our system, and must similarly drive the system with discrete time signals.

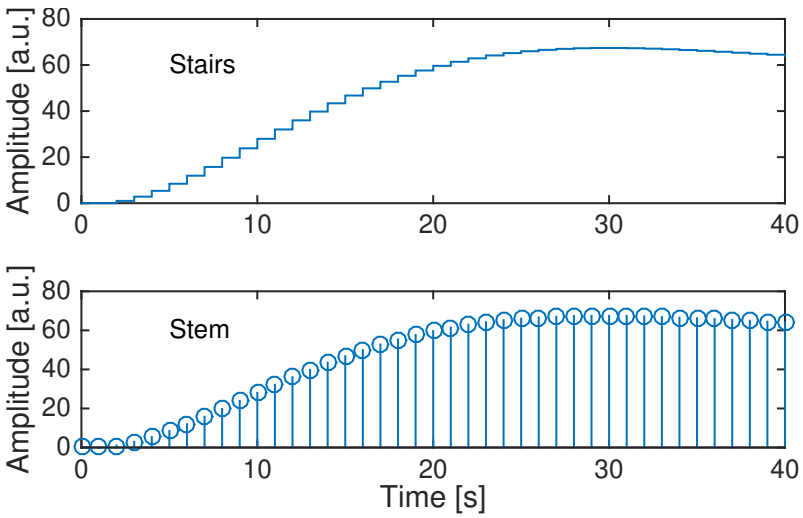


Figure 2.16: Two representations of the time responses of a discrete time system produced by Matlab.

Figure 2.17: A typical sampled data control system.

One approach to designing a controller for a sampled-time system is to design it in continuous time and then convert to discrete time afterwards. If the sampling rate is sufficiently high then we can ignore the sampling and convert our design directly.

- Convert by implementing the system in the time domain. This is commonly done with PID variants, where we construct discrete time approximations of integration and differentiation and sum the various terms using the continuous time PID gains.
- Convert mathematically by seeking some mapping between the s and z domains. That is, we design a controller $C(s)$ and seek some equivalent $\widehat{C}(z)$ that we will use in implementation.

Sampling a system always introduces delays into the system, so we should always expect a continuous time controller to perform less well after discretization.

If the sampling rate approaches the system bandwidth then we need to be careful because the delays in the discrete time portions of the system become significant. We must therefore include the effects of any anti-aliasing and reconstruction filters, as well as considering the transfer function of the sampling process. Modelling the antialiasing (and reconstruction) filters is straightforward, but in practice care should be taken with their design because of the difficulties they can cause with excess phase shift.

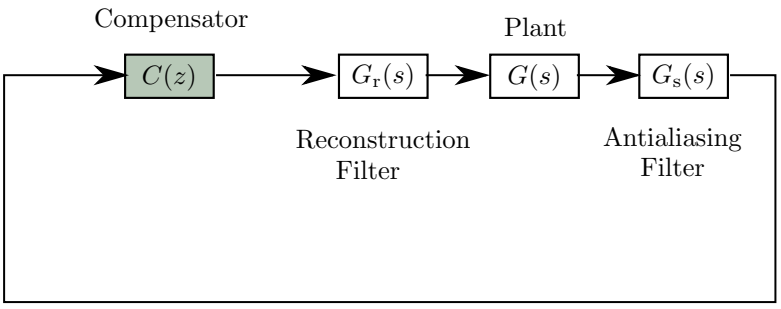


Figure 2.18: A schematic representation of a discrete time controller, from the point of view of the controller.

From the point of view of the controller, the reconstruction filter and the anti-aliasing filter appear as part of the plant, so their transfer functions should be included with that of the plant if they are significant.

Consider a signal which is held constant over each sampling time period. That is, the signal is approximated by a zeroth order polynomial of time for each sampling period.

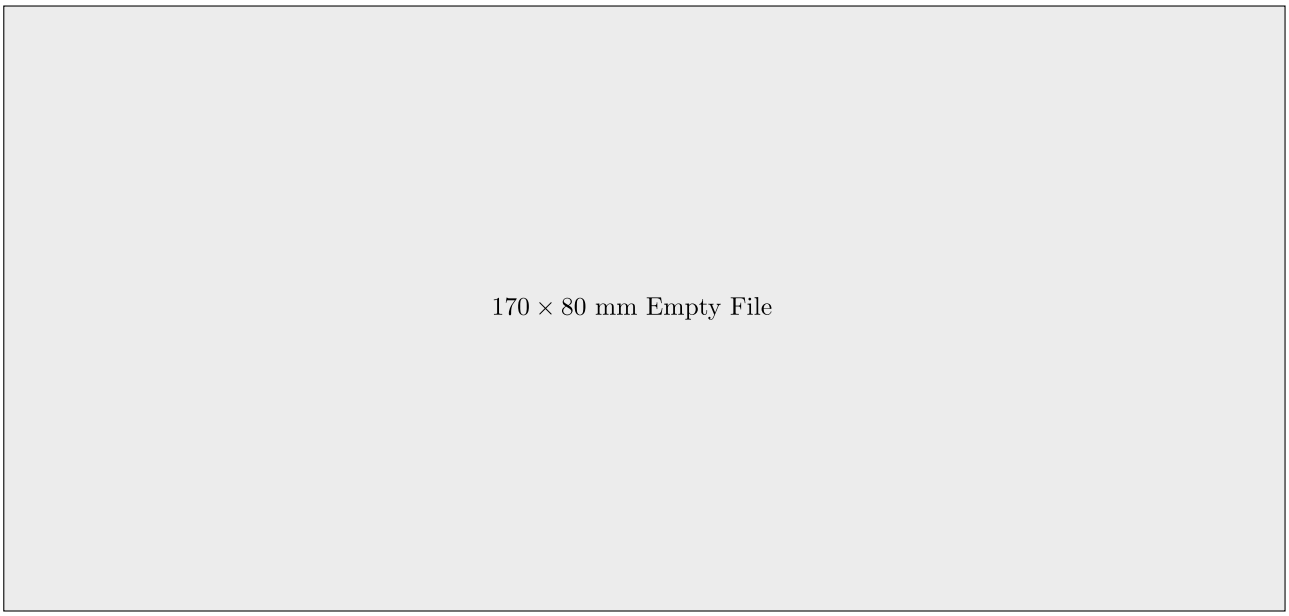


Figure 2.19: Comparison between a continuous time waveform and its notional form after being sampled at rate t_s and passed through a zero order hold.

We would like to find the continuous time transfer function of such a system. The sampler produces an impulse train, and each of those impulses is preserved for one sampling period. That is, if we sample $f(t)$ at instant $t = 0$ we get a signal $f(t)\delta(0)$, which after the hold becomes $f(0)\big(\mathrm{u}(t) - \mathrm{u}(t + t_\mathrm{s})\big)$. That is, the transfer function of the zero order hold is

$$\begin{aligned}\mathcal{L}\left\{\mathrm{u}(t) - \mathrm{u}(t + t_\mathrm{s})\right\} &= \frac{1}{s} - \frac{\mathrm{e}^{-st_\mathrm{s}}}{s} \\ &= \frac{1 - \mathrm{e}^{-st_\mathrm{s}}}{s}\end{aligned}$$

If we wish to model the effect that the zero order hold (zoh) will inevitably impose on controller transfer function we find that the exponential is awkward to work with. We could instead seek a rational function to approximate the exponential. The simplest of these is the so-called first order Padé approximant.

$$\mathrm{e}^{st_\mathrm{s}} \approx \frac{1 - s\frac{t_\mathrm{s}}{2}}{1 + s\frac{t_\mathrm{s}}{2}}$$

So, using the approximant with our sampler's transfer function

$$\begin{aligned}\frac{1 - \mathrm{e}^{st_\mathrm{s}}}{s} &\approx \frac{1}{s}\left(1 - \frac{1 - \frac{st_\mathrm{s}}{2}}{1 + \frac{st_\mathrm{s}}{2}}\right) \\ &= \frac{t_\mathrm{s}}{1 + s\frac{t_\mathrm{s}}{2}} \\ &\equiv \frac{2}{s + \frac{2}{t_\mathrm{s}}}\end{aligned}$$

Which is just a first order low pass filter. Be careful, because this is not a real filter! That is, don't let this transfer function lead you to believe that there is automatically an antialiasing filter produced by the sampler.

The Padé approximant is useful whenever we wish to approximate a time delay with a polynomial in s. The first order variant used here is often sufficient to design a control law, but if higher fidelity is

required then higher order approximants can be used. These result in a higher order polynomial being generated to use as the approximation to the exponential, which is important if one is attempting to capture the high phase shift that a delay generates beyond about $\frac{1}{t_{\text{delay}}}$.

The second order approximant to the exponential delay is already much better than the first, and is given by

$$e^{st_s} \approx \frac{1 - s\frac{t_s}{2} + s^2\frac{t_s^2}{12}}{1 + s\frac{t_s}{2} + s^2\frac{t_s^2}{12}}$$

This transfer function has two poles at $-\frac{3}{t_s} \pm \frac{\sqrt{3}}{t_s}\mathrm{j}$ and two symmetric zeros at $\frac{3}{t_s} \pm \frac{\sqrt{3}}{t_s}\mathrm{j}$.

Matlab's `pade` command can be used to generate Padé approximants to an exponential of any desired order. You can even pass it a system that includes a delay and it will return a modified system with the delay replaced neatly by a Padé approximant of the specified order.

2.6.1 Sampling rate selection

We have seen that the sampling rate plays an important role in determining the characteristics of the z-plane, the effect of sampling (both directly and through anti-aliasing filters) and we will shortly see that it similarly plays an important role in the conversion from continuous to discrete time. Clearly choosing a good sampling frequency is critical to good discrete time controller design, yet sadly there is no easy rule of thumb to tell us the right answer. We must rely on simulation to guide us. Begin with a sampling rate ten times the desired closed loop unity gain bandwidth of the system and refine from there. For difficult systems it is often necessary to sample even faster.

A well designed continuous time control system is not normally too affected by high frequency noise. This is because the plant acts as a low pass filter, so high frequency noise tends not to perturb it too much. However, one must be careful in a discrete time system, because aliasing can move the high frequency noise down into a frequency range where the system can respond. This turns non-problematic noise into a potential area of difficulty. The problem with aliasing is that it appears like it is causing an error in the control loop at the aliased frequency, so the control system responds by applying feedback at that aliased frequency. But that is *not* the frequency of the problem, so the feedback can only make things worse.

The temptation is to add a hard anti-aliasing filter to remove any high frequency noise. However, this can cause its own difficulties because of the phase shift introduced by the filter. In some cases we need to increase the sampling frequency so there is sufficient attenuation by the Nyquist frequency. Alternately, for lower order plants we might be able to use a more conservative sampling frequency and simply include the anti-aliasing filter's transfer function as part of the system to be controlled.

It is often possible to sample the input (and reference) at a much higher frequency than required, and then use some sort of signal averaging to down sample to the desired sample period. This has two main benefits:

1. Simplifies the design of the anti-aliasing filter.

2. (Potentially) improves the resolution of the sampled signals.

Reconstruction filters

Reconstruction filters can often be omitted from a control system design if the plant is sufficiently slow to do the filtering itself. However, always consider whether the system might have some high frequency dynamics that you are ignoring in a simplified model. It is easy to design a gentle controller, only to forget that the computer will implement it by sending a rapid series of step changes into your plant.

Sometimes our system contains different components having different sampling times. For example, smart sensors of various types are used in many modern control and automation systems. These sensors often include their own computation, so have their own sampling frequency. Interaction between different sampling frequencies and phases can cause subtle problems. In such systems it is therefore wise to err on the side of a higher frequency for the main control system sampling rate. At least 20 times the closed loop bandwidth is advisable.

2.7 Converting continuous time systems to discrete time

If we have a continuous time system that we want to convert to discrete time then there are a variety of transformations available. We will discuss a few alternatives, but not attempt a comprehensive survey. See Ogata "Discrete Time Control Systems" for more detail. This conversion arises in two main situations,

1. In converting a continuous time plant into discrete time so that design can be done in the discrete domain. For this application it is important to include any anti-aliasing or reconstruction filters as well as the effect of the sampler.

2. In converting a continuous time controller into discrete time for implementation. In this case the anti-aliasing or reconstruction filters and the sampler would not be included (they are in the hardware!).

The bilinear transformation (or Tustin's approximation) replaces s with expressions in z according to

$$s \mapsto \frac{2}{t_\mathrm{s}} \frac{z-1}{z+1}$$

The bilinear transformation maps the left half of the s-plane maps to the inside of the unit circle in the z-plane, so any transfer function that is stable in one domain is guaranteed to be stable in the other. Similarly a minimum phase system will remain so after the transformation. Be aware that the bilinear transform produces significant mismatch in both the time and frequency domain characteristics of the resulting discrete time approximation. It is sufficient only in cases where the sampling rate is quite fast compared to the dynamics of the system.

The bilinear transformation does not do a good job of matching the high frequencies relative to the sampling frequencies. For example, if you have a filter with at a relatively high frequency and then apply the normal bilinear transformation then you will find that the resulting continuous time filter approximation will have a different corner frequency. Prewarping provides a partial solution to this problem, by first shifting root locations so that they will be better placed after a subsequent bilinear transformation. To do this, identify all of the relevant root locations a and replace them according to

$$a \mapsto \frac{2}{t_\mathrm{s}} \tan \frac{a t_\mathrm{s}}{2}$$

This is followed by the bilinear transformation as normal.

If we have a transfer function $G(s)$ (including the anti-aliasing and/or reconstruction filters), then we can seek a z-domain transfer function that matches the impulse response at the sampling instants. That is, we want to find a $\widehat{G}(z)$ such that

$$\begin{aligned} \mathcal{Z}^{-1}\left\{\widehat{G}(z)\right\} &= t_\mathrm{s}\, \mathcal{L}^{-1}\left\{G(s)\right\}\big|_{t=kt_\mathrm{s}} \\ \implies \widehat{G}(z) &= t_\mathrm{s} \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{G(s)\right\}\right\} \end{aligned}$$

That is, we convert the continuous time transfer function into the time domain, and then convert again to the z domain. Using this conversion leads to a matched impulse response, but makes no promises about any other correspondences between the continuous and discrete transfer functions.

If we have a transfer function $G(s)$ (including the anti-aliasing and/or reconstruction filters), then we can seek a z-domain transfer function that has the same step response at the sampling instants. That is, we want to find a $\widehat{G}(z)$ such that

$$\begin{aligned} \mathcal{Z}^{-1}\left\{\frac{1}{1-z^{-1}}\widehat{G}(z)\right\} &= \mathcal{L}^{-1}\left\{\frac{1}{s}G(s)\right\}\Big|_{t=kt_\mathrm{s}} \\ \implies \widehat{G}(z) &= (1-z^{-1})\mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{G(s)}{s}\right\}\right\} \end{aligned}$$

or, equivalently

$$\widehat{G}(z) = \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{1-\mathrm{e}^{st_\mathrm{s}}}{s}G(s)\right\}\right\}$$

In this method we replace each root at r_s the continuous time system with a root at the corresponding location $r_\mathrm{z} = \mathrm{e}^{r_\mathrm{s}t_\mathrm{s}}$ in the z domain. One complication of this method is that we must also map infinite zeros and poles that we do not normally consider in a continuous transfer function. A continuous time transfer functions can always be considered as having an equal number of poles and zeros, but any excess roots that don't appear in the "normal" transfer function are at (complex) infinity.

$$\text{For example, informally } \frac{a}{s+a} \equiv \frac{a(s+\infty)}{s+a}$$

These excess zeros (or more rarely poles) from infinity must be mapped to $z = -1$ (as z=-1 is the highest possible frequency in the z-plane, it is in some sense our best approximation to infinity). That is, we find the difference between the number of poles and zeros in the continuous time transfer function and place a zero at $z = -1$ for each.

A second minor complication that needs to be dealt with is to match the gain of the two filters. For a low pass filter we simply compare the dc gains, at $G(s)|_{s=0}$ and $G(z)|_{z=1}$. For high pass filters we match the gains $G(s)|_{s\to\infty}$ and $G(z)|_{z=-1}$.

Comparison of conversion methods

Consider the low pass filter $G(s) = \dfrac{1000}{s+1000}$ which has a corner frequency at 1000 rads^{-1} and unity dc gain. Find the equivalent z-domain transfer functions using each of the conversion methods, and compare the time and frequency domain performances.

2.8 Implementation

Our control laws will look like polynomials in z^{-1}. We need to convert those back into difference equations for implementation. For example, consider a second order discrete time control law having the transfer function

$$\begin{aligned} C(z) &:= \frac{U(z)}{E(z)} \\ &= \frac{1 + A_1 z^{-1} + A_2 z^{-2}}{1 + B_1 z^{-1} + B_2 z^{-2}} \\ \implies \left(1 + B_1 z^{-1} + B_2 z^{-2}\right) U(z) &= \left(1 + A_1 z^{-1} + A_2 z^{-2}\right) E(z) \end{aligned}$$

We now inverse z transform both sides.

$$\begin{aligned} u(t) + B_1 u(t-1) + B_2 u(t-2) &= e(t) + A_1 e(t-1) + A_2 e(t-2) \\ \text{That is, } u(t) &= e(t) + A_1 e(t-1) + A_2 e(t-2) - B_1 u(t-1) - B_2 u(t-2) \end{aligned}$$

Remember here that $t \in \mathbb{Z}_+$, so $e(t)$ is the error signal measured at time t, while $e(t-1)$ is the error signal one sampling period earlier. Similarly for the control signal $u(t)$ and its variants ($u(t)$ does not denote a step function here!)

It is easy to write code that takes a long time to do the calculations required to produce each output. An ideal discrete time systems produces the output $u(t)$ as soon as $e(t)$ is known, but in reality of course, it takes time to calculate $u(t)$. The most obvious trap here is to shuffle previous inputs $e(t) \to e(t-1)$ (and so on) before producing an output. We also often need to do things like data logging, error checking etc as part of the normal operation of the control code. Do not do those things until *after* producing

an output! For example, when implementing the previous filter you should precalculate a temporary variable

$$\text{temp} = A_1 e(t-1) + A_2 e(t-2) - B_1 u(t-1) - B_2 u(t-2)$$

You can then very quickly perform:

1. Read $e(t)$
2. Calculate $u(t) = e(t) + \text{temp}$
3. Write $u(t)$

2.9 Converting discrete time system to continuous time

On occasion it is convenient to convert a discrete time system to continuous time. That is, we would like to take some $G(z)$ and find some "equivalent" $\hat{(}G)(s)$ so that the conversion preserves some properties of the system we care about. The exponential involved makes this awkward, so instead of seeking a perfect conversion, we instead make various simplifying approximations that are good enough for the task we have in mind. The conversion methods are the same as those discussed above, but done in reverse.

To assess stability of a discrete time system we need to find whether there are any poles outside the unit circle in the z-plane. We already know some methods for assessing stability in continuous time, such as the Routh-Hurwitz and Nyquist's stability criterion. A simplified form of the bilinear transformation can be used to convert a discrete time system to continuous time *for the purpose of assessing stability.* The bilinear transformation has the property that the inside of the unit circle in the z-plane maps to the left half of the s-plane, but that is also true for the simpler version

$$z \mapsto \frac{1+r}{1-r}$$

That is we replace each z appearing in $G(z)$ with $\frac{1+r}{1-r}$ to find a transfer function that we can then use with Routh-Hurwitz or Nyquist. Note that we are here using r to be a variable somewhat like s, but the use of r is intended to remind the user that we are not attempting a full conversion. The normal bilinear transformation does a better job of mapping between the unit circle in the z-domain and the left half of the s-plane, but at the price of additional unnecessary algebra if one only intends to do a Routh-Hurwitz test.

The Jury test is a algebraic method that directly tests for discrete time poles lying within the unit circle. It can therefore be used in much the same way as the Routh-Hurwitz test in continuous time. However, in many cases the Jury test is algebraically awkward. Given the simplicity of the bilinear transformation and the availability of computation tools, there seems little justification in learning the Jury test, but it is perhaps useful to know that it exists.

3 Introduction to Modern Control

The tools of classical control are sufficient for building controllers for many real world applications. However, sometimes that we need some additional design power. In ECEN415 we will use *modern control* to deal with a richer set of problems. In particular we will look at

- Systems having multiple outputs that we can control with multiple inputs;
- Systems that exist in discrete rather than continuous time;
- *Optimal* controllers, which allow us to be confident that we have designed the best possible controller.

Conceptually this follows from classical control, but we will be using a different set of underpinning tools. These are are based on linear algebra rather than Laplace or z transforms.

While classical control techniques can be further developed to tackle some of these problems, in general the field has adopted an alternative set of techniques which are collectively known as *modern control*. Modern control (first developed in the 1950's-60's) provides the underlying language of most control developed since. Modern control has very nice mathematical properties, but it is sometimes harder to see the link to the real world than is the case with classical control. You should try to maintain a classical control view to gain insight. For example, you should look at Bode or Nyquist plots of any compensators that we design.

3.1 An example 2 input, 2 output system

Sometimes we can use a classical control approach to deal with systems having multiple inputs and/or outputs. By way of motivation, consider a system that has two inputs $\{I_{\text{heater}}, S_{\text{valve}}\} := \{u_1, u_2\}$ and two outputs $\{\text{flow}, \text{temperature}\} := \{y_1, y_2\}$ as shown in figure 3.1.

We could model this system as a two input, two output system and describe its input-output relations by a set of transfer functions. For example, we could define $G_{11}(s) := \frac{Y_1(s)}{U_1(s)}$, and have a matrix $\boldsymbol{G}$ that captures all four transfer functions.

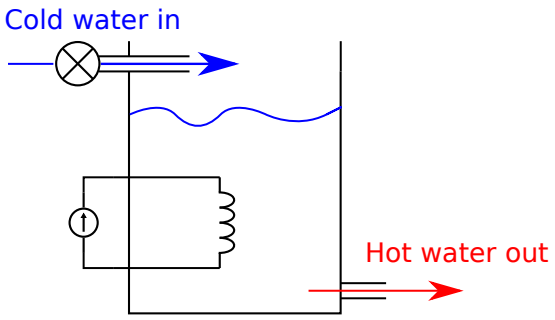


Figure 3.1: A tank intended to supply hot water.

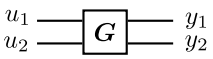


Figure 3.2: A system described by a matrix of transfer functions ($\boldsymbol{G}$) which has two inputs (u_1, u_2) and two outputs (y_1, y_2).

Sometimes only a single input affects each output. These systems are essentially a set of uncoupled SISO systems and we could proceed to design multiple compensators using classical control techniques. See for example figure 3.3, which shows the time response of a system of this kind. Notice that a step change in u_1 only effects y_1 and u_2 only effects y_2. Figure 3.4 shows the matrix of Bode plots relating the two inputs to the two outputs. Notice that only the transfer functions relating particular pairs of inputs and output have a meaningful plot. Though it is not visible in the figure, the other two transfer functions have zero gain.

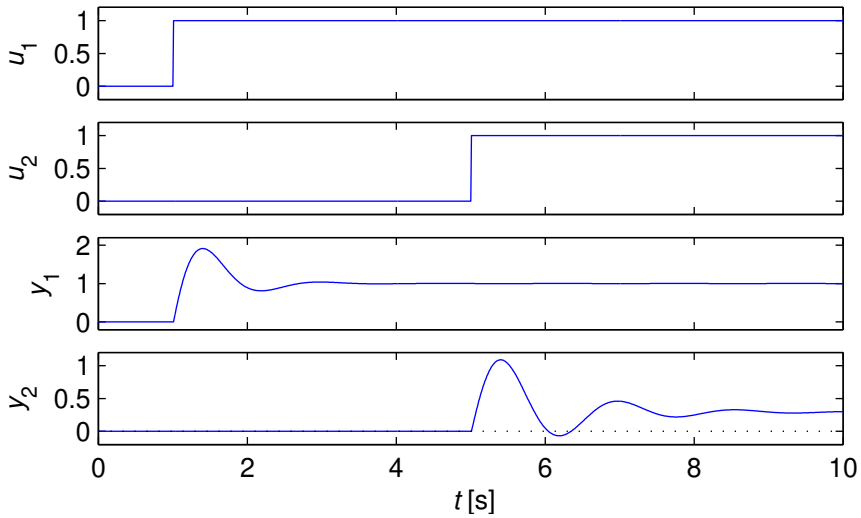


Figure 3.3: Step response of a two-input two-output system with uncoupled dynamics.

A typical implementation of a control system would be as shown in figure 3.5.

However, many systems show strong coupling between all of the inputs and the outputs. That is, all of the inputs have effects on all of the outputs, or at least have effects on enough that the coupling cannot be safely ignored.

An example of the type of system where a full array of compensators might be needed is shown in figure 3.7, which shows the time response and figure 3.8 which shows the matrix of Bode plots. Notice that each of the two inputs now effects the two outputs and that all four Bode plots now show non-zero gain.

Classical control can become unmanageable for such systems unless the coupling is mild. We may be able to ignore the coupling and essentially treat the minor coupling as a disturbance. This approach is indicated in figure 3.9. This has the virtue of simplicity, and will sometimes work fine.

In the example shown in figure 3.9 we design the compensator $C_{11}(s)$ to control y_1, and simply ignore the existence of the cross-coupling transfer function $G_{21}(s)$. When $C_{11}(s)$ takes action via u_1 it will disturb y_2, but (hopefully) compensator $C_{22}(s)$ can deal with that disturbance. Of course, the action of $C_{22}(s)$ will in turn disturb y_1 again. In practice this is always worth exploring, as you may be able to devise a simple and robust controller. This tends to work best if the coupling is very mild and/or when the frequency responses of various parts of the systems are well isolated.

Sometimes there is enough coupling that we need to use full feedback control, which quickly becomes complicated. If we have m inputs and p outputs then we have $m \times p$ different interdependent compensators to design. Recall that even a PID compensator requires the determination of three parameters, so this quickly becomes a difficult problem. This approach is illustrated in figure 3.10.

Happily the tools of modern control provide us with a much simpler alternative to this approach. Modern control automatically designs all necessary feedback paths simultaneously.

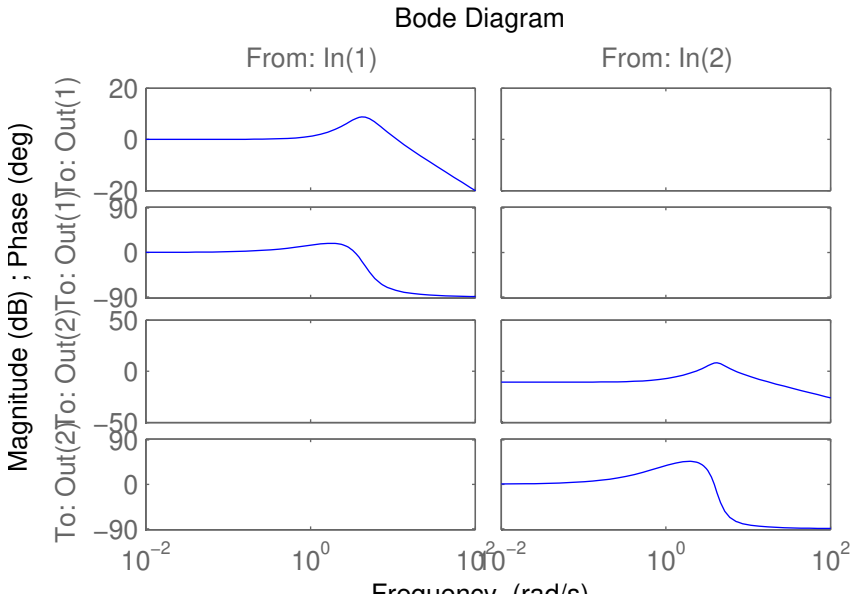


Figure 3.4: Bode plots of a two-input two-output system with uncoupled dynamics. Each Bode plot shows a frequency response that indicates the effect that a particular input has on a particular output.

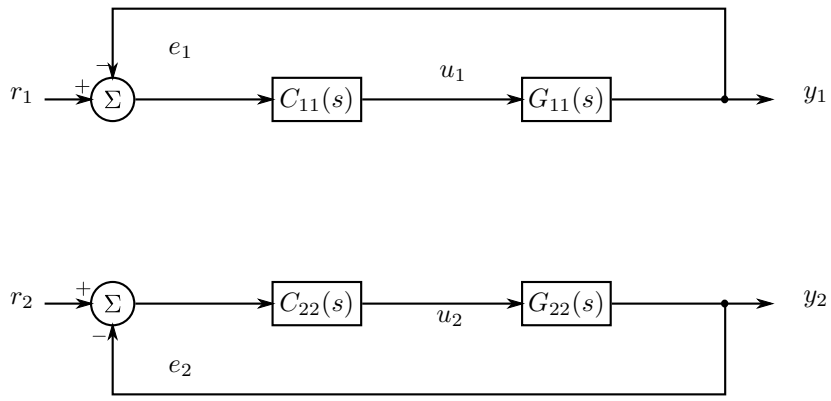


Figure 3.5: Schematic of a two-input, two-output system with feedback control that ignores the cross coupling.

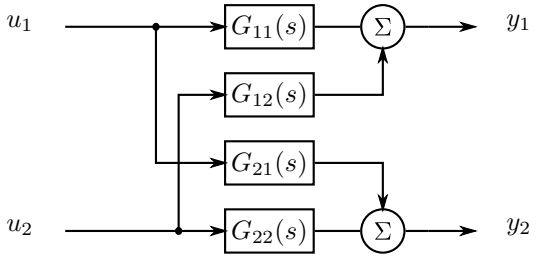


Figure 3.6: Schematic of a two-input, two-output coupled system.

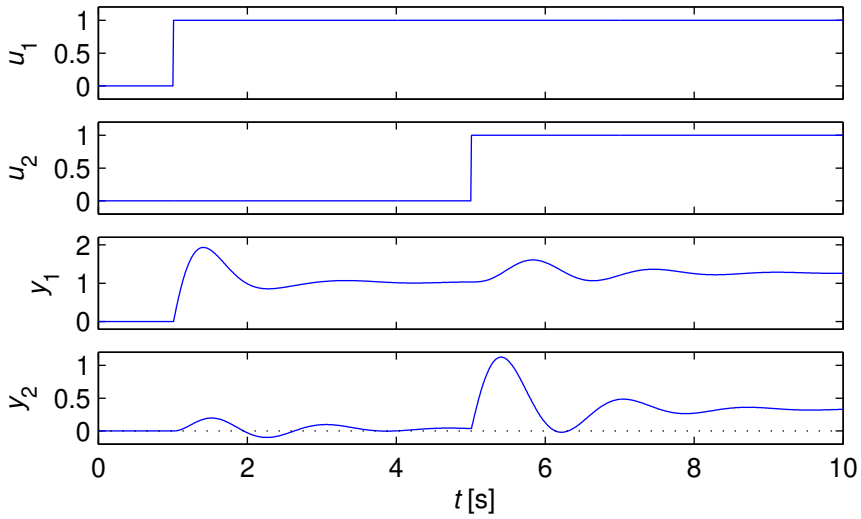


Figure 3.7: Step response of a two-input two-output system with coupled dynamics.

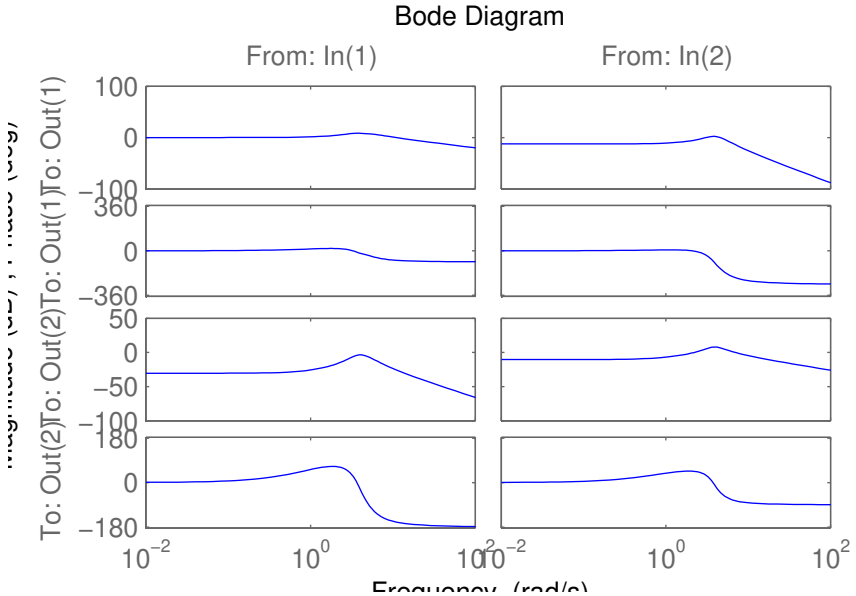


Figure 3.8: Bode plots of a two-input two-output system with coupled dynamics.

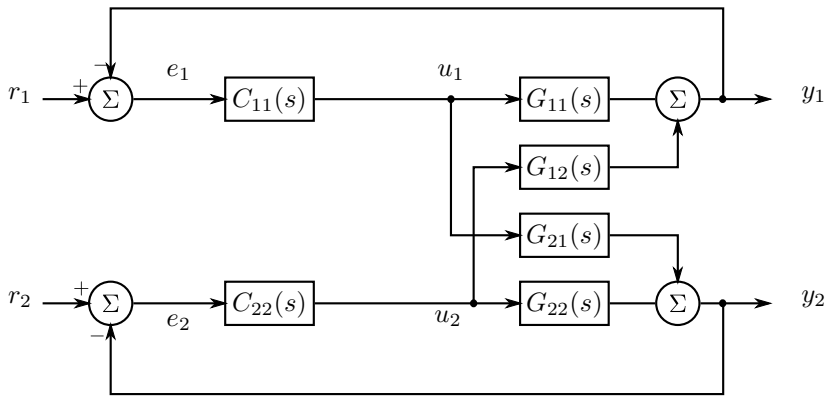


Figure 3.9: Schematic of a two-input, two-output system with feedback control that ignores the cross coupling.

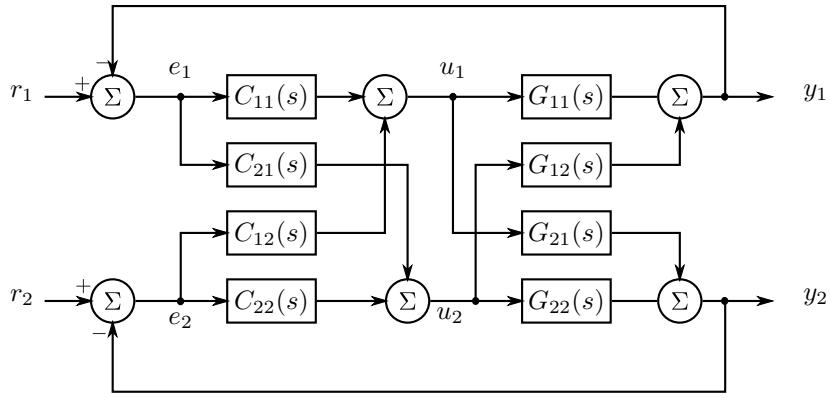


Figure 3.10: Schematic of a two-input, two-output system with complete feedback control.

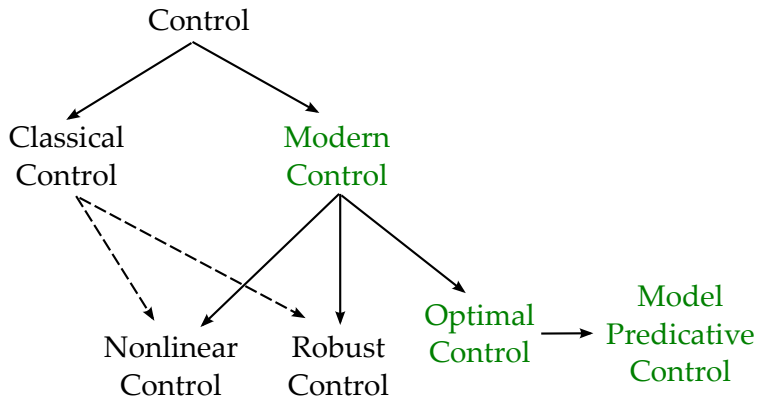


Figure 3.11: Dependencies in subfields of control engineering.

Modern control is the basis for a wide variety of more advanced control techniques. These techniques enable us to deal with noise, with uncertainty and with nonlinearities in the plant and let us maximise a specified performance criterion.

Figure 3.11 shows the relation between various branches of control engineering.

We will build modern control in a number of steps. There are several salient features of our approach that contrast with classical control methods.

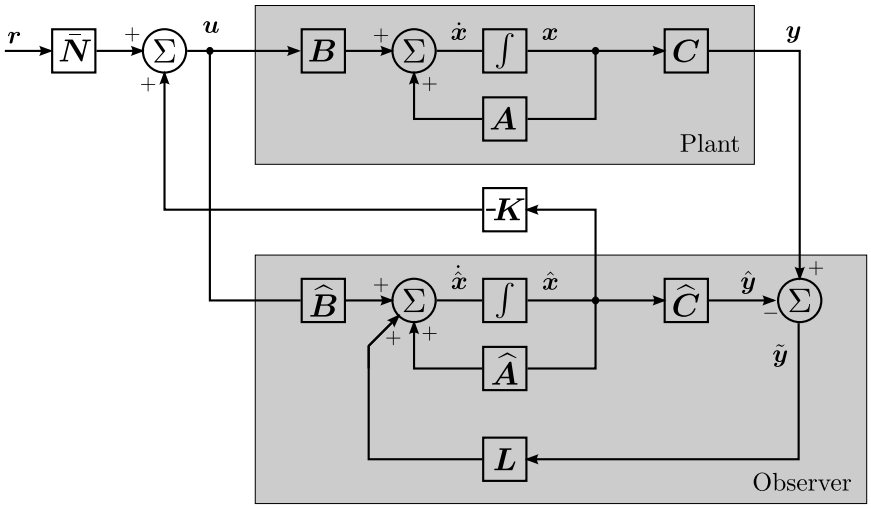


Figure 3.12: A state space system, including the plant model, estimated state feedback taken from a Luenberger observer and a prefilter $\bar{N}$ to set the desired dc gain.

- We use a *state space* model to describe a system.
- The model describes the internal structure of a plant and we try to infer internal information about the system using an *observer*.
- Linear algebra is used to manipulate the models.

This can be contrasted with classical control, where we only considered the behaviour at the input and output of each block, and described it with a transfer function.

The Plan: Outline of Lectures

The course will include the following topics, in more or less this order.

1. State space modelling of electrical and mechanical systems.
2. Discrete time systems and conversion to discrete time.
3. Solving for the time response of a state space system.
4. Converting to and from state space formulation.
5. Compensators for regulator problems.
6. Compensators for servo problems, including integral action.
7. Optimal (LQ) control.
8. State observers
9. Nonlinear systems or MPC or digital controller implementation or robust control or system identification or loop shaping or ...

4 A Review of Linear Algebra

4.1 Vectors

A vector can be considered as containing a list of coordinates for the head of the vector. For example consider a two element column vector containing real numbers:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

where x_1 and x_2 are the *components* of $\boldsymbol{x}$. The vector $\boldsymbol{x}$ can describe *any* position in a two dimensional space of real numbers. Hence we say $\boldsymbol{x} \in \mathbb{R}^2$.

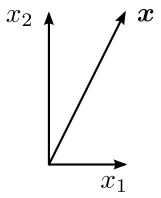


Figure 4.1: A vector $\boldsymbol{x} \in \mathbb{R}^2$.

To generalise, an n element vector can describe an arbitrary location in an n-dimensional space $\mathbb{R}^n$.

A vector quantity is represented by a bold italic symbol: $\boldsymbol{x}$. A scalar quantity is represented by (normal) italics: x. We often refer to the elements of a vector using subscripts,

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

The transpose of a vector is indicated by the $^\mathsf{T}$ superscript. This notation is very convenient for saving space when writing out the components of a vector;

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\mathsf{T} \quad .$$

4.2 Matrices

Matrices are two dimensional collections of elements (usually real numbers for this course).

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

The scalar in the i-th row and j-th column of $\boldsymbol{C}$ is denoted c_{ij}. The vector that is the i-th column of $\boldsymbol{C}$ is denoted $\boldsymbol{c}_i \in \mathbb{R}^n$. The vector that is the i-th row of $\boldsymbol{C}$ is denoted $\boldsymbol{c}_i^\mathsf{T} \in \mathbb{R}^{1 \times m}$.

The entries of a matrix need not be real. We will occasionally see matrices with complex elements, or more exotic example with rational functions (transfer functions) or functions of time as their elements.

We can form a matrix by concatenating a number of other matrices, though the constituent matrices must have compatible sizes.

For example, for $\boldsymbol{A} \in \mathbb{R}^{2\times2}$, $\boldsymbol{B} \in \mathbb{R}^{2\times1}$, $\boldsymbol{C} \in \mathbb{R}^{1\times2}$ and $\boldsymbol{D} \in \mathbb{R}^{1\times1}$.

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_{11} \\ a_{21} & a_{22} & b_{21} \\ c_{11} & c_{12} & d_{11} \end{bmatrix}$$

The *transpose* of a matrix is formed by rearranging its rows into columns.

$$\boldsymbol{A}^{\mathsf{T}} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

4.2.1 Mapping with a matrix

For our purposes, a matrix described a "machine" that takes a vector as an input and produces another vector as output. We say that the matrix produces a mapping from one vector to another. Consider $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^2, \boldsymbol{C} \in \mathbb{R}^{2\times 2}$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{11}x_1 + c_{12}x_2 \\ c_{21}x_1 + c_{22}x_2 \end{bmatrix}$$

Notice that each y_i is a linear combination of the components of $\boldsymbol{x}$. A matrix therefore describes a *linear* mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$.

A matrix is essentially a convenient method for expressing a set of simultaneous equations,

$$y_1 = c_{11}x_1 + c_{12}x_2$$
$$y_2 = c_{21}x_1 + c_{22}x_2$$

Note that a matrix element c_{ij} describes how the j-th "input" affects the i-th output. The i-th row of a matrix describes the gains from all of the inputs to the i-th output. The j-th column describes the effect that the j-th input has on all of the outputs.

The matrix/vector multiply above is how most people are taught to complete the operation. Consider an alternate, but equivalent, method.

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{11}x_1 + c_{12}x_2 \\ c_{21}x_1 + c_{22}x_2 \end{bmatrix}$$
$$\boldsymbol{y} = \boldsymbol{c}_1 x_1 + \boldsymbol{c}_2 x_2$$

where $\boldsymbol{c}_i$ denotes the i-th column of $\boldsymbol{C}$. Matrix multiplication produces a linear combination of the columns of the matrix. This view of matrix/vector multiplication will prove very useful.

For example, let's consider an $\boldsymbol{x}$ vector $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and find the $\boldsymbol{y}$ that results from the mapping $\boldsymbol{y} = \begin{bmatrix} 3 & 0 \\ 1 & -1 \end{bmatrix} \boldsymbol{x}$. We can compute this numerically, which yields $\boldsymbol{y} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$, but we can also examine the calculation geometrically as in figure 4.2.

4.2.2 The Identity matrix

One important mapping is the one that maps a vector to itself (does nothing). This corresponds to the identity matrix $\boldsymbol{I}$.

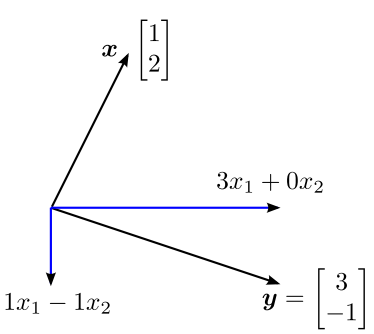
$$\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x}, \quad \forall \boldsymbol{x}$$

The identity matrix has ones on its diagonal and is zero elsewhere.

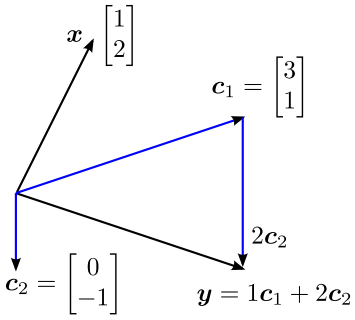
$$\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The dimension of $\boldsymbol{I}$ is normally apparent from context, but if not we provide a subscript as shown.

There are several common mapping matrices that appear in many contexts. They include



(a) Matrix mapping the "normal" way.



(b) Matrix mapping as a combination of the columns of the matrix.

Figure 4.2: Two alternate but equivalent pictures of mapping a vector $\begin{bmatrix}1\\2\end{bmatrix}$ with the matrix $\begin{bmatrix}3 & 0\\1 & -1\end{bmatrix}$.

- Dilation matrices.
- Reflection matrices.
- Shear matrices.
- Rotation matrices.

We will look at some of these in two dimensions to build intuition about how mappings work. However, the various operations extend naturally into higher dimensions. In the graphs that follow, we have applied the mapping $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ for a set of $\boldsymbol{x}$ values to illustrate the global behaviour of the map. Normally we will map single $\boldsymbol{x}$ vectors, but we will also find it useful to consider this larger picture.

4.2.3 Scaling matrices

When the matrix is a multiple of $\boldsymbol{I}$ the scaling is isotropic. The effect of such mappings is illustrated in figure 4.3.

We can have more general mapping matrix, which scales the different directions by different amounts. For example, the matrix $\boldsymbol{C} = \begin{bmatrix}2 & 0\\0 & 0.7\end{bmatrix}$ scales inputs by a factor of 2 in the x_1 dimension and by 0.7 in the x_2 dimension as shown in figure 4.4.

A negative element on the diagonal of the matrix results in reflection. Figure 4.5 shows the effect of such mappings.

When the off-diagonal elements of the mapping matrix are non-zero, then we get coupling between the different dimensions as shown in figure 4.6.

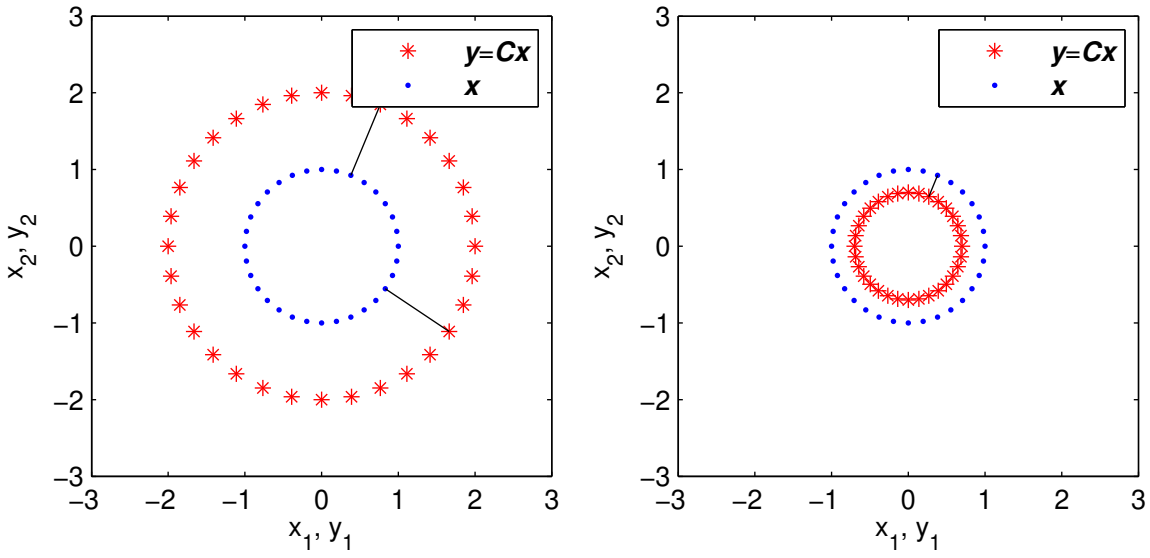
4.2.4 Rotation matrices

A rotation of θ radians is produced by the matrix $\boldsymbol{C}$ with form $\boldsymbol{C} = \begin{bmatrix}\cos\theta & -\sin\theta\\\sin\theta & \cos\theta\end{bmatrix}$. The figure in 4.7 shows an example with $\theta = 5°$.

4.2.5 Mapping with a non-square matrix

We can produce a mapping with a non-square matrix. In this case the number of elements in the output vector will have a different number of elements from the input vector.

$$\text{Example:} \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \begin{bmatrix}1 & 0 & 2\\3 & 1 & 2\end{bmatrix}\begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}1\\3\end{bmatrix} + 2\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}2\\2\end{bmatrix} = \begin{bmatrix}3\\7\end{bmatrix}$$



(a) Effect of the matrix mapping $\boldsymbol{y} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \boldsymbol{x}$.

(b) Effect of the matrix mapping $\boldsymbol{y} = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix} \boldsymbol{x}$.

Figure 4.3: The blue points indicate a set of $\boldsymbol{x}$ vectors that are mapped by $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ to the corresponding $\boldsymbol{y}$ set shown in red. The black line shows the particular correspondence between a examples for $\boldsymbol{x}$ and $\boldsymbol{y}$.

In this case we have mapped a vector in $\mathbb{R}^3$ to a vector in $\mathbb{R}^2$. In general, mapping with a matrix in $\mathbb{R}^{n \times m}$ takes an input in $\mathbb{R}^m$ and produces an output in $\mathbb{R}^n$.

Mapping with fat matrix produces a lower dimensional output vector. It does this by "squashing" the input space down into lower dimension.

Mapping with a thin matrix pushes the input vector into a *higher* dimensional space. Note though that the output vector is in a subspace that has the dimension (at most) the same as the input. In the example, every location in the two dimensional input is mapped somewhere into a plane ($\in \mathbb{R}^2$) that is living in $\mathbb{R}^3$.

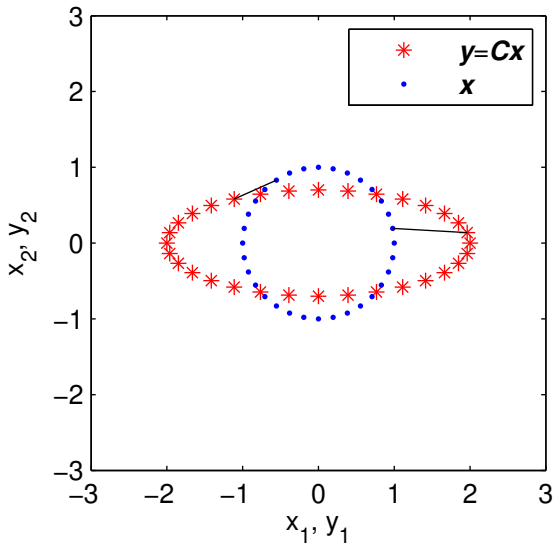
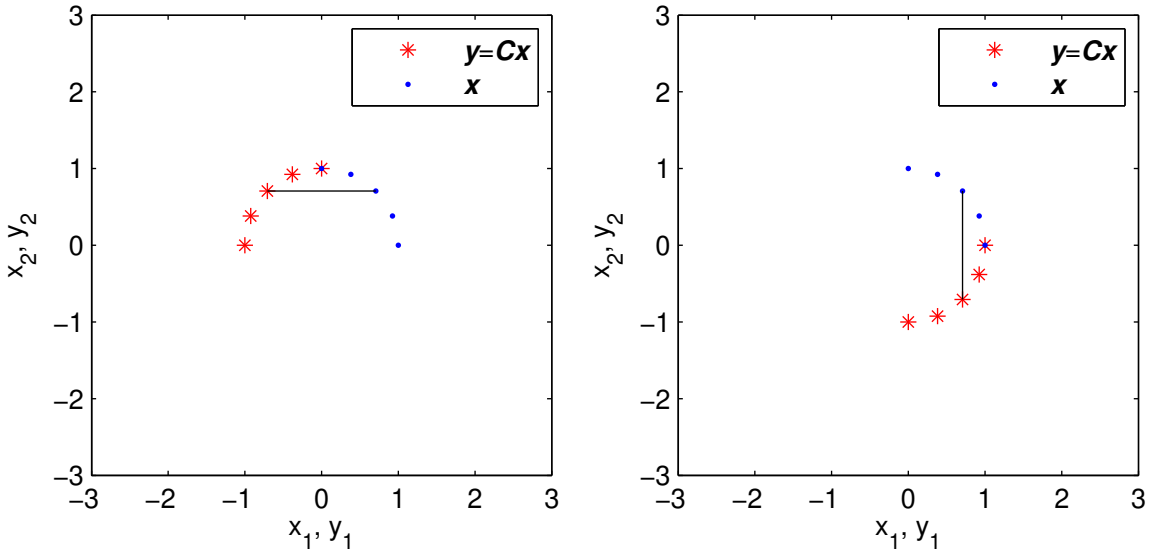


Figure 4.4: Effect of the anisotropic mapping arising from $\boldsymbol{y} = \begin{bmatrix} 2 & 0 \\ 0 & 0.7 \end{bmatrix} \boldsymbol{x}$.



(a) Effect of the matrix mapping $\boldsymbol{y} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}$.

(b) Effect of the matrix mapping $\boldsymbol{y} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \boldsymbol{x}$.

Figure 4.5: The blue points indicate a set of $\boldsymbol{x}$ vectors that are mapped by $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ to the corresponding $\boldsymbol{y}$ set shown in red. The black line shows the particular correspondence between a examples for $\boldsymbol{x}$ and $\boldsymbol{y}$.

4.3 Range, Rank and Span

Consider a vector $\boldsymbol{x} \in \mathbb{R}^2$ and a matrix $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$. If we form $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, is it possible for $\boldsymbol{y}$ to be anywhere in $\mathbb{R}^2$?

There is no guarantee that a mapping preserves the dimensionality of the input vector. This depends on the structure of $\boldsymbol{C}$. Consider $\boldsymbol{C} = \begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix}$ that was used in the figure 4.11. $\boldsymbol{y}$ must be a linear combination of the columns of $\boldsymbol{C}$, but the columns of $\boldsymbol{C}$ are clearly not independent, because the second column can be expressed as a multiple of the first. The dimension of the result of matrix multiplication is equal to the number of independent columns in the matrix. In our example we do not have independent

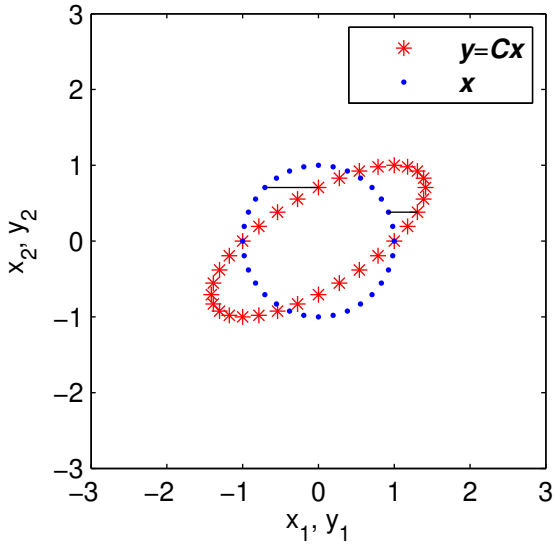


Figure 4.6: Effect of a shear mapping with $\boldsymbol{y} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}$

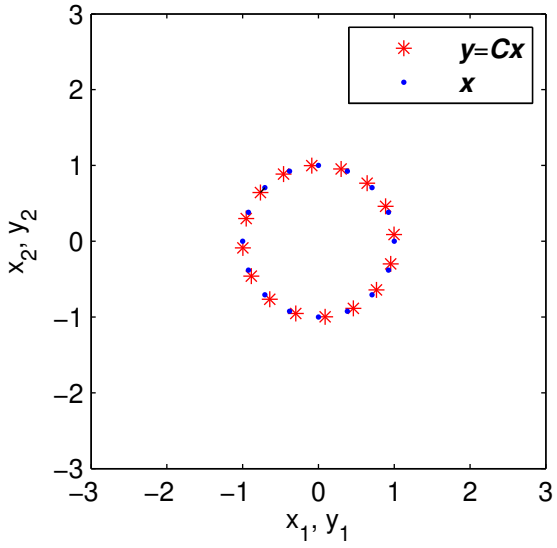


Figure 4.7: Effect of a rotation mapping with $\boldsymbol{y} = \begin{bmatrix} \cos 5^\circ & -\sin 5^\circ \\ \sin 5^\circ & \cos 5^\circ \end{bmatrix} \boldsymbol{x}$

columns in $\boldsymbol{C}$, so the resulting mapping produces a one dimensional space.

The outputs of a mapping are vectors, but they do not necessarily "fill" all of the possible space. We call the subspace that the outputs of $\boldsymbol{Cx}$ inhabit the *range* of the mapping corresponding to the matrix $\boldsymbol{C}$. A matrix takes an arbitrary input vector from its input space and maps it to somewhere within the matrix's range. We denote the range of a matrix by $\mathcal{R}$. So, for example, if

$$\boldsymbol{y} = \boldsymbol{Cx} \text{ then } \boldsymbol{y} \in \mathcal{R}(\boldsymbol{C}) \quad \forall \boldsymbol{x} \in \mathbb{R}^n \quad .$$

If we had a mapping with a $\mathbb{R}^{3\times 3}$ matrix that only had *two* independent columns, then the mapping would generate a three dimensional output. However, we would find that *all* outputs are in a two dimensional subspace. That is, the matrix mapping acts like it only has two dimensions.

If the $\mathbb{R}^{3\times 3}$ matrix that only had *one* independent column, then the mapping would generate a three

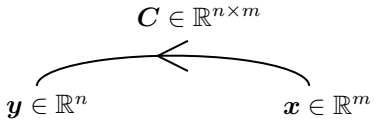


Figure 4.8: Mapping from an input vector $\boldsymbol{x}$ to an output vector $\boldsymbol{y}$ via the matrix $\boldsymbol{C}$.

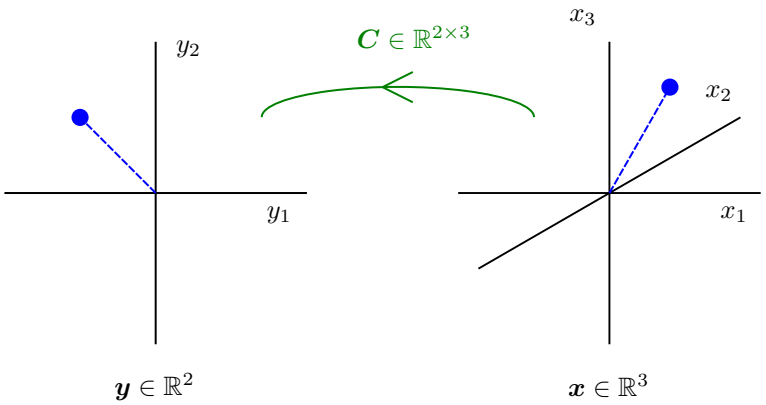


Figure 4.9: Mapping with a matrix from $\mathbb{R}^{2\times 3}$ reduces a three dimensional input to a two dimensional output.

dimensional output. However, we would find that *all* outputs are on a one dimensional subspace (a line).

We will also draw a link between the range of $\boldsymbol{C}$ and the space *spanned* by the columns of $\boldsymbol{C}$;

$$\mathcal{R}(\boldsymbol{C}) = \text{span}(\boldsymbol{c}_i)$$

For $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$ it is trivial to see whether the columns are independent. It gets a bit harder for larger matrices. For example, are the columns of the following matrix independent?

$$\boldsymbol{C} = \begin{bmatrix} 1 & -7 & 3 \\ 3 & 3 & 1 \\ 2 & -8 & 4 \end{bmatrix}$$

In fact the second column is twice the first column subtract three times the third column ($\boldsymbol{c}_2 = 2\boldsymbol{c}_1 - 3\boldsymbol{c}_3$). Formally we would prove linear independence of the columns of a matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}$ by showing that there exists no non-zero set of a_i's for which

$$a_1\boldsymbol{c}_1 + a_2\boldsymbol{c}_2 + \cdots + a_m\boldsymbol{c}_m = 0.$$

In an introductory algebra course you would have seen how to reduce $\boldsymbol{C}$ to reduced row echelon form so that the independence (or otherwise) becomes obvious. The reduced row echelon form of the above matrix is

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & \frac{2}{3} \\ 0 & 1 & -\frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

from which it should be apparent that the columns of $\boldsymbol{C}$ are not independent. We will not need to do that in this course (though you can use matlab's `rref(C)` if you wish. We will see a number of quick methods for determining the number of independent columns as we proceed.

4.3.1 Rank

We call the number of independent columns of a matrix its *rank*.

- The dimension of the space formed by mapping with a matrix is equal to its rank.

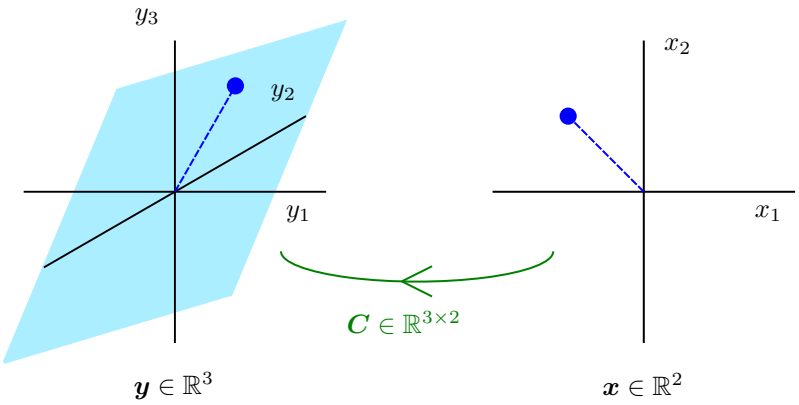


Figure 4.10: Mapping with a matrix from $\mathbb{R}^{3\times 2}$ embeds the two dimensional input vector within a three dimensional output. All of the inputs lie in a plane within $\mathbb{R}^3$ in this example.

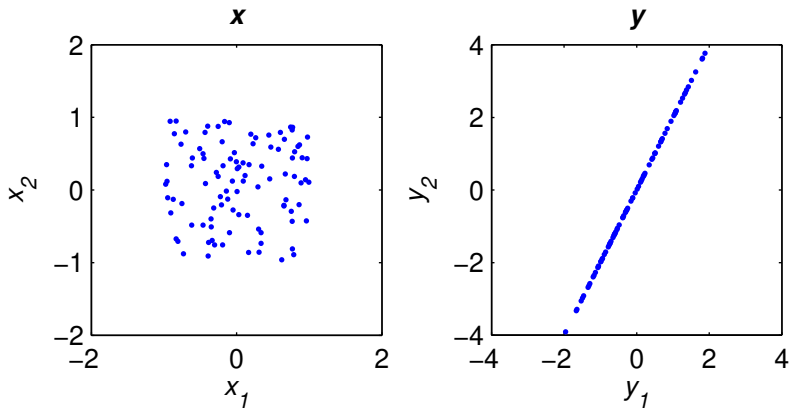


Figure 4.11: Mapping with a rank deficient matrix. A total of 100 random vectors are chosen for $\boldsymbol{x}$ such that x_1 and x_2 are iid and each is $\sim \mathcal{U}(-1, 1)$. Each is mapped to a corresponding $\boldsymbol{y}$.

- The number of independent rows of a matrix is also equal to its rank.
- The largest possible rank for a square matrix is equal to its size.

$$\text{rank}\, \boldsymbol{C} \leq n; \boldsymbol{C} \in \mathbb{R}^{n\times n}$$

- If $\text{rank}\, \boldsymbol{C} = n$ then the columns of $\boldsymbol{C}$ are linearly independent and we describe $\boldsymbol{C}$ as *full rank*.
- If $\text{rank}\, \boldsymbol{C} < n$ then the columns of $\boldsymbol{C}$ are linearly dependent and we describe $\boldsymbol{C}$ as *rank deficient*.

Matlab's `rank` command will reveal the rank of any matrix.

Consider a fat matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}, n < m$. $\boldsymbol{C}$ cannot have more than n independent columns, so $\text{rank}\, \boldsymbol{C} \leq n$. Similarly consider a thin matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}, n > m$. $\boldsymbol{C}$ cannot have more than m independent rows, so $\text{rank}\, \boldsymbol{C} \leq m$. Thus, for any matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}$, we have

$$\text{rank}\, \boldsymbol{C} \leq \min(n, m)$$

That is, the rank of a matrix can be no larger than the smallest of its dimensions. If the rank is equal to this number then we say it is full rank, otherwise it is rank deficient.

4.4 Matrix multiplication

We often need to cascade multiple mappings. That is we want to form $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$ and then use that to find $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{y}$.

$$\boldsymbol{z} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{x})$$

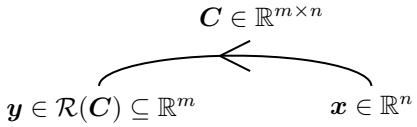


Figure 4.12: Mapping from an input vector $\boldsymbol{x}$ to an output vector $\boldsymbol{y}$ via the matrix $\boldsymbol{C}$.

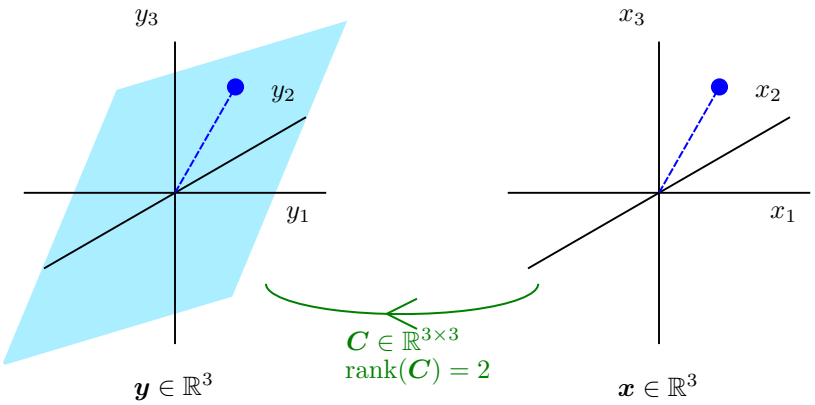


Figure 4.13: Mapping with a rank deficient matrix from $\mathbb{R}^{3\times 3}$ pushes a three dimensional input into another three output. However, because the matrix is only has rank two, all outputs live in a two dimensional subspace within the three dimensional output space.

Matrix multiplication allows us to compose a new matrix $\boldsymbol{C}$ so that $\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x}$, where $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$. This is matrix multiplication, where elements of $\boldsymbol{C}$ are given by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad \text{or, } c_{ij} = \boldsymbol{a}_i^{\mathsf{T}} \cdot \boldsymbol{b}_j$$

For example,

$$\begin{aligned}
\boldsymbol{A} \times \boldsymbol{B} &= \begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 3 & 4 & 0 \end{bmatrix} \times \begin{bmatrix} 3 & 3 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1\cdot 3 + 2\cdot 2 & 1\cdot 3 + 2\cdot 1 & 2\cdot 1 \\ 2\cdot 3 & 2\cdot 3 + 1\cdot 1 & 0 \\ 3\cdot 3 + 4\cdot 2 & 3\cdot 3 + 4\cdot 1 & 4\cdot 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 2 \\ 6 & 7 & 0 \\ 17 & 13 & 4 \end{bmatrix}
\end{aligned}$$

or,

$$\begin{aligned}
\boldsymbol{A} \times \boldsymbol{B} &= \begin{bmatrix} 3\boldsymbol{a}_1 + 2\boldsymbol{a}_2 & 3\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3 & \boldsymbol{a}_2 \end{bmatrix} \\
&= \begin{bmatrix} 3\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 2\begin{bmatrix} 2 \\ 0 \\ 4 \end{bmatrix} & 3\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 0 \\ 4 \end{bmatrix} + 1\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 2 \\ 0 \\ 4 \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} 7 & 5 & 2 \\ 6 & 7 & 0 \\ 17 & 13 & 4 \end{bmatrix}
\end{aligned}$$

Remember the following important properties of matrix multiplication,

- In general matrix multiplication does not commute; $\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}$
- You can only multiply two matrices if their sizes are compatible; That is, $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ only makes sense only for $\boldsymbol{A} \in \mathbb{K}^{p\times m}$ and $\boldsymbol{B} \in \mathbb{K}^{m\times n}$. Then the answer $\boldsymbol{C} \in \mathbb{K}^{p\times n}$.
- When composing multiple mappings, the matrix for the first mapping is placed furthest to the right.

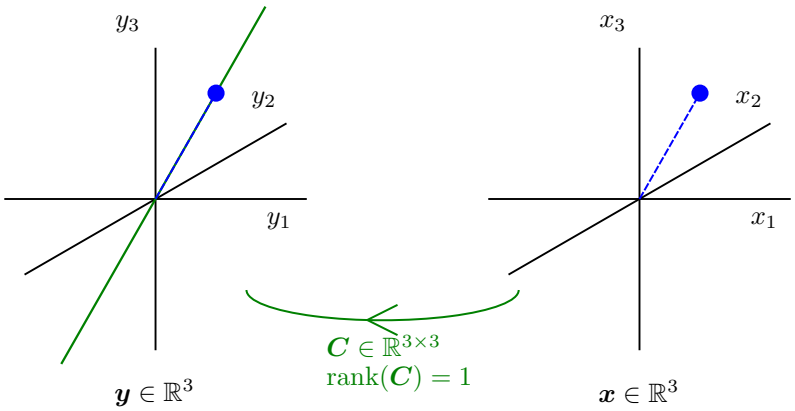


Figure 4.14: Mapping with a rank deficient matrix from $\mathbb{R}^{3\times 3}$ pushes a three dimensional input into another three output. However, because the matrix is only has rank two, all outputs live in a two dimensional subspace within the thre dimensional output space.

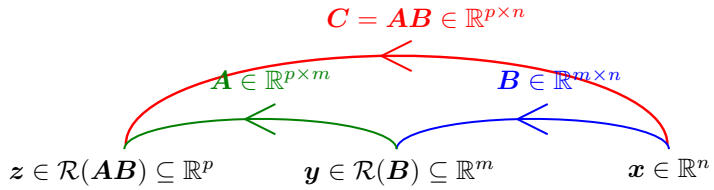


Figure 4.15: Matrix multiplication as a composite mapping by multiple matrices.

4.5 Matrix inversion

Given that a matrix performs a mapping from one vector to another, it would seem reasonable to ask whether it is possible to reverse the mapping. That is, if $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$, can we construct an inverse matrix (denoted $\boldsymbol{C}^{-1}$) such that $\boldsymbol{x} = \boldsymbol{C}^{-1}\boldsymbol{y}$? Essentially, only matrices that do not change the dimension of a

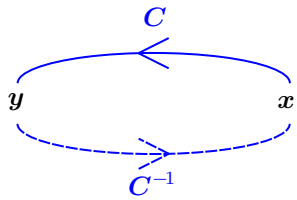


Figure 4.16: Mapping from an input vector $\boldsymbol{x}$ to an output vector $\boldsymbol{y}$ via the matrix $\boldsymbol{C}$ and the reverse mapping via $\boldsymbol{C}^{-1}$.

vector when used as a mapping have an inverse.

- Non-square matrices do not have inverses.
- Rank deficient matrices do not have inverses.

If the inverse exists than a matrix is called *invertible* or *non-singular*.

From figure 4.16 it should be evident that doing a mapping with matrix $\boldsymbol{C}$ and then following with a mapping by $\boldsymbol{C}^{-1}$ is the same as doing nothing. That is, $\boldsymbol{C}^{-1}\boldsymbol{C}\boldsymbol{x} = \boldsymbol{x}$. That is, mapping in this way is equivalent to mapping with the identity matrix, $\boldsymbol{I}$, that leaves the vector undisturbed.

Thus we can write

$$\boldsymbol{C}\boldsymbol{C}^{-1} = \boldsymbol{C}^{-1}\boldsymbol{C} = \boldsymbol{I}$$

We will mostly perform matrix inversion numerically. However, we will need to remember that

$$\boldsymbol{C}^{-1} = \frac{\text{adj}\,\boldsymbol{C}}{\det \boldsymbol{C}}$$

where $\operatorname{adj} \boldsymbol{C}$ is the adjugate of $\boldsymbol{C}$ and $\det \boldsymbol{C}$ is its determinant. Though we will not find much use for direct calculation of the adjugate, we will make extensive use of the determinant. There are a number of common methods for calculating determinants. Mostly they differ in the method used to keep track of the signs of the various products that must be combined. Make sure that you revise whichever method you know. Note in particular that $\det \boldsymbol{C} = 0$ when $\boldsymbol{C}$ is not invertible.

Inversion of 2 by 2 matrices is sufficiently frequent that it will help to remember the equation.

$$\text{If } \boldsymbol{C} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \text{ then } \boldsymbol{C}^{-1} = \frac{\operatorname{adj} \boldsymbol{C}}{\det \boldsymbol{C}}$$
$$= \frac{1}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$
$$= \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

If you take a region of space and push it through a mapping described by a matrix $\boldsymbol{A}$, then the size (area, volume, etc) of the mapped region will be multiplied by $\det \boldsymbol{A}$. Remembering this property will help you remember that $\det(\boldsymbol{AB}) = \det \boldsymbol{A} \det \boldsymbol{B}$. This simply says that the expansion or contraction in volume is the same whether it is done in one single mapping or a series of mappings. Remember that non-invertible matrices have $\det \boldsymbol{A} = 0$. You might like to think of what that would mean in the context of the earlier discussion of spanned spaces. Exercise: Look back at the fundamental mappings earlier in these notes and see whether the effect of the mapping on the area of the regions is consistent with the determinant of the corresponding mapping matrices.

Matrix multiplication allows us to express a cascade of matrix multiplications. We would also like to be able to undo such a cascade. Consider a matrix $\boldsymbol{C} = \boldsymbol{AB}$. To undo a mapping with $\boldsymbol{C}$ we would first need to undo the effect of $\boldsymbol{A}$ and then undo the effect of $\boldsymbol{B}$. That is,

$$\boldsymbol{C}^{-1} = (\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$$

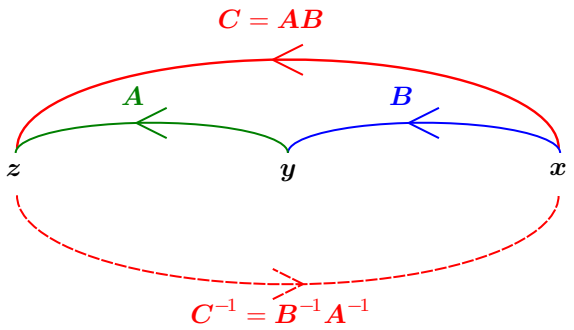


Figure 4.17: Inverse of a composite mapping.

4.6 Eigenanalysis

Consider again a mapping with some $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$. For many choices of input vector $\boldsymbol{x}$, the output vector $\boldsymbol{y}$ will be rotated relative to $\boldsymbol{x}$. However, there will be some input vectors where the output vectors are not rotated; they will just be scaled versions of the input vectors. This can be seen for the red and blue vectors in figure 4.18.

The "special" input vectors that are not rotated under transformation by $\boldsymbol{A}$ are called *eigenvectors*. Each different $\boldsymbol{A}$ matrix has its own set of eigenvectors. The scaling experienced *by an eigenvector* after the operation of $\boldsymbol{A}$ is its associated *eigenvalue*. Thus, each eigenvector has its own associated eigenvalue. Note that different eigenvectors can have the same eigenvalue. That just means that the scaling is the same for two different input vectors.

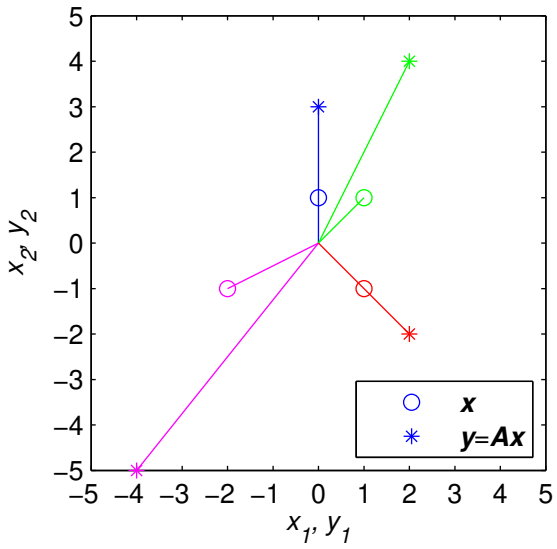


Figure 4.18: Rotation of points about the points when undergoing mapping by a matrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$.

If we want to find the eigenvectors, we can write the above in the standard form:

$$\boxed{\lambda \boldsymbol{v} = \boldsymbol{A}\boldsymbol{v}}$$

Where $\boldsymbol{v}$ is an eigenvector and $\lambda \in \mathbb{C}$ is its associated eigenvalue. We can rearrange this to solve for $\boldsymbol{v}$:

$$(\lambda \boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{v} = 0$$

This equation only has a non-trivial solution if $\boldsymbol{A} - \lambda \boldsymbol{I}$ is singular. This happens when the determinant of $\boldsymbol{A} - \lambda \boldsymbol{I}$ is zero. Thus to find the eigenvectors we must solve

$$\left|\lambda \boldsymbol{I} - \boldsymbol{A}\right| = 0.$$

Let's calculate the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}$ used in the figures 4.18 and 4.19 above.

$$\begin{aligned} \left|\lambda \boldsymbol{I} - \boldsymbol{A}\right| &= 0 \\ \det\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}\right) &= 0 \\ \begin{vmatrix} \lambda - 2 & 0 \\ -1 & \lambda - 3 \end{vmatrix} &= 0 \\ (\lambda - 2)(\lambda - 3) &= 0 \\ \implies \lambda &= 2, 3 \end{aligned}$$

This example is particularly simple – in general you will need to solve a polynomial equation to find λ_i.

To find the eigenvectors we substitute each eigenvalue λ_i into $\lambda \boldsymbol{v} = \boldsymbol{A}\boldsymbol{v}$ in turn and solve to find the

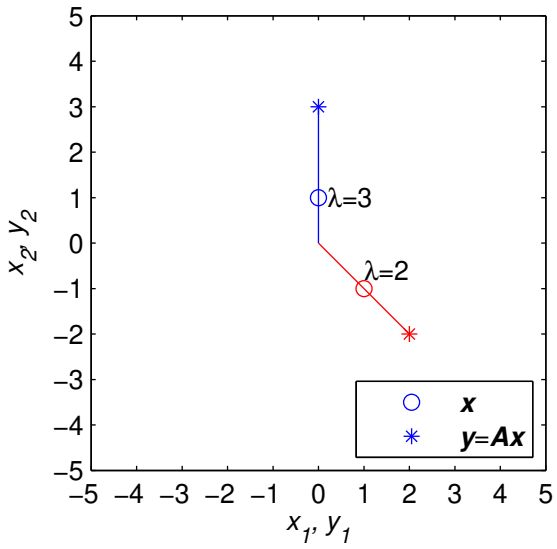


Figure 4.19: Eigenvectors of the matrix $\boldsymbol{A}$ with associated eigenvalues shown.

associated eigenvector $\boldsymbol{v}$.

$$
\begin{aligned}
2\boldsymbol{v} &= \boldsymbol{A}\boldsymbol{v} \\
2\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\
\begin{bmatrix} 2v_1 \\ 2v_2 \end{bmatrix} &= \begin{bmatrix} 2v_1 \\ v_1 + 3v_2 \end{bmatrix} \\
\implies 2v_2 &= v_1 + 3v_2 \\
v_2 &= -v_1 \\
\boldsymbol{v} &= \begin{bmatrix} v_1 \\ -v_1 \end{bmatrix}
\end{aligned}
$$

We often choose to set $\|\boldsymbol{v}\| = 1$, so, $\boldsymbol{v} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$.

Similarly,

$$
\begin{aligned}
3\boldsymbol{v} &= \boldsymbol{A}\boldsymbol{v} \\
3\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\
\begin{bmatrix} 3v_1 \\ 3v_2 \end{bmatrix} &= \begin{bmatrix} 2v_1 \\ v_1 + 3v_2 \end{bmatrix} \\
\implies 3v_2 &= v_1 + 3v_2 \\
v_1 &= 0 \\
\boldsymbol{v} &= \begin{bmatrix} 0 \\ v_2 \end{bmatrix}
\end{aligned}
$$

Choosing $\|\boldsymbol{v}\| = 1$ we get $\boldsymbol{v} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

So, we have an eigenvector $\boldsymbol{v}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\mathsf{T}$ with associated eigenvalue $\lambda_1 = 3$ and a second eigenvector $\boldsymbol{v}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^\mathsf{T}$ with associated eigenvalue $\lambda_2 = 2$. This is consistent with our plot of the mapping produced by $\boldsymbol{A}$ as seen in figure 4.19.

4.6.1 Complex eigenvalues and eigenvectors

When solving the characteristic equation of a matrix, you might find that you get complex eigenvalues. These will always come in complex conjugate pairs and will result in eigenvectors that are also complex conjugate pairs. In the interests of brevity we won't go into the details of this situation here. Suffice it to say that when we have such a situation, we find that repeated mappings by the matrix cause both rotation and stretching that is confined to a special *plane*.

4.7 The Cayley-Hamilton theorem

Theorem 3. *Every matrix satisfies its own characteristic equation.*

This is useful for calculating matrix powers. For example

$$\begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix} \text{ has characteristic equation } \left|\lambda \boldsymbol{I} - \boldsymbol{A}\right| = 0$$

$$\implies (\lambda - 2)(\lambda - 3) = 0$$

$$\lambda^2 - 5\lambda + 6 = 0$$

So, the CH theorem says that $\boldsymbol{A}^2 - 5\boldsymbol{A} + 6\boldsymbol{I} = 0$

$$\boldsymbol{A}^2 = 5\boldsymbol{A} - 6\boldsymbol{I}$$

$$= \begin{bmatrix} 10 & 0 \\ 5 & 15 \end{bmatrix} - \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}$$

$$\boldsymbol{A}^2 = \begin{bmatrix} 4 & 0 \\ 5 & 9 \end{bmatrix}$$

We can also use the Cayley-Hamilton theorem to find matrix inverses.Consider a matrix $\boldsymbol{A}$ with an associated characteristic polynomial

$$\lambda^n + a_{n-1}\lambda^{n-1} + a_{n-2}\lambda^{n-2} + \cdots + a_0 = 0$$

$$\implies \boldsymbol{A}^n + a_{n-1}\boldsymbol{A}^{n-1} + a_{n-2}\boldsymbol{A}^{n-2} + \cdots + a_1\lambda + a_0\boldsymbol{I} = 0$$

$$\boldsymbol{A}^n + a_{n-1}\boldsymbol{A}^{n-1} + a_{n-2}\boldsymbol{A}^{n-2} + \cdots + a_1\boldsymbol{A} = -a_0\boldsymbol{I}$$

$$\boldsymbol{A}\left(\boldsymbol{A}^{n-1} + a_{n-1}\boldsymbol{A}^{n-2} + a_{n-2}\boldsymbol{A}^{n-3} + \cdots + a_1\boldsymbol{I}\right) = -a_0\boldsymbol{I}$$

$$\boldsymbol{A}\left(-\frac{1}{a_0}\left(\boldsymbol{A}^{n-1} + a_{n-1}\boldsymbol{A}^{n-2} + a_{n-2}\boldsymbol{A}^{n-3} + \cdots + a_1\boldsymbol{I}\right)\right) = \boldsymbol{I}$$

But since $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$,the bracketed expression must be $\boldsymbol{A}^{-1}$.

$$\text{That is, } \boldsymbol{A}^{-1} = \left(-\frac{1}{a_0}\left(\boldsymbol{A}^{n-1} + a_{n-1}\boldsymbol{A}^{n-2} + a_{n-2}\boldsymbol{A}^{n-3} + \cdots + a_1\boldsymbol{I}\right)\right)$$

This is a useful expression, as it allows us to find a matrix inverse as a polynomial function of $\boldsymbol{A}$, which is usually easier than finding the inverse as the ratio of the adjugate and determinant.

For our previous example we have

$$\begin{aligned}
\boldsymbol{A}^2 - 5\boldsymbol{A} + 6\boldsymbol{I} &= 0 \\
\boldsymbol{A}^2 - 5\boldsymbol{A} &= -6\boldsymbol{I} \\
\boldsymbol{A}\left(\boldsymbol{A} - 5\boldsymbol{I}\right) &= -6\boldsymbol{I} \\
\boldsymbol{A}\left(-\frac{1}{6}(\boldsymbol{A} - 5\boldsymbol{I})\right) &= \boldsymbol{I} \\
\text{But,} \quad \boldsymbol{A}\boldsymbol{A}^{-1} &= \boldsymbol{I} \\
\implies \boldsymbol{A}^{-1} &= -\frac{1}{6}(\boldsymbol{A} - 5\boldsymbol{I}) \\
&= -\frac{1}{6}\left(\begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}\right) \\
&= \begin{bmatrix} \frac{1}{2} & 0 \\ -\frac{1}{6} & \frac{1}{3} \end{bmatrix}
\end{aligned}$$

Perhaps a more straightforward method for doing this calculation is as follows.

$$\begin{aligned}
\boldsymbol{A}^2 - 5\boldsymbol{A} + 6\boldsymbol{I} &= 0 \\
\boldsymbol{A}^2 - 5\boldsymbol{A} &= -6\boldsymbol{I}
\end{aligned}$$

Now post-multiply by $\boldsymbol{A}^{-1}$ throughout

$$\begin{aligned}
\boldsymbol{A}^2\boldsymbol{A}^{-1} - 5\boldsymbol{A}\boldsymbol{A}^{-1} &= -6\boldsymbol{I}\boldsymbol{A}^{-1} \\
\boldsymbol{A} - 5\boldsymbol{I} &= -6\boldsymbol{A}^{-1}
\end{aligned}$$

We now have an expression for the inverse

$$\begin{aligned}
\implies \boldsymbol{A}^{-1} &= -\frac{1}{6}(\boldsymbol{A} - 5\boldsymbol{I}) \\
&= -\frac{1}{6}\left(\begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}\right) \\
&= \begin{bmatrix} \frac{1}{2} & 0 \\ -\frac{1}{6} & \frac{1}{3} \end{bmatrix}
\end{aligned}$$

as before.

Those tricks might prove useful in the course, however, the most common use we will make of the Cayley-Hamilton theorem arises indirectly.

Corollary 4. *Every $n \times n$ matrix can be expressed as a polynomial of that matrix having at most degree $n - 1$.*

That is, if $\boldsymbol{A}$ is an $n \times n$ matrix, then we can express it as a linear combination of matrices up to the $(n-1)^{\text{th}}$ power of $\boldsymbol{A}$. We will find this useful several times, as it will allow us to terminate otherwise seemingly infinite series of matrix powers.

5 State Space Modelling

5.1 Introduction

State space models provide an alternative representation of a system to that provided by transfer functions.

- State space models more easily deal with multiple inputs and outputs.
- State space models have good numerical properties. This is particularly useful when dealing with large (complex) systems.
- Continuous time and discrete time state space models are very similar. We therefore can use one mathematical formalism for both problem domains.

A state variable describes a piece of information about a system that can *change* as the system evolves. Moreover, it is a piece of information that we *must* know to be able describe how the system will evolve. The state vector is the various state variables gathered together into a vector. The state vector is almost universally denoted as $\boldsymbol{x}$ in control. For example, a ball at position x_1, moving autonomously in one dimension with velocity $\dot{x}_1$ is specified completely by the state vector $[x_1, \dot{x}_1]^\mathsf{T}$, where x_1 and $\dot{x}_1$ are the state variables.

As another example, consider the inverted pendulum balanced on a cart, as shown in figure 5.1. The cart is free to move horizontally, with its position specified by x_1. The angle of the pendulum θ can also change as the system moves. As a result both x_1 and θ will be state variables.

However, knowing x_1 and θ alone is not sufficient to predict the future of the system though. If someone told you what the positions were you would also need to know the rate of change of the two variables. That is, you would also need to know $\dot{\boldsymbol{x}}_1$ and $\dot{\theta}$. Thus, for the inverted pendulum example, the state vector is $\boldsymbol{x} = \begin{bmatrix} x_1 & \dot{x}_1 & \theta & \dot{\theta} \end{bmatrix}^\mathsf{T}$.

Notice that the mass of the cart or the moment of inertia of the pendulum arm are not state variables. We do need to know those quantities to determine how the system evolves, but we can assume that they do not change. This is not to say that mass could never be a state variable. Consider the case of a rocket, which changes ts mass as its fuel is consumed.

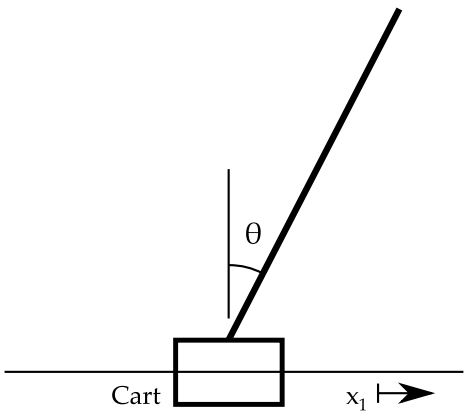


Figure 5.1: An inverted pendulum mounted on a cart.

5.1.1 From DEs to State Space

Many systems can be described by a system of differential equations of form

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big)$$

where $\boldsymbol{x}(t)$ is the state vector, $\boldsymbol{u}(t)$ is a vector of inputs and $\boldsymbol{f}$ is a function that describes how the various states and inputs influence the rate of change in the state. The time derivative of $\boldsymbol{x}$ is a just a

vector of the derivatives of each state variable,

$$\dot{\boldsymbol{x}} := \begin{bmatrix} \dot{x}_1 & \dot{x}_2 & \dots & \dot{x}_n \end{bmatrix}^{\mathsf{T}} \quad .$$

In the general case $\boldsymbol{f}$ can be non-linear and vary with time. However, if we restrict ourselves to systems that can be modelled as time invariant then we can make a slight simplification to the state equation.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$$

Many control systems can be adequately described by linear equations, so we will concentrate on LTI systems for most of this course. We will assume that the evolution of each state variable depends only on the current state and any external inputs. That is, in a system having n state variables and m inputs, we model the evolution of a state variable x_i with an equation

$$\begin{aligned} \dot{x}_i = \frac{\mathrm{d}x_i}{\mathrm{d}t} &= a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \\ &\quad + b_{i1}u_1 + b_{i2}u_2 + \ldots + b_{im}u_m \end{aligned}$$

where a_{ij} and b_{ij} are real, constant coefficients.

We have a similar equation for each of our state variables. Thus we have a system of coupled differential equations:

$$\begin{aligned} \dot{x}_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ &\quad + b_{11}u_1 + b_{12}u_2 + \ldots + b_{1m}u_m \\ \dot{x}_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ &\quad + b_{21}u_1 + b_{22}u_2 + \ldots + b_{2m}u_m \\ &\vdots \\ \dot{x}_n &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \\ &\quad + b_{n1}u_1 + b_{n2}u_2 + \ldots + b_{nm}u_m \end{aligned}$$

We can write this system of equations more efficiently as

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

where $\boldsymbol{x}$ is a vector of state variables, $\boldsymbol{u}$ is a vector of inputs and $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant matrices.

For a system with n state variables and m inputs

$$\begin{aligned} \boldsymbol{x} \text{ and } \dot{\boldsymbol{x}} \text{ are } n \times 1 \text{ vectors } &(\boldsymbol{x}, \dot{\boldsymbol{x}} \in \mathbb{R}^n) \\ \boldsymbol{u} \text{ is an } m \times 1 \text{ vector } &(\boldsymbol{u} \in \mathbb{R}^m) \\ \boldsymbol{A} \text{ is an } n \times n \text{ matrix } &\big(\boldsymbol{A} \in \mathbb{R}^{n \times n}\big) \\ \boldsymbol{B} \text{ is an } n \times m \text{ matrix } &\big(\boldsymbol{B} \in \mathbb{R}^{n \times m}\big) \end{aligned}$$

We will see later that it is occasionally convenient to allow the elements of $\boldsymbol{A}$ to be complex, but we normally consider only real $\boldsymbol{A}$.

Figure 5.2 is a graphical representation of the equation describing the relationship between $\dot{\boldsymbol{x}}$ and the current state $\boldsymbol{x}$ and the inputs $\boldsymbol{u}$.

5.1.2 Reduction to Systems of First Order DEs

Notice that the state space form only contains first derivatives. However, we can use it to describe systems that are governed by differential equations including higher derivatives, as we can express higher order differential equations as systems of first order equations. For example, consider a second order DE:

$$\ddot{r} = C_1\dot{r} + C_2 r + C_3 u$$

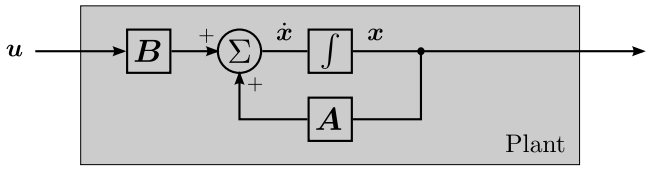


Figure 5.2: Graphical representation of the equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$

We can define two state variables x_1 and x_2 such that $x_1 \equiv r$ and $x_2 \equiv \dot{r}$. We then arrive at an alternative representation of our DE:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= C_1 x_2 + C_2 x_1 + C_3 u \\ \text{or, } \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ C_2 & C_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ C_3 \end{bmatrix} u \end{aligned}$$

We can extend this argument to deal with any n^{th}-order LTI DEs.

As an example, we wish to form a state space model for the mass spring system shown schematically in figure 5.3. First write a differential equation describing the motion of the mass: $m\ddot{x} = \sum F_{\text{ext}} =$

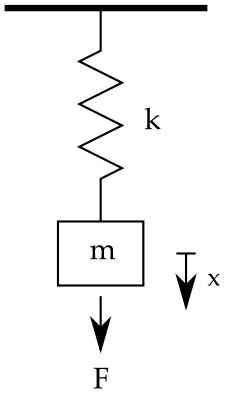


Figure 5.3: A mass spring system with an external applied force F.

$F - kx$
Define the state variables and input:

$$\text{Let } x_1 \equiv x, x_2 \equiv \dot{x} \text{ and } u \equiv F$$

Rewrite the DE in terms of the state variables:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ m\dot{x}_2 &= u - kx_1 \\ \implies \dot{x}_2 &= \frac{1}{m}u - \frac{k}{m}x_1 \end{aligned}$$

Rewrite in a state space form:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

5.1.3 Exercise: Add a damper to the previous example

We can add a damper to the system and see how that changes the state space model. Figure 5.4 shows a schematic of the modified system.

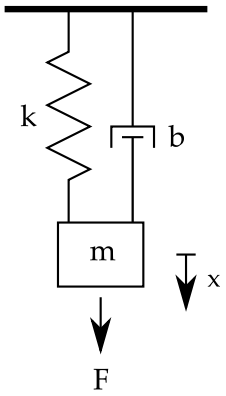


Figure 5.4: A mass spring damper system with an external applied force F.

$$m\ddot{x} = F - kx - b\dot{x}$$

Let $x_1 \equiv x$ and $x_2 \equiv \dot{x}$ be the state variables and $u \equiv F$ be the input.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ m\dot{x}_2 &= u - kx_1 - bx_2 \\ \implies \dot{x}_2 &= \frac{u}{m} - \frac{k}{m}x_1 - \frac{b}{m}x_2 \end{aligned}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

5.1.4 An example that begins with multiple DEs

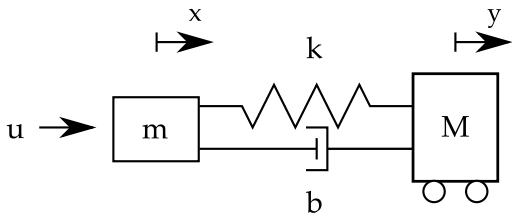


Figure 5.5: Two carts separated with a spring damper system.

We can also consider more complex examples. For example, the two mass system showed in figure 5.5 is described by a pair of coupled second order differential equations.

$$\begin{aligned} m\ddot{x} &= +k(y - x) + b(\dot{y} - \dot{x}) + u \\ M\ddot{y} &= -k(y - x) - b(\dot{y} - \dot{x}) \end{aligned}$$

Let the state vector be $[x_1, x_2, x_3, x_4]^\mathsf{T} \equiv [x, \dot{x}, y, \dot{y}]^\mathsf{T}$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{k}{m}x_3 + \frac{b}{m}x_4 + \frac{1}{m}u \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k}{M}x_1 + \frac{b}{M}x_2 - \frac{k}{M}x_3 - \frac{b}{M}x_4 \end{aligned}$$

Writing this in the conventional form we get $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ with

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m} & -\frac{b}{m} & \frac{k}{m} & \frac{b}{m} \\ 0 & 0 & 0 & 1 \\ \frac{k}{M} & \frac{b}{M} & -\frac{k}{M} & -\frac{b}{M} \end{bmatrix} \text{ and } \boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \\ 0 \\ 0 \end{bmatrix}$$

5.1.5 Evolution of the State Vector

The state of the system can be thought of as being described by a point in an abstract, higher dimensional *state space*. This space, has dimension equal to the number of state variables needed to describe the total system.

At each moment the system has a particular value for each of its state variables. The vector of these values defines the systems location in the state space.

The equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$ then describes how the state vector evolves. $\boldsymbol{A}$ describes the system's natural trajectory in state space and $\boldsymbol{B}$ describes how it is affected by external influences.

If we know the state transition matrix $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$ then we can predict the future (and past) values of $\boldsymbol{x}$ for a given input $\boldsymbol{u}$.

In two dimensions we can draw a map of the path taken by the solution of a DE. This is known as a phase portrait. (State space is almost exactly the same as the physics notion of *phase space*.) A simple pendulum of length L that is undergoing small, unforced, oscillations can be shown to have a state matrix $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix}$. We can plot how the state evolves by starting at an arbitrary point in state space and then updating the state appropriately. The resulting figure is shown in figure 5.6. Notice that the trajectory taken by the system is everywhere tangent to the direction of $\dot{\boldsymbol{x}}$. Consequently the phase portrait allows us to visualise the trajectory that will be taken by a system.

5.2 Outputs in State Space

Control systems usually have one or more quantities that can be identified as *outputs*. These might be things that we can measure, or things that we want to control. Given that our state vector, $\boldsymbol{x}$, encapsulates all information about the system, we expect that our output vector, $\boldsymbol{y}$, is a function of the current state variables. In the general case the outputs may be a function of the current inputs, $\boldsymbol{u}$, as well. Thus, $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$. In the (common) case where the outputs are linear functions of state and input then

$$\begin{aligned} y_i = {} & c_{i1}x_1 + c_{i2}x_2 + \ldots + c_{in}x_n \\ & + d_{i1}u_1 + d_{i2}u_2 + \ldots + d_{im}u_m \end{aligned}$$

for $c_{ij}, d_{ij} \in \mathbb{R}$.

Note that we often find that $\boldsymbol{y}$ has no direct dependence on $\boldsymbol{u}$ so we have $d_{ij} = 0 \quad \forall(i,j)$.

Rewriting these equations as we did for the state equations above, we get the matrix equation

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

If we have p outputs from our system then $\boldsymbol{C}$ is an $p \times n$ matrix ($\boldsymbol{C} \in \mathbb{R}^{p \times n}$) and $\boldsymbol{D}$ is an $p \times m$ matrix ($\boldsymbol{D} \in \mathbb{R}^{p \times m}$). So, returning to the previous two cart example;

- If the position of M were the output, then we have $y = x_3$.
  We could thus write $y = \boldsymbol{C}\boldsymbol{x}$ with $\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$.

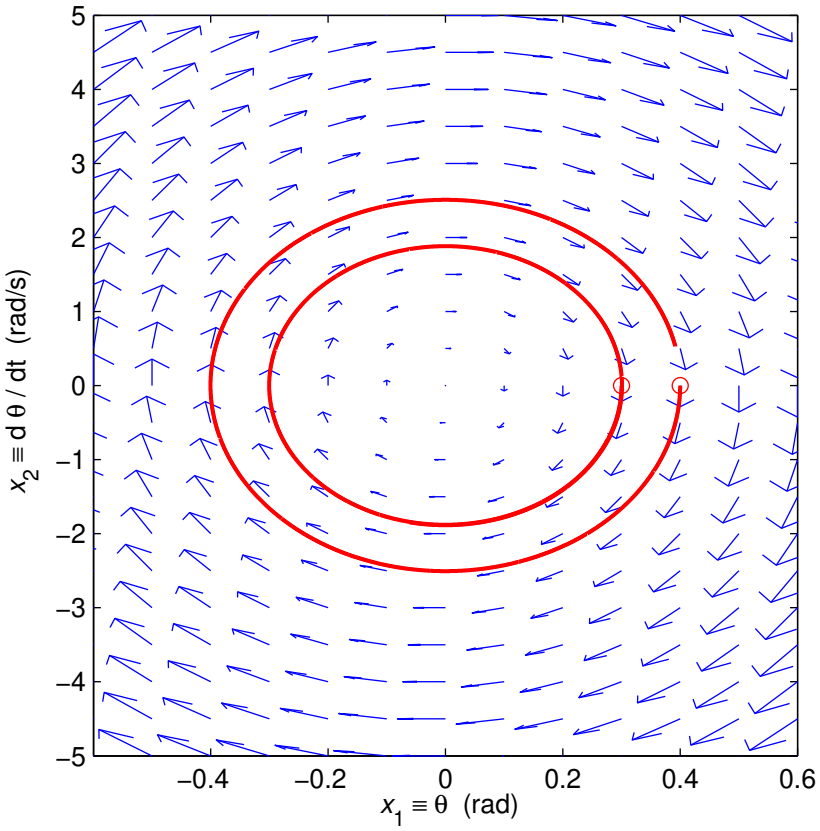


Figure 5.6: Phase portrait for a planar pendulum undergoing small oscillations. The blue arrows show the local direction and magnitude of $\dot{\boldsymbol{x}}$. The red tracks indicate the path in state space that will be taken by two different systems that begin at the red circles.

- If we were to measure both the position and velocity of mass m instead, we would have $y_1 = x_1$ and $y_2 = x_2$, so we would have

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Each row of the $\boldsymbol{C}$ matrix can be interpreted as corresponding to a separate physical sensor. The matrix entries are the gains of the sensors, which map from whatever units we are using to represent the state into the measured units. Rows of the $\boldsymbol{C}$ matrix can have more than one entry. For example, in our two mass example we could have a sensor that measures the distance between the two masses $(y - x)$. This would be represented as a row of $\boldsymbol{C}$ equal to $\begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}$. If we were to place an accelerometer on mass m, then we would have $\boldsymbol{C} = \begin{bmatrix} -\frac{k}{m} & -\frac{b}{m} & \frac{k}{m} & \frac{b}{m} \end{bmatrix}$ and $\boldsymbol{D} = \begin{bmatrix} \frac{1}{m} \end{bmatrix}$. We will later see state space representations where the states do not have clean physical meaning. For such systems the $\boldsymbol{C}$ matrix can be messier, but it still does the job of mapping the internal state representation into real measurements.

We will also later see that the outputs do not need to correspond to physical measurement. On occasion we will care about quantities that are linear combinations of the state, but we will not measure them directly. We should therefore not get *too* enthusiastic about regarding outputs as corresponding to physical measurements, as we will use exactly the same formalism to model these "extra" outputs. However the measurement picture is a useful one to build intuition.

We can extend our graphical representation of the system so that it also generates the outputs as shown in figure 5.7.

We now describe the complete state space model with a pair of equations,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

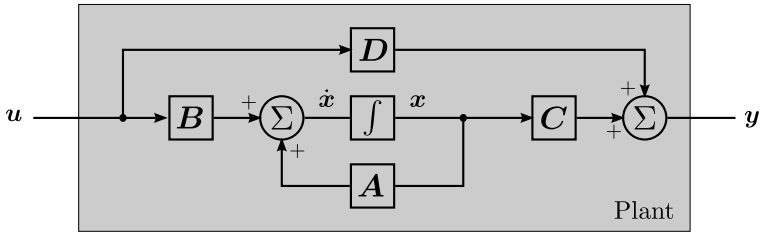


Figure 5.7: Graphical representation of the complete state space model for a system.

where $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{y}$ are each functions of time.

5.3 Examples of State Space models for Electronic Circuits

5.3.1 A passive circuit

Consider a state space model for the circuit shown in 5.8. We can sum the currents into the two nodes having voltages v_c and v_o to find two coupled differential equations that describe the circuit operation.

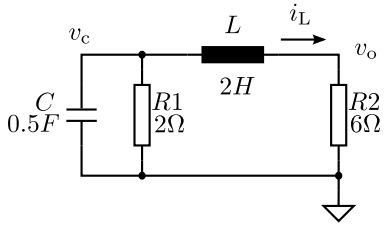


Figure 5.8: A passive circuit with no input.The output of this circuit is the node v_o.

$$\begin{aligned} C\frac{\mathrm{d}v_\mathrm{c}}{\mathrm{d}t} + \frac{v_\mathrm{c}}{R_1} + i_\mathrm{L} &= 0 \\ -i_\mathrm{L} + \frac{v_\mathrm{o}}{R_2} &= 0 \end{aligned} \tag{5.1}$$

But $v_\mathrm{o} = v_\mathrm{c} - L\dfrac{\mathrm{d}i_\mathrm{L}}{\mathrm{d}t}$, so we can substitute into (5.1) to find

$$-i_\mathrm{L} + \frac{v_\mathrm{c}}{R_2} - \frac{L}{R_2}\frac{\mathrm{d}i_\mathrm{L}}{\mathrm{d}t} = 0$$

We now choose our state variables to be $x_1 = v_c$ and $x_2 = i_L$. Rewriting the DEs we get

$$\begin{aligned} \dot{x}_1 &= -\frac{1}{R_1 C}x_1 - \frac{1}{C}x_2 \\ \dot{x}_2 &= -\frac{1}{L}x_1 - \frac{R_2}{L}x_2 \end{aligned}$$

and thus

$$\dot{x} = \begin{bmatrix} -\dfrac{1}{R_1 C} & -\dfrac{1}{C} \\ -\dfrac{1}{L} & -\dfrac{R_2}{L} \end{bmatrix} x$$

We also know that $v_o = i_L R_2$, thus if we set the output $y = v_o$, we can write $y = R_2 x_2$. Thus we have $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$, $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$ with

$$\boldsymbol{A} = \begin{bmatrix} -\dfrac{1}{R_1 C} & -\dfrac{1}{C} \\ -\dfrac{1}{L} & -\dfrac{R_2}{L} \end{bmatrix}$$
$$\boldsymbol{C} = \begin{bmatrix} 0 & R_2 \end{bmatrix}$$

Note that because we have no input to this circuit we have $\boldsymbol{B} = \boldsymbol{D} = 0$ in this case.

5.3.2 A circuit with multiple inputs

Let's consider a second example, this time having multiple inputs. An opamp circuit that sums two inputs and filters the result with a first order, low pass, filter is shown in figure 5.9.

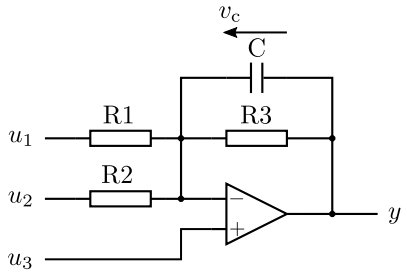


Figure 5.9: An opamp circuit with multiple inputs.

Summing the currents flowing into the inverting terminal we can write a single DE to describe the system.

$$0 = \frac{u_1}{R_1} + \frac{u_2}{R_2} - \frac{v_c}{R_3} - C\frac{\mathrm{d}v_c}{\mathrm{d}t}$$
$$y + v_c = u_3 \qquad \text{(Virtual short)}$$

We choose our state variable $x_1 = v_c$.

$$\implies \dot{x}_1 = \frac{1}{C}\left(-\frac{x_1}{R_3} + \frac{u_1}{R_1} + \frac{u_2}{R_2}\right)$$
$$= -\frac{x_1}{R_3 C} + \frac{u_1}{R_1 C} + \frac{u_2}{R_2 C}$$
$$y = u_3 - x_1$$

Thus we have

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$
$$y = \boldsymbol{C}x + \boldsymbol{D}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

with $\boldsymbol{A} = \begin{bmatrix} -\dfrac{1}{R_3 C} \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} \dfrac{1}{R_1 C} & \dfrac{1}{R_2 C} & 0 \end{bmatrix}$, $\boldsymbol{C} = \begin{bmatrix} -1 \end{bmatrix}$ and $\boldsymbol{D} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$.

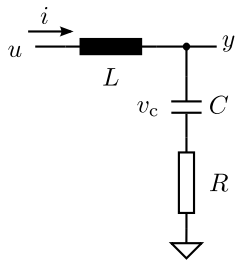


Figure 5.10: An RLC circuit.

5.3.3 An RLC circuit

Consider the RLC filter shown in figure 5.10. We can write and rearrange differential equations describing its behaviour.

$$L\frac{\mathrm{d}i}{\mathrm{d}t} + v_\mathrm{c} + Ri = u \implies \frac{\mathrm{d}i}{\mathrm{d}t} = \frac{u - Ri - v_\mathrm{c}}{L}$$
$$i = C\frac{\mathrm{d}V_\mathrm{c}}{\mathrm{d}t} \implies \dot{v}_\mathrm{c} = \frac{i}{C}$$

Let's define our state vector as $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\mathsf{T} = \begin{bmatrix} i & v_\mathrm{c} \end{bmatrix}^\mathsf{T}$ and rewrite our circuit equations in terms of the state variables:

$$\dot{x}_1 = -\frac{R}{L}x_1 - \frac{1}{L}x_2 + \frac{1}{L}u$$
$$\dot{x}_2 = \frac{1}{C}x_1$$

In matrix terms,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} u$$

Also, we have

$$\begin{aligned} y &= v_c + iR \\ &= x_2 + x_1 R \\ &= \begin{bmatrix} R & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

Thus, our system is described by the matrices

$$\boldsymbol{A} = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} R & 1 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix}.$$

5.3.4 Summary of Continuous Time Systems

An LTI dynamical system with n states, m inputs and p outputs can be described by the state space model:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}\ ,\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

where

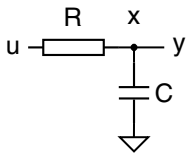


Figure 5.11: An RC low pass filter.

$\boldsymbol{x}$ is the state vector and is an n element vector,	$\boldsymbol{x}, \dot{\boldsymbol{x}} \in \mathbb{R}^n$
$\boldsymbol{u}$ is the input vector and is an m element vector,	$\boldsymbol{u} \in \mathbb{R}^m$
$\boldsymbol{y}$ is the output vector and is a p element vector,	$\boldsymbol{u} \in \mathbb{R}^p$
$\boldsymbol{A}$ is an $n \times n$ matrix,	$\boldsymbol{A} \in \mathbb{R}^{n \times n}$
$\boldsymbol{B}$ is an $n \times m$ matrix,	$\boldsymbol{B} \in \mathbb{R}^{n \times m}$
$\boldsymbol{C}$ is a $p \times n$ matrix,	$\boldsymbol{C} \in \mathbb{R}^{p \times n}$
$\boldsymbol{D}$ is a $p \times m$ matrix,	$\boldsymbol{D} \in \mathbb{R}^{p \times m}$

5.4 Manipulation of State Space Models

We will frequently need to combine state space models for different systems. The most common situations in which this will be necessary are when you consider adding the dynamics of a plant with those of its actuators or sensors. In each case the principle will be to produce an augmented state vector which is a combination of the states of the two subsystems. We will then add the appropriate coupling terms to the augmented system matrices.

Consider the two state space models S_1 and S_2 having matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$ for $i \in \{1, 2\}$. We have $\dot{\boldsymbol{x}}_i = \boldsymbol{A}_i\boldsymbol{x}_i + \boldsymbol{B}_i\boldsymbol{u}_i; \boldsymbol{y}_i = \boldsymbol{C}_i\boldsymbol{x}_i$. Let's form a new system that has the two systems as constituent parts. For now we will ignore any coupling between the two systems. We will use an augmented state matrix that is a combination of the state variables of the two subsystems; $\boldsymbol{x} := \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2\end{bmatrix}^\mathsf{T}$.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{bmatrix} &= \begin{bmatrix}\boldsymbol{A}_1 & 0 \\ 0 & \boldsymbol{A}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}_1 & 0 \\ 0 & \boldsymbol{B}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_1 \\ \boldsymbol{u}_2\end{bmatrix} \\
\dot{\boldsymbol{x}} &= \begin{bmatrix}\boldsymbol{A}_1 & 0 \\ 0 & \boldsymbol{A}_2\end{bmatrix}\boldsymbol{x} + \begin{bmatrix}\boldsymbol{B}_1 & 0 \\ 0 & \boldsymbol{B}_2\end{bmatrix}\boldsymbol{u} \\
\begin{bmatrix}\boldsymbol{y}_1 \\ \boldsymbol{y}_2\end{bmatrix} &= \begin{bmatrix}\boldsymbol{C}_1 & 0 \\ 0 & \boldsymbol{C}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{bmatrix}
\end{aligned}$$

This new system simultaneously describes the evolution of the two subsystems.

Let's consider the case of two SISO systems with the output of S_1 connected to the input of S_2. That is, we would like to set $u_2 = y_1$. We will make a model of this situation by coupling the two systems together and eliminating the connections between the outside world and both u_2 and y_1.

$$\begin{aligned}
\dot{\boldsymbol{x}_1} &= \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{B}_1 u_1 \\
\dot{\boldsymbol{x}_2} &= \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{B}_2 u_2 \\
&= \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{B}_2 y_1 \\
&= \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{B}_2\boldsymbol{C}_1\boldsymbol{x}_1 \\
\rightsquigarrow \dot{\boldsymbol{x}} &= \begin{bmatrix}\boldsymbol{A}_1 & 0 \\ \boldsymbol{B}_2\boldsymbol{C}_1 & \boldsymbol{A}_2\end{bmatrix}\boldsymbol{x} + \begin{bmatrix}\boldsymbol{B}_1 \\ 0\end{bmatrix} u_1 \\
y &= \begin{bmatrix}0 & \boldsymbol{C}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{bmatrix}
\end{aligned}$$

As an example, imagine that we have some system S described by the state space model $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}\}$. We want to connect a low pass RC filter to u_1 of the plant and make a new state space model for the augmented system. Let's first find the state space model of the filter in figure 5.11. Let u be the

input voltage for the circuit, x be the capacitor voltage and y be the output.

$$\dot{x}_1 = \frac{i}{C} = \frac{u_1 - x_1}{RC} = -\frac{1}{RC}x_1 + \frac{1}{RC}u_1$$
$$\implies \boldsymbol{A}_\mathrm{f} = \left[-\tfrac{1}{RC}\right], \boldsymbol{B}_\mathrm{f} = \left[\tfrac{1}{RC}\right], \boldsymbol{C}_\mathrm{f} = \left[1\right], \boldsymbol{D}_\mathrm{f} = \left[0\right]$$

Without loss of generality, let's consider a plant with three states and two inputs. We now add a fourth state (the input filter capacitor voltage). We also want to make our new input replace u_1 of the original system.

$$\text{Before:} \qquad \dot{\boldsymbol{x}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \boldsymbol{u}$$

$$\text{After:} \qquad \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{x}_\mathrm{f} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_{11} \\ a_{21} & a_{22} & a_{23} & b_{21} \\ a_{31} & a_{32} & a_{33} & b_{31} \\ 0 & 0 & 0 & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_\mathrm{f} \end{bmatrix} + \begin{bmatrix} 0 & b_{12} \\ 0 & b_{22} \\ 0 & b_{32} \\ \frac{1}{RC} & 0 \end{bmatrix} \boldsymbol{u}$$

We can now see that the new filter takes u_1 and couples it into the fourth state variable, which keeps track of the capacitor voltage. That voltage is connected to the system where the old u_1 was, so its effect on the remainder of the state vector is the same as u_1 used to be.

We will frequently find that we want to treat different inputs differently in our problems. For example, we might apply feedback to a system, while others are left unconnected. Or we might use some inputs to model disturbances entering the system (inputs that the universe uses to push on the system), while others are accessible to the control designer. In such cases we can split the $\boldsymbol{B}$ matrix into constituent parts. Consider a $\boldsymbol{B}$ matrix that couples three inputs into a system, so we have $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \end{bmatrix}$. That is, we have

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

We can pull out u_3 (say) so that we can use it for some other purpose.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_3 \end{bmatrix} u_3$$

5.5 Discrete Time State Space Models

Sometimes it is more convenient to describe a state space system in discrete time, rather than the continuous time treatment used above.

- The system/plant might actually be a true discrete time system. An example would be a state space system describing the number of students enrolled in various courses in different years.
- We might want to use a computer to control an underlying continuous time system, so we need to approximate the continuous time system with a sampled time version. We will postpone such systems for now.

Discrete time systems are usually modelled using difference equations. A difference equation gives the next state as a function of the current state and any inputs.

$$\boldsymbol{x}(t+1) = \boldsymbol{A}_\mathrm{d}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}(t)$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$
$$t \in \mathbb{Z}_+$$

We have used the "d" subscript on the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices to highlight that these are *not* the same matrices as for a continuous time system. Often these subscripts can be omitted, as the meaning of the various matrices is clear from context. Notice that the state equation has an important difference to the continuous time equation as it specifies what the next state will be, not what the change in the state will be. This is the reason that there are differences between some of the properties of the Laplace and z transforms, despite the fact that they play analogous roles for continuous and discrete time systems respectively.

Imagine that you had two bank accounts. We will represent the balance of the first account as x_1 and that of the second as x_2. In a single time interval the first account generates 3% interest in each time interval, all of which stays in account one. The second account generates 2% interest per time interval, which you split between the two accounts.

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 1.03 & 0.01 \\ 0 & 1.01 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$
$$\boldsymbol{x}(t+1) = \begin{bmatrix} 1.03 & 0.01 \\ 0 & 1.01 \end{bmatrix} \boldsymbol{x}(t)$$

Notice that we are not concerned here about what the time interval actually is, but that the model would need to change if we decided to model in a different time step (months rather than years for example).

You deposit a total amount u_1 per time period, which you split evenly between the two accounts. You also make withdrawals u_2, which all come from account one. We can incorporate the effects of the "inputs" as

$$\boldsymbol{x}(t+1) = \begin{bmatrix} 1.03 & 0.01 \\ 0 & 1.01 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0.5 & -1 \\ 0.5 & 0 \end{bmatrix} \boldsymbol{u}(t)$$

We will consider the output of this system to be the total amount of money. That is, $y = x_1 + x_2$. This implies a discrete time state space model with

$$\boldsymbol{A} = \begin{bmatrix} 1.03 & 0.01 \\ 0 & 1.01 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0.5 & -1 \\ 0.5 & 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix}.$$

We can redraw our our graphical representation of the system in discrete time. Notice that the only change is that we change the integral to a unit delay which is typically denoted z^{-1}. This revised schematic is shown in figure 5.12. as shown in figure 5.12

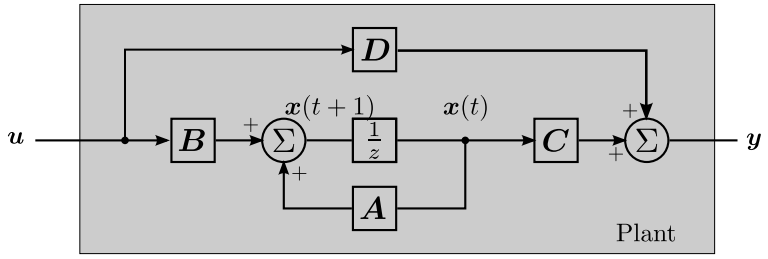


Figure 5.12: Graphical representation of the complete state space model for a discrete time system.

5.6 Linearisation

Most systems are non-linear in reality. However, in many cases using a linear model is reasonable, at least in restricted regions of operation. In particular, a control system often has a known operating point around which we can formulate a good model. Consider a non-linear, time-invariant, autonomous system with evolution governed by $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t)\big)$ for $\boldsymbol{f}$ some non-linear functions. Here $\boldsymbol{f}$ is a group of functions with f_i describing the evolution of x_i. Imagine that the system is operating near the point $\boldsymbol{x}_\mathrm{o}$. That is,

$$\boldsymbol{x}(t) = \boldsymbol{x}_\mathrm{o} + \boldsymbol{\xi}(t)$$

where $\boldsymbol{\xi}$ is a "small" perturbation away from the operating point $\boldsymbol{x}_\mathrm{o}$.

If the nonlinearity is smooth (ie differentiable everywhere) then we can linearise it by finding a linear approximation at each point. This is shown schematically in figure 5.13, which shows the dependence of a single state variable x_i on the state. We form a straight line approximation to $f_1(\boldsymbol{x})$ that is a tangent at $\boldsymbol{x}_\text{o}$.

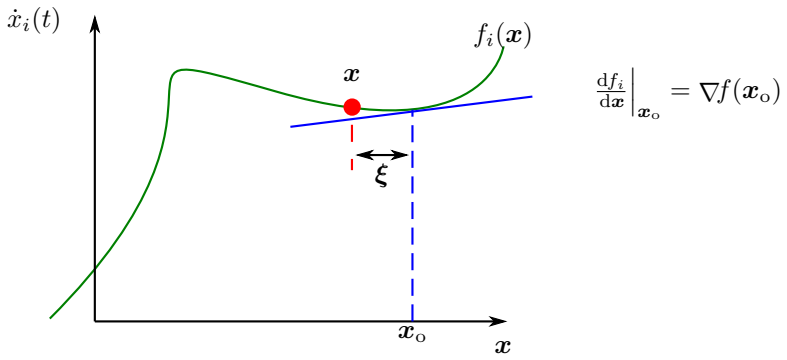


Figure 5.13: Linearisation of a system in the x_i direction. The system is operating near the operating point $\boldsymbol{x}_\text{o}$.

Now, we know that

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\big(\boldsymbol{x}(t)\big)\\ \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x}_\text{o}+\boldsymbol{\xi}) &= \boldsymbol{f}\big(\boldsymbol{x}_\text{o}+\boldsymbol{\xi}\big)\\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\text{o}+\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi} &= \boldsymbol{f}\big(\boldsymbol{x}_\text{o}+\boldsymbol{\xi}\big)\end{aligned}$$

We can find the Taylor series expansion of $\boldsymbol{f}$ around $\boldsymbol{x}_\text{o}$.

$$\begin{aligned}\implies \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\text{o}+\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi} &= \boldsymbol{f}(\boldsymbol{x}_\text{o})+\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\text{o})(\boldsymbol{x}_\text{o}-\boldsymbol{x})+o\big((\boldsymbol{x}_\text{o}-\boldsymbol{x})^2\big)\\ &= \boldsymbol{f}(\boldsymbol{x}_\text{o})+\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\text{o})\boldsymbol{\xi}+o(\boldsymbol{\xi}^2)\end{aligned}$$

Where $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}$ is called the Jacobian of $\boldsymbol{f}$.

Now $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\text{o} = \boldsymbol{f}(\boldsymbol{x}_\text{o})$ by definition, so we can cancel those terms from each side of our expansion.

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\text{o})\boldsymbol{\xi}+o(\boldsymbol{\xi}^2)$$

Because we assume that $\boldsymbol{\xi}$ is small, we know that $\boldsymbol{\xi}^2$ is very small, so we neglect the quadratic and higher order terms.

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi} \approx \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\text{o})\boldsymbol{\xi}$$

If you remember that the Jacobian of $\boldsymbol{f}$ at $\boldsymbol{x}_\text{o}$ is simply a matrix, we can see that this is simply a new state space model, expressed in terms of the state vector $\boldsymbol{\xi}$, rather than $\boldsymbol{x}$.

That is, we have

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi}$$

for

$$\boldsymbol{A} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\text{o}) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}_\text{o}) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}_\text{o}) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{x}_\text{o}) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{x}_\text{o}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{x}_\text{o}) & \frac{\partial f_n}{\partial x_2}(\boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_n}{\partial x_n}(\boldsymbol{x}_\text{o}) \end{bmatrix}$$

We typically don't bother talking about $\boldsymbol{\xi}$, but simply form a state space model in $\boldsymbol{x}$, with the understanding that the model applies only in the vicinity of the operating point $\boldsymbol{x}_\text{o}$.

We also need to also consider the possibility of having states that depend nonlinearly on the inputs $\boldsymbol{u}$. We will consider the input to be $\boldsymbol{u} = \boldsymbol{u}_\text{o} + \boldsymbol{v}$. If the states and the inputs are not intertwined, we can expand our model to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{u})$$

then we can proceed to linearise $\boldsymbol{g}$ as we did for $\boldsymbol{f}$ above. That is, we find

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{v}$$

for

$$\boldsymbol{B} = \frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{u}}(\boldsymbol{u}_\text{o}) := \begin{bmatrix} \frac{\partial g_1}{\partial u_1}(\boldsymbol{u}_\text{o}) & \frac{\partial g_1}{\partial u_2}(\boldsymbol{u}_\text{o}) & \cdots & \frac{\partial g_1}{\partial u_n}(\boldsymbol{u}_\text{o}) \\ \frac{\partial g_2}{\partial u_1}(\boldsymbol{u}_\text{o}) & \frac{\partial g_2}{\partial u_2}(\boldsymbol{u}_\text{o}) & \cdots & \frac{\partial g_2}{\partial u_n}(\boldsymbol{u}_\text{o}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial u_1}(\boldsymbol{u}_\text{o}) & \frac{\partial g_m}{\partial u_2}(\boldsymbol{u}_\text{o}) & \cdots & \frac{\partial g_m}{\partial u_n}(\boldsymbol{u}_\text{o}) \end{bmatrix}$$

Sometimes we have systems where the state and input are coupled (eg $\dot{x}_2 = x_1 u_1$), so we can't separate the model to $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{u})$, but must instead deal with the more general form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$. We proceed as before, but must evaluate the partial derivatives at $\boldsymbol{x}_\text{o}, \boldsymbol{u}_\text{o}$.

$$\boldsymbol{A} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_n}{\partial x_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_n}{\partial x_n}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) \end{bmatrix}$$

$$\boldsymbol{B} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{u}}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) := \begin{bmatrix} \frac{\partial f_1}{\partial u_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_1}{\partial u_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial g_1}{\partial u_m}(\boldsymbol{u}_\text{o}) \\ \frac{\partial f_2}{\partial u_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_2}{\partial u_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_2}{\partial u_m}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \frac{\partial f_n}{\partial u_2}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) & \cdots & \frac{\partial f_n}{\partial u_m}(\boldsymbol{u}_\text{o}, \boldsymbol{x}_\text{o}) \end{bmatrix}$$

It is worth remembering that a system may sometimes be linear in its state but not its inputs (or vise versa). In such cases it is not necessary to calculate the Jacobian for the linear section, as the required $\boldsymbol{A}$ or $\boldsymbol{B}$ can be written as is normal for a linear system.

As an example, consider a simple pendulum with length l and a bob with mass m as shown in figure 5.14. We can use Newton's second law to determine the motion of the bob.

$$\begin{aligned} I\ddot{\theta} &= \tau_\text{g} \\ &= -lF_\text{g}\sin\theta \\ &= -mgl\sin\theta \\ \ddot{\theta} &= -\frac{mgl}{ml^2}\sin\theta \\ &= -\frac{g}{l}\sin\theta \end{aligned}$$

Define $\begin{bmatrix} x_1, x_2 \end{bmatrix}^\mathsf{T} \equiv \begin{bmatrix} \theta, \dot{\theta} \end{bmatrix}^\mathsf{T}$

$$\rightsquigarrow \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{g}{l}\sin x_1 \end{cases}$$

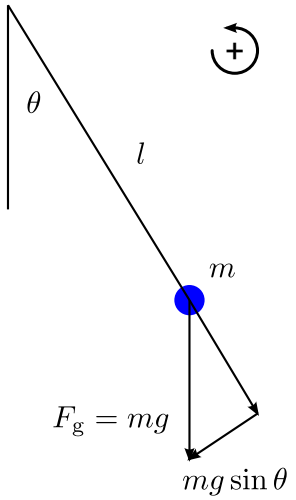


Figure 5.14: A simple pendulum

We can now linearise these equations around the operating point $\boldsymbol{x} = 0$.

$$\begin{aligned}
\dot{x}_1 &= f_1(\boldsymbol{x}) \equiv x_2 \\
\dot{x}_2 &= f_2(\boldsymbol{x}) \equiv -\frac{g}{l} \sin x_1 \\
\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(0) &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(0) & \frac{\partial f_1}{\partial x_2}(0) \\ \frac{\partial f_2}{\partial x_1}(0) & \frac{\partial f_2}{\partial x_2}(0) \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} \cos(x_1)\big|_{\boldsymbol{x}=0} & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{bmatrix}
\end{aligned}$$

Notice that in this case the fact that $x_2 = 0$ at the operating point was irrelevant. The linearised model is therefore insensitive to x_2, but only holds when x_1 is small.

Notice that the expression for the Jacobian applies everywhere if the nonlinearity is differentiable everywhere. For most of this course we will only worry about linearising about a single known operating point, but one can extend this idea to systems where the system is moving through state space. For such systems we only need to find the Jacobian expression once, and we could then substitute the changing operating point to extract a new linearised model around the evolving operating point.

The Matlab symbolic toolbox is useful for finding Jacobians.

```
syms g l x1 x2
f = [x2; -g/l*sin(x1)];
Df = jacobian(f, [x1 x2]);   % Find the Jacobian.
subs(Df, [x1, x2], [0, 0]);  % Substitute operating point.
ans =

[    0, 1]
[ -g/l, 0]
```

5.7 Nonlinear outputs

In practice outputs that are nonlinear functions of the state are common, but these are rarely troublesome to work with. For example, if you had a sensor that calculated x_i^3, it would be easy to infer x_i using a

"virtual sensor"; a combination of the real sensor and a postprocessing calculation. There are of course practical complications that arise with this approach, such as unusual noise characteristics or dynamic range issues.

Nonetheless, we will therefore typically assume that linear sensors are available, so we won't go through linearisation of output equations to find approximate $\boldsymbol{C}$ and $\boldsymbol{D}$ matrices, though this is certainly possible if a problem calls for it.

5.8 Appendix: Vector Differential Calculus

In the body of these notes we simply introduced the Jacobian as the correct way to linearise a nonlinear vector function at some operating point. If that treatment was not sufficiently clear you might appreciate the expanded version given here.

Let's begin by linearising $\dot{x} = f(x)$ around the operating point x_o. That is, we assume the system is near $\boldsymbol{x}_\mathrm{o}$, in fact we assume $\boldsymbol{x} = \boldsymbol{x}_\mathrm{o} + \xi$, where ξ is a small perturbation from the desired operating point.

We do this with a Taylor series expansion.

$$\begin{aligned}\dot{x} &= f(x) \\ &\approx f(x_\mathrm{o}) + \frac{\mathrm{d}f}{\mathrm{d}x}(x_\mathrm{o})(x - x_\mathrm{o}) + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_\mathrm{o})(x - x_\mathrm{o})^2 + o\left((x - x_\mathrm{o})^3\right)\end{aligned}$$

where $\frac{\mathrm{d}f}{\mathrm{d}x}(x_\mathrm{o})$ means that we take the partial derivative of f with respect to x and then evaluate the resulting function at $x = x_\mathrm{o}$.

$$\dot{x_\mathrm{o}} + \dot{\xi} = f(x_\mathrm{o}) + \frac{\mathrm{d}f}{\mathrm{d}x}(x_\mathrm{o})\xi + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_\mathrm{o})\xi^2 + o\left(\xi^3\right) \text{ where } \xi := x - x_\mathrm{o}$$

Now, $\dot{x}_\mathrm{o}$ is zero by definition. In addition $f(x_\mathrm{o})$ must also be zero because $\dot{x}_\mathrm{o} = f(x_\mathrm{o})$. If we neglect the quadratic and higher order terms we obtain

$$\dot{\xi} \approx \frac{\mathrm{d}f}{\mathrm{d}x}(x_\mathrm{o})\xi \quad \text{or equivalently } \dot{\xi} = a\xi \text{ for } a = \frac{\mathrm{d}f}{\mathrm{d}x}(x_\mathrm{o}).$$

Notice that we have now formed a linear model, where the scalar a is the slope of the linear equation that maps ξ into $\dot{\xi}$. That is, a is the scalar that maps the distance from the operating point to changes in that distance. In effect we have found the line that is tangent to $f(x)$ at the point $x = x_\mathrm{o}$, which we propose to use as an approximation to $f(x)$. As we move away from the operating point (as ξ becomes large) this approximation may or may not become a poor approximation.

To fully appreciate how we linearise a multivariable function, we need to generalise the notion of the derivative. Up until now you have seen how to differentiate a single variable function using normal differentiation, but you have also used partial differentiation to find how a multivariable function changes as a particular variable changes.

For example, say we have a function of two dimensions, $f(x_1, x_2)$. If we differentiate with respect to x_1 (that is we find $\frac{\partial f}{\partial x_1}$), we find the slope of the function as we move in the direction parallel to the x_1 axis.

Let's extend our treatment to consider how a single state variable, x_i, changes in a system that has many state variables. That is, consider a system $x_i = f_i(\boldsymbol{x})$. Again we perform a Taylor series expansion of f_i and discard higher order terms.

$$\begin{aligned}\dot{x}_i &= f_i(\boldsymbol{x}) \\ &\approx f_i(\boldsymbol{x}_\mathrm{o}) + \frac{\partial f_i}{\partial x_1}(\boldsymbol{x}_\mathrm{o})\xi_1 + \frac{\partial f_i}{\partial x_2}(\boldsymbol{x}_\mathrm{o})\xi_2 + \cdots + \frac{\partial f_i}{\partial x_n}(\boldsymbol{x}_\mathrm{o})\xi_n\end{aligned}$$

Notice that each of these terms is a partial derivative that tells us how quickly f_i changes as we move in the x_i direction. We then multiply by how far we have moved in the x_i direction to find the effect on $f_i(\boldsymbol{x})$.

$$\rightsquigarrow \dot{\xi}_i = \frac{\partial f_i}{\partial x_1}(\boldsymbol{x}_\mathrm{o})\xi_1 + \frac{\partial f_i}{\partial x_2}(\boldsymbol{x}_\mathrm{o})\xi_2 + \cdots + \frac{\partial f_i}{\partial x_n}(\boldsymbol{x}_\mathrm{o})\xi_n \implies \dot{\xi}_i = \nabla f_i(\boldsymbol{x}_\mathrm{o})^\mathsf{T}\boldsymbol{\xi}$$

where $\nabla f_i(\boldsymbol{x}_\mathrm{o})^\mathsf{T}$ is the row vector that is the gradient of f with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x} = \boldsymbol{x}_\mathrm{o}$ and $\boldsymbol{\xi} := \begin{bmatrix}\xi_1 & \xi_2 & \dots & \xi_n\end{bmatrix}^\mathsf{T}$.

We have again formed a linear model, but this time the linearisation is described by the gradient. That is, our original function $x_i = f(\boldsymbol{x})$ is a n-dimensional surface, which we have are approximating with an (n-1)-plane that is tangent to $f(\boldsymbol{x})$ at $\boldsymbol{x} = \boldsymbol{x}_\mathrm{o}$.

Note that we haven't done anything by introducing the gradient notation $\nabla \equiv \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}}$. It is just a convenient way of writing the derivative of a scalar function with respect to a vector. That is, the gradient is just the extension of our normal notion of the derivative to a multivariable function.

We finally can address our full problem, where we need to find an appropriate linearisation of *every* state variable with respect to the full state vector. We can find a separate equation to linearise each state variable, each of which will look as before. This will yield a set of n equations, that depend on the gradients of the different functions driving the state variables.

$$\begin{aligned}\dot{\boldsymbol{\xi}} &= \begin{bmatrix}\nabla f_1(\boldsymbol{x}_\mathrm{o})^\mathsf{T} \\ \nabla f_2(\boldsymbol{x}_\mathrm{o})^\mathsf{T} \\ \vdots \\ \nabla f_n(\boldsymbol{x}_\mathrm{o})^\mathsf{T}\end{bmatrix}\boldsymbol{\xi} \\ &= \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_\mathrm{o})\boldsymbol{\xi}\end{aligned}$$

as before. That is, the Jacobian is the generalisation of the gradient to a vector valued function.

6 Time Response of State Space Systems

6.1 Autonomous linear dynamical systems

Consider the autonomous ($\boldsymbol{u} = 0$) continuous-time dynamical system described by

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t)\\ t &\in \mathbb{R}_+\end{aligned}$$

We would like to find the solution for $\boldsymbol{x}$. That is, we would like an expression that describes how $\boldsymbol{x}$ evolves with time. To find this we must solve the differential equation $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$. We will then also be able to find $\boldsymbol{y}$ using $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$. Let's take the Laplace transform of the state equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) \overset{\mathcal{L}}{\Longleftrightarrow} s\boldsymbol{X}(s) - \boldsymbol{x}(0) = \boldsymbol{A}\boldsymbol{X}(s)$$

where $\boldsymbol{X}(s)$ here indicates $\boldsymbol{x}(t)$ transformed to the Laplace domain. (We have broken our normal convention that capital italics indicate matrices.) Looking at the components of this equation, and remembering that we have n total states, we see we have a set of equations, each having form

$$\begin{aligned}\frac{\mathrm{d}x_i(t)}{\mathrm{d}t} &= a_{i1}x_1(t) + a_{i2}x_2(t) + \cdots + a_{in}x_n(t)\\
&\Longrightarrow \begin{cases} sX_1(s) - a_{11}X_1 - a_{12}X_2 \cdots - a_{1n}X_n &= x_1(0)\\ sX_2(s) - a_{21}X_1 - a_{22}X_2 \cdots - a_{2n}X_n &= x_2(0)\\ & \vdots\\ sX_n(s) - a_{n1}X_1 - a_{n2}X_2 \cdots - a_{nn}X_n &= x_n(0)\end{cases}\\
&\Longrightarrow \begin{cases} (s-a_{11})X_1 - a_{12}X_2 \cdots - a_{1n}X_n &= x_1(0)\\ -a_{21}X_1 + (s-a_{22})X_2 \cdots - a_{2n}X_n &= x_2(0)\\ & \vdots\\ -a_{n1}X_1 - a_{n2}X_2 \cdots + (s-a_{nn})X_n &= x_n(0)\end{cases}\\
&\Longrightarrow \left(\begin{bmatrix} s & 0 & \cdots & 0\\ 0 & s & \cdots & 0\\ \vdots & & & \\ 0 & 0 & \cdots & s\end{bmatrix} - \begin{bmatrix} a_{11} & a_{11} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn}\end{bmatrix}\right)\begin{bmatrix}\boldsymbol{X}_1(s)\\ \boldsymbol{X}_2(s)\\ \vdots\\ \boldsymbol{X}_n(s)\end{bmatrix} = \begin{bmatrix}x_1(0)\\ x_2(0)\\ \vdots\\ x_n(0)\end{bmatrix}\end{aligned}$$

or,

$$\begin{aligned}(s\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{X}(s) &= \boldsymbol{x}(0)\\ \boldsymbol{X}(s) &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{x}(0)\end{aligned}$$

To find an expression for $\boldsymbol{x}(t)$ we take the inverse Laplace transform of the above.

$$\begin{aligned}\boldsymbol{X}(s) &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{x}(0)\\ \Longrightarrow\ \boldsymbol{x}(t) &= \mathcal{L}^{-1}\left\{(s\boldsymbol{I} - \boldsymbol{A})^{-1}\right\}\boldsymbol{x}(0)\\ &= \boldsymbol{\Phi}(t)\boldsymbol{x}(0)\end{aligned}$$

where $\boldsymbol{\Phi}(t) := \mathcal{L}^{-1}\left\{(s\boldsymbol{I} - \boldsymbol{A})^{-1}\right\}$ is the *state-transition* matrix. It performs the action of taking the system state at one time ($t = 0$) here, and producing the state at another time. To find $\boldsymbol{\Phi}$ one could first calculate $(s\boldsymbol{I} - \boldsymbol{A})^{-1}$, which will be a matrix of rational functions. The inverse Laplace transform can then calculated element by element. The above analysis assumes that $(s\boldsymbol{I} - \boldsymbol{A})^{-1}$ exists, which is not obvious (nor true for all s). We will return to this shortly.

Imagine that we have some autonomous continuous time system governed by $\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \boldsymbol{x}$. Let's find $\boldsymbol{\Phi}(t)$ for this system.

$$\begin{aligned}
\boldsymbol{A} &= \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \\
s\boldsymbol{I} - \boldsymbol{A} &= \begin{bmatrix} s & -1 \\ 2 & s+3 \end{bmatrix} \\
\implies (s\boldsymbol{I} - \boldsymbol{A})^{-1} &= \frac{1}{s(s+3)+2} \begin{bmatrix} s+3 & 1 \\ -2 & s \end{bmatrix} \\
&= \frac{1}{s^2+3s+2} \begin{bmatrix} s+3 & 1 \\ -2 & s \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{s+3}{s^2+3s+2} & \dfrac{1}{s^2+3s+2} \\ \dfrac{-2}{s^2+3s+2} & \dfrac{s}{s^2+3s+2} \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
(s\boldsymbol{I} - \boldsymbol{A})^{-1} &= \begin{bmatrix} \dfrac{s+3}{s^2+3s+2} & \dfrac{1}{s^2+3s+2} \\ \dfrac{-2}{s^2+3s+2} & \dfrac{s}{s^2+3s+2} \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{s+3}{(s+1)(s+2)} & \dfrac{1}{(s+1)(s+2)} \\ \dfrac{-2}{(s+1)(s+2)} & \dfrac{s}{(s+1)(s+2)} \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{2}{s+1} - \dfrac{1}{s+2} & -\dfrac{1}{s+1} + \dfrac{1}{s+2} \\ -\dfrac{2}{s+1} + \dfrac{2}{s+2} & -\dfrac{1}{s+1} + \dfrac{2}{s+2} \end{bmatrix} \\
\overset{\mathcal{L}^{-1}}{\rightsquigarrow} \boldsymbol{\Phi}(t) &= \begin{bmatrix} 2\mathrm{e}^{-t} - \mathrm{e}^{-2t} & -\mathrm{e}^{-t} + \mathrm{e}^{-2t} \\ -2\mathrm{e}^{-t} + 2\mathrm{e}^{-2t} & -\mathrm{e}^{-t} + 2\mathrm{e}^{-2t} \end{bmatrix}
\end{aligned}$$

We can now use this to find $\boldsymbol{x}$ at an time, given $\boldsymbol{x}(0)$.

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{\Phi}\boldsymbol{x}(0) \\
&= \begin{bmatrix} 2\mathrm{e}^{-t} - \mathrm{e}^{-2t} & -\mathrm{e}^{-t} + \mathrm{e}^{-2t} \\ -2\mathrm{e}^{-t} + 2\mathrm{e}^{-2t} & -\mathrm{e}^{-t} + 2\mathrm{e}^{-2t} \end{bmatrix} \boldsymbol{x}(0)
\end{aligned}$$

Writing out the components of $\boldsymbol{x}(t)$ we have

$$\begin{aligned}
x_1(t) &= 2x_1(0)\mathrm{e}^{-t} - x_1(0)\mathrm{e}^{-2t} - x_2(0)\mathrm{e}^{-t} + x_2(0)\mathrm{e}^{-2t} \\
x_2(t) &= -2x_1(0)\mathrm{e}^{-t} + 2x_1(0)\mathrm{e}^{-2t} - x_2(0)\mathrm{e}^{-t} + 2x_2(0)\mathrm{e}^{-2t}
\end{aligned}$$

6.1.1 $\boldsymbol{\Phi}$ and the Matrix Exponential

As an alternative, let us again consider $\boldsymbol{\Phi} = \mathcal{L}^{-1}\left\{(s\boldsymbol{I} - \boldsymbol{A})^{-1}\right\}$. We cannot analytically find the inverse Laplace transform for the general bracketed expression here, but we can rewrite it using a Laurent series expansion,

$$(s\boldsymbol{I} - \boldsymbol{A})^{-1} \approx \frac{1}{s}\boldsymbol{I} + \frac{1}{s^2}\boldsymbol{A} + \frac{1}{s^3}\boldsymbol{A}^2 + \cdots$$

Now we can calculate the inverse Laplace transform to get

$$\begin{aligned}\boldsymbol{\Phi} &= \mathcal{L}^{-1}\left\{(s\boldsymbol{I}-\boldsymbol{A})^{-1}\right\} \\ &= \boldsymbol{I}\mathcal{L}^{-1}\left\{\frac{1}{s}\right\} + \boldsymbol{A}\mathcal{L}^{-1}\left\{\frac{1}{s^2}\right\} + \boldsymbol{A}^2\mathcal{L}^{-1}\left\{\frac{1}{s^3}\right\} + \cdots \\ &= \boldsymbol{I} + t\boldsymbol{A} + \frac{1}{2!}(t\boldsymbol{A})^2 + \cdots\end{aligned}$$

This series may look familiar, as it is very similar to the Taylor series expansion for a (scalar) exponential.

$$\mathrm{e}^a = \sum_{k=0}^{\infty}\frac{a^k}{k!} = 1 + a + \frac{a^2}{2!} + \cdots + \frac{a^k}{k!} + \cdots$$

Let's define the *matrix exponential* analogously. That is,

$$\mathrm{e}^{\boldsymbol{M}} := \sum_{k=0}^{\infty}\frac{1}{k!}\boldsymbol{M}^k \text{ and therefore } \mathrm{e}^{t\boldsymbol{A}} = \boldsymbol{I} + t\boldsymbol{A} + \frac{1}{2!}(t\boldsymbol{A})^2 + \cdots$$

Note that the matrix exponential is *not* generally formed by taking the scalar exponential of each element of a matrix, though unfortunately that is what matlab will do. You must use `expm` in matlab instead.

We have therefore defined the form of the state transition matrix in continuous time,

$$\boldsymbol{\Phi}(t) = \mathrm{e}^{t\boldsymbol{A}}\,.$$

We can use this matrix to calculate the state at an arbitrary time,

$$\boldsymbol{x}(t) = \mathrm{e}^{t\boldsymbol{A}}\boldsymbol{x}(0)\,.$$

In fact, we can write the state at any time given the state at any other time

$$\boldsymbol{x}(t_2) = \mathrm{e}^{(t_2-t_1)\boldsymbol{A}}\,\boldsymbol{x}(t_1)\,.$$

It is sometimes convenient to extend the our notation for $\boldsymbol{\Phi}$ to include two time instants; $\boldsymbol{\Phi}(t_1, t_0) := \mathrm{e}^{(t_1-t_0)\boldsymbol{A}}$.

6.1.2 State Transition example

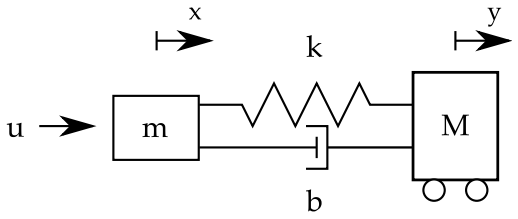


Figure 6.1: A system of two carts separated by a damped spring.

Consider the dual mass system we saw earlier, which had

$$\boldsymbol{x} := \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m} & -\frac{b}{m} & \frac{k}{m} & \frac{b}{m} \\ 0 & 0 & 0 & 1 \\ \frac{k}{M} & \frac{b}{M} & -\frac{k}{M} & -\frac{b}{M} \end{bmatrix}$$

Let's consider $m = 1$ kg, $M = 4$ kg, $k = 1$ N/m and $b = 0.2$ Ns/m and begin the simulation by pulling mass m one metre to the left. Thus $\boldsymbol{x}(0) = \begin{bmatrix}-1 & 0 & 0 & 0\end{bmatrix}^{\mathsf{T}}$. We have assumed an autonomous system, so $\boldsymbol{u} = 0$. Figure 6.2 shows the resulting evolution of the state vector with time. In this case the evolution was determined by calculating $\boldsymbol{\Phi}$ for each time instant and calculating the new state vector directly.

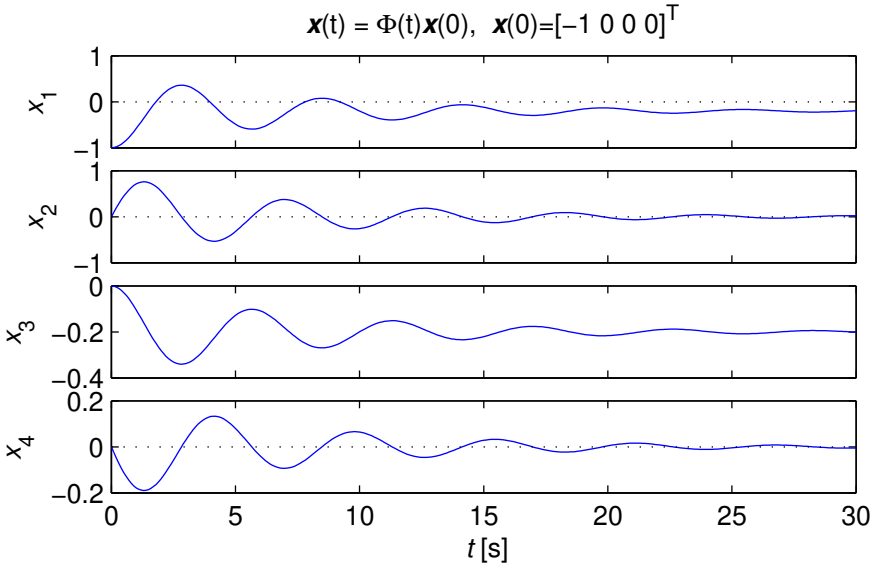


Figure 6.2: Response of the state variables of the system in figure 6.1 as it relaxes from the initial condition $\boldsymbol{x}(0) = [-1\ 0\ 0\ 0]^\mathsf{T}$.

6.2 Qualitative Evolution of Autonomous Systems

Recall that we previously discussed the fact that the inverse of $(s\boldsymbol{I} - \boldsymbol{A})$ does not exist everywhere. We know that

$$(s\boldsymbol{I} - \boldsymbol{A})^{-1} = \frac{\text{adj}(s\boldsymbol{I} - \boldsymbol{A})}{\det(s\boldsymbol{I} - \boldsymbol{A})},$$

so the inverse does not exist when $\det(s\boldsymbol{I} - \boldsymbol{A}) = 0$. Therefore, for $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, there are n values of s (possibly not distinct) for which the inverse does not exist. This happens at values of s that cause the matrix $s\boldsymbol{I} - \boldsymbol{A}$ to drop rank. You can think of looking at each column in turn and choosing a value of s that forces that column to be a linear combination of other columns. This is analogous to the transfer function representation when the denominator becomes zero, resulting in an undefined transfer function. The values of s for which this occurred are known as *poles* and are intimately linked to the response of the system.

The poles of $(s\boldsymbol{I} - \boldsymbol{A})^{-1}$ again lead directly to the system modes.

$$(s\boldsymbol{I} - \boldsymbol{A})^{-1} = \frac{\text{adj}(s\boldsymbol{I} - \boldsymbol{A})}{\det(s\boldsymbol{I} - \boldsymbol{A})} = \frac{\begin{bmatrix} n_{11}(s) & n_{12}(s) & \cdots & n_{1n}(s) \\ n_{21}(s) & n_{22}(s) & \cdots & n_{2n}(s) \\ \vdots & \vdots & \ddots & \vdots \\ n_{n1}(s) & n_{n2}(s) & \cdots & n_{nn}(s) \end{bmatrix}}{\chi(s)},$$

where $\chi(s) = \det(s\boldsymbol{I} - \boldsymbol{A})$ is a polynomial of s and each of the $n_{ij}(s)$ terms is also a polynomial. In the Laplace domain, we can write the influence on each state variable of all of the state variables (including itself). We will denote the i-th row of a matrix $\boldsymbol{M}$ as $[\boldsymbol{M}]_{i:}$. For example,

$$\begin{aligned} X_1(s) &= \left[(s\boldsymbol{I} - \boldsymbol{A})^{-1}\right]_{1:} \boldsymbol{x}(0) \\ &= \frac{n_{11}(s)}{\chi(s)} x_1(0) + \frac{n_{12}(s)}{\chi(s)} x_2(0) + \cdots + \frac{n_{1n}(s)}{\chi(s)} x_n(0) \end{aligned}$$

Generalising, the state variable x_i has a Laplace transform given by

$$\begin{aligned} X_i(s) &= \frac{n_{i1}(s)}{\chi(s)}x_1(0) + \frac{n_{i2}(s)}{\chi(s)}x_2(0) + \cdots + \frac{n_{in}(s)}{\chi(s)}x_n(0) \\ \overset{\mathcal{L}^{-1}}{\rightsquigarrow}\ x_i(t) &= \mathcal{L}^{-1}\left\{\frac{n_{i1}(s)}{\chi(s)}\right\}x_1(0) + \cdots + \mathcal{L}^{-1}\left\{\frac{n_{in}(s)}{\chi(s)}\right\}x_n(0) \\ x_i(t) &= \sum_{j=1}^{n} x_j(0)\mathcal{L}^{-1}\left\{\frac{n_{ij}(s)}{\chi(s)}\right\} \end{aligned}$$

We therefore expect that $x_i(t)$ will exhibit modes according to the roots of $\chi(s)$, which is called the *characteristic polynomial* of $\boldsymbol{A}$. Not all modes associated with $\chi(s)$ need to appear in all the $x_i(t)$, as pole-zero cancellation can occur with the $n_{ij}(s)$. That is, there might be roots of $n_{ij}(s)$ that are also roots of $\chi(s)$. In this case the cancelled system modes will not occur in the corresponding state variable.

Note that this pole-zero cancellation is a reflection of the mathematical structure of a system. As a result the cancellation is "perfect" and does not suffer from any of the potential difficulties of pole-zero cancellation that we saw in previous courses.

We will later look at putting $\boldsymbol{A}$ into a special form (modal canonical form) so that each $x_i(t)$ has as simple a modal structure as possible.

6.3 Finding system modes

We will need a general method of finding the modes of a system. By definition this must be done by equating the characteristic polynomial to zero, which leads us to the *characteristic equation*.

$$\det(s\boldsymbol{I} - \boldsymbol{A}) = 0$$

You will recall that this was the equation that we solved to find the eigenvalues of the matrix $\boldsymbol{A}$. That is, the poles of a system are the same as the eigenvalues of its $\boldsymbol{A}$ matrix. This provides a very convenient method for finding system poles using tools like Matlab.

6.3.1 Modal Responses - example

Let's construct a random continuous time state space system having 5 states and 2 outputs. `S=rss(5,2); A=S.A;`

$$\boldsymbol{A} = \begin{bmatrix} -1.5304 & -0.61981 & 1.2697 & -2.1308 & 3.4795 \\ 0.089332 & -0.86263 & -1.2335 & 4.5293 & -8.2102 \\ -0.07122 & 1.4616 & -2.2078 & -0.42013 & 2.0314 \\ 1.9256 & -4.7075 & 1.6305 & -1.6624 & 0.6961 \\ -3.863 & 8.048 & -0.80761 & -1.9444 & -1.2127 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0.40184 \\ 1.4702 \\ -0.32681 \\ 0.81232 \\ 0.54554 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \\ 0.61251 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} -1.0516 & 0 & 0 & -0.42506 & -0.062791 \\ 0.39747 & 1.5163 & 1.636 & 0.58943 & -2.022 \end{bmatrix}$$

We can find the eigenvalues of the matrix $\boldsymbol{A}$.

```
>> eig(A)
-0.7483 +10.5927i
-0.7483 -10.5927i
-3.2353
-1.6087
-1.1354
```

We would therefore expect the state variables (and the outputs) to be dominated by a slowly decaying oscillatory mode ($\lambda = -0.75 \pm \mathrm{j}10.59$) and a set of faster exponential modes corresponding to $\lambda = -1.1, -1.6, -3.2$.

Figure 6.3 shows that this this is indeed the case. This figure shows the response of the system when it was placed into an initial state $\boldsymbol{x}(0) = \begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}^\mathsf{T}$ and allowed to evolve naturally. As expected from the eigenvalues that are present we see a combination of decaying exponential modes and exponentially modulated sinusoidal modes.

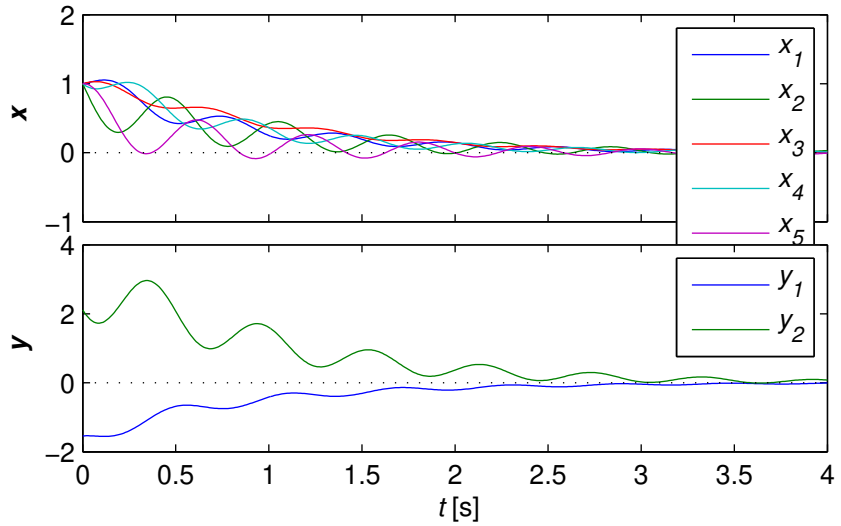


Figure 6.3: Response of a five-state, two-output state space model to an arbitrary initial condition $\boldsymbol{x}(0) = [1\ 1\ 1\ 1\ 1]^\mathsf{T}$. Notice that the response of each state variable and of each output is a combination of different system modes.

It is revealing to look at the response that the system would have if we set the initial conditions to one of the eigenvectors of $\boldsymbol{A}$. Consider what happens when we set the initial state of dynamical system to one of the system's eigenvector and allow the system to evolve with time.

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x}\\ \boldsymbol{x}(0) &= \boldsymbol{v} \text{ where } \lambda\boldsymbol{v} = \boldsymbol{A}\boldsymbol{v} \text{ for some } \lambda.\end{aligned}$$

We know that the state now evolves according to

$$\begin{aligned}\boldsymbol{x}(t) &= \mathrm{e}^{t\boldsymbol{A}}\boldsymbol{x}(0)\\ &= \mathrm{e}^{t\boldsymbol{A}}\boldsymbol{v}\\ &= \left[\boldsymbol{I} + t\boldsymbol{A} + \frac{(t\boldsymbol{A})^2}{2!} + \frac{(t\boldsymbol{A})^3}{3!} + \cdots\right]\boldsymbol{v}\\ &= \boldsymbol{v} + t\boldsymbol{A}\boldsymbol{v} + \frac{t^2\boldsymbol{A}^2\boldsymbol{v}}{2!} + \frac{t^3\boldsymbol{A}^3\boldsymbol{v}}{3!} + \cdots\end{aligned}$$

Now we know that $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$. Therefore

$$\begin{aligned}\boldsymbol{A}^2\boldsymbol{v} &= \boldsymbol{A}\boldsymbol{A}\boldsymbol{v}\\ &= \boldsymbol{A}\lambda\boldsymbol{v}\\ &= \lambda\boldsymbol{A}\boldsymbol{v}\\ &= \lambda^2\boldsymbol{v}\end{aligned}$$

and similarly, $\boldsymbol{A}^k\boldsymbol{v} = \lambda^k\boldsymbol{v}$

Substituting back into our evolution equation we get

$$\begin{aligned}\boldsymbol{x}(t) &= \boldsymbol{v} + t\boldsymbol{A}\boldsymbol{v} + \frac{t^2\boldsymbol{A}^2\boldsymbol{v}}{2!} + \frac{t^3\boldsymbol{A}^3\boldsymbol{v}}{3!} + \cdots \\ &= \boldsymbol{v} + t\lambda\boldsymbol{v} + \frac{t^2\lambda^2\boldsymbol{v}}{2!} + \frac{t^3\lambda^3\boldsymbol{v}}{3!} + \cdots \\ \boldsymbol{x}(t) &= \mathrm{e}^{\lambda t}\boldsymbol{v}\end{aligned}$$

That is, if we start a dynamical system with its state vector equal to an eigenvector (or a multiple) then it will evolve according to a simple exponential. Unless acted upon by an external input the system will remain in the subspace defined by the eigenvector.

Returning to our example, we find (numerically) that the eigenvectors of $\boldsymbol{A}$ are

$$\begin{bmatrix} -0.28 + 0.027\mathrm{j} \\ 0.63 \\ -0.12 - 0.10\mathrm{j} \\ -0.08 + 0.35\mathrm{j} \\ -0.039 - 0.61\mathrm{j} \end{bmatrix}, \begin{bmatrix} -0.28 - 0.027\mathrm{j} \\ 0.63 \\ -0.12 + 0.10\mathrm{j} \\ -0.08 - 0.35\mathrm{j} \\ -0.039 + 0.61\mathrm{j} \end{bmatrix}, \begin{bmatrix} 0.34 \\ 0.10 \\ -0.71 \\ 0.46 \\ 0.39 \end{bmatrix}, \begin{bmatrix} -0.68 \\ -0.17 \\ 0.21 \\ 0.62 \\ 0.29 \end{bmatrix}, \begin{bmatrix} 0.52 \\ 0.40 \\ 0.63 \\ 0.39 \\ 0.14 \end{bmatrix}$$

We will place the system into an initial state that is one of the eigenvectors (or a multiple of an eigenvector) and then watch the evolution of the system. We arbitrarily choose the last eigenvector, which is associated with the eigenvalue $\lambda = -1.1354$. The resulting plot can be seen in figure 6.4. Notice that each of the state variables (and the outputs) have a very simple response that is a scaled eigenmode corresponding to the eigenvalue of -1.1354. That is, in this case each state variable responds as $c_i\mathrm{e}^{-1.13t}$ for an appropriately chosen constant c_i.

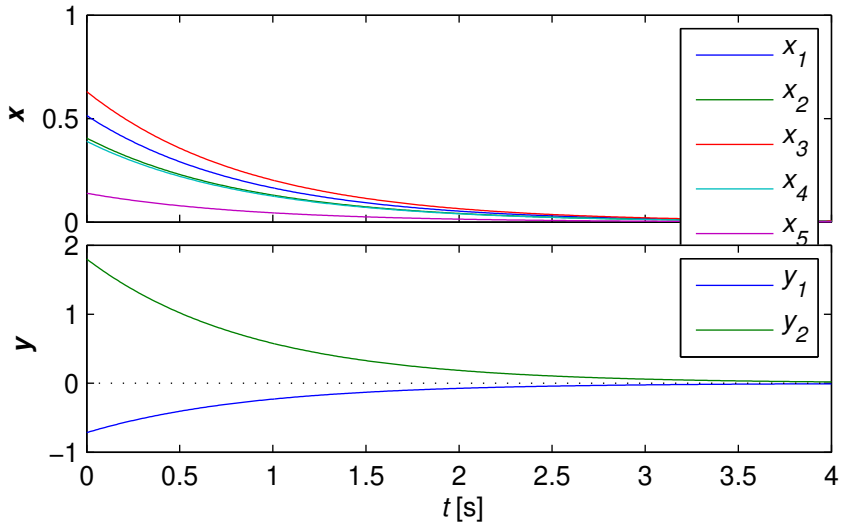


Figure 6.4: Response of the system when the initial condition is set to the eigenvector associated with eigenvalue -1.1354.

The same pattern is apparent if we examine the response to an initial condition corresponding to the other eigenvectors, as shown in figures 6.5 to 6.8. We again see that in each case all of the state variables respond as $c_i\mathrm{e}^{\lambda t}$. In the case of the complex eigenvalues you can see that the multiplier of each mode can be complex, as we observe different phase shifts in the different response.

6.3.2 Complex modes

We could extend this argument to the case where the eigenvalue (and eigenvector) are complex. In this case we can't prepare the system in the direction defined by the eigenvector, because we can't

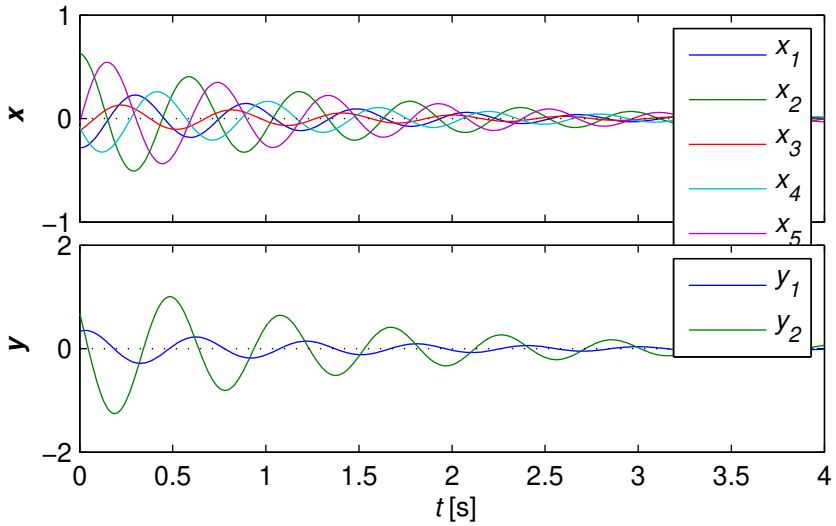


Figure 6.5: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue $-0.7483 + 10.5927\text{j}$.

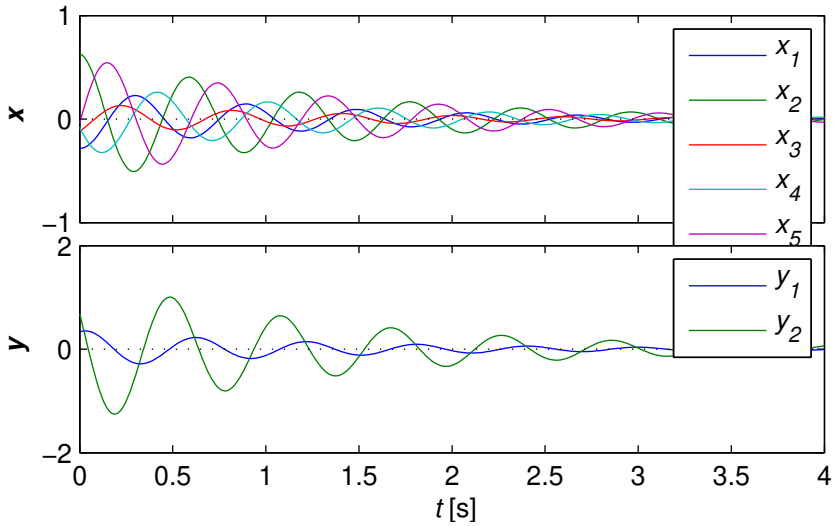


Figure 6.6: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue $-0.7483 - 10.5927\text{j}$.

have a complex state variable. However, we could put the system into an initial state that is a linear combination of the two conjugate eigenvectors. The algebra is somewhat tedious, but we find that this results in a mode that combines elements of rotation and amplification in the plane that is a basis for the two eigenvectors. This happens in the way you would expect, with the real part of the eigenvalue determining the growth or shrinkage rate, while the imaginary part of the eigenvalue determines how rapidly the mode rotates.

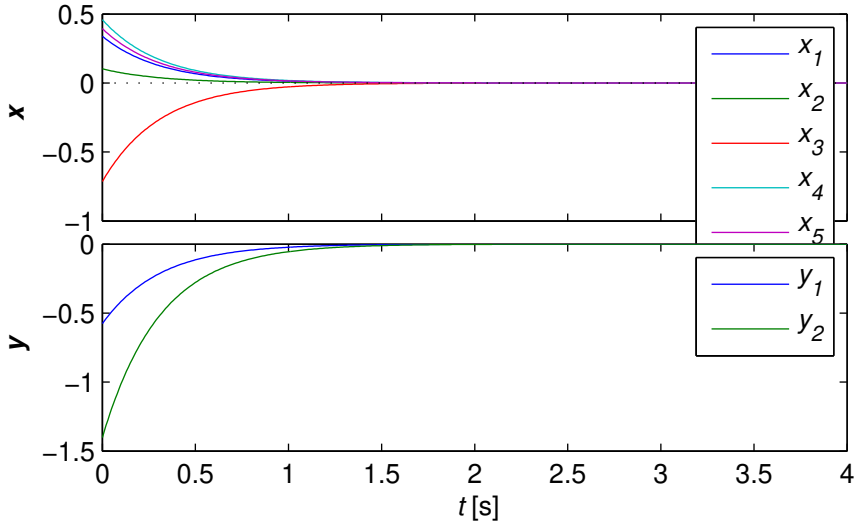


Figure 6.7: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue -3.2353.

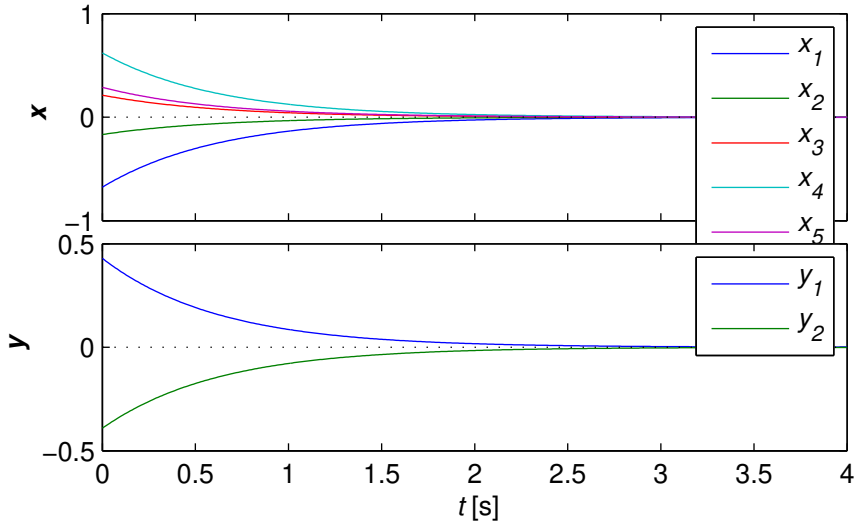


Figure 6.8: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue -1.6087.

6.4 Non-autonomous Systems

We will now repeat the above analysis but include an input term $\boldsymbol{Bu}$.

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{Ax} + \boldsymbol{Bu} \\
s\boldsymbol{X}(s) - \boldsymbol{x}(0) &= \boldsymbol{AX}(s) + \boldsymbol{BU}(s) \\
(s\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{X}(s) &= \boldsymbol{x}(0) + \boldsymbol{BU}(s) \\
\boldsymbol{X} &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{x}(0) + (s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{BU}(s)
\end{aligned}$$

Again, we need to find the inverse Laplace transform of this expression. Notice that the last term is formed by multiplying two expressions in the s-domain, which corresponds to convolution in the time domain.

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{\Phi}(t)\boldsymbol{x}(0) + \boldsymbol{\Phi}(t) * \mathcal{L}^{-1}\left\{\boldsymbol{BU}(s)\right\} \\
&= \boldsymbol{\Phi}(t)\boldsymbol{x}(0) + \int_0^t \boldsymbol{\Phi}(t-\tau)\boldsymbol{Bu}(\tau)\,\mathrm{d}\tau
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{x}(t) &= \mathrm{e}^{t\boldsymbol{A}}\boldsymbol{x}(0) + \int_0^t \mathrm{e}^{(t-\tau)\boldsymbol{A}}\boldsymbol{Bu}(\tau)\,\mathrm{d}\tau \\
\boldsymbol{y}(t) &= \boldsymbol{C}\mathrm{e}^{t\boldsymbol{A}}\boldsymbol{x}(0) + \boldsymbol{C}\int_0^t \mathrm{e}^{(t-\tau)\boldsymbol{A}}\boldsymbol{Bu}(\tau)\,\mathrm{d}\tau + \boldsymbol{Du}(t)
\end{aligned}$$

These equations are awkward to work with directly because of the convolution, so to simulate a system using matlab you can use the `lsim` command. However if there is no input ($\boldsymbol{u} = 0$) then you are arguably better off using the matrix exponential directly.

6.5 Converting Continuous Time to Discrete Time Systems

There are many occasions in which we have a model for a system that is formulated in continuous time, but we wish to work in discrete time. This might be because we plan to implement a controller with a computer for example. We would therefore like a general method for converting a continuous time system to an equivalent model in discrete time, for some specified sampling time.

Consider the problem of sampling a continuous time system with a sampling interval t_s. We want to form a discrete time model that gives the state at a given sample as a function of previous states and inputs. We will assume that the input is constant during a sampling period (that is, we assume zero order hold). So we want to take the continuous time state equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_\mathrm{c}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{c}\boldsymbol{u}(t)$$

and form a discrete time approximation

$$\boldsymbol{x}(t + t_\mathrm{s}) = \boldsymbol{A}_\mathrm{d}\boldsymbol{x}_\mathrm{d}(t) + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}_\mathrm{d}(t)$$

with the understanding that $\boldsymbol{x}_\mathrm{d}$ and $\boldsymbol{u}_\mathrm{d}$ are constant for all times between t and $t + t_\mathrm{s}$.

For convenience we introduce some extra notation for the discrete time system.

$$\begin{aligned}
\boldsymbol{x}(t + t_\mathrm{s}) = \boldsymbol{x}(t_\mathrm{s}(k+1)) &\equiv \boldsymbol{x}_\mathrm{d}(k+1) \\
\boldsymbol{x}(t) = \boldsymbol{x}(kt_\mathrm{s}) &\equiv \boldsymbol{x}_\mathrm{d}(k) \\
\boldsymbol{u}(t) = \boldsymbol{u}(kt_\mathrm{s}) &\equiv \boldsymbol{u}_\mathrm{d}(k)
\end{aligned}$$

Thus the state space equation becomes

$$\begin{aligned}
\boldsymbol{x}_\mathrm{d}(k+1) &= \boldsymbol{A}_\mathrm{d}\boldsymbol{x}_\mathrm{d}(k) + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}_\mathrm{d}(k) \\
k &\in \mathbb{Z}_+
\end{aligned}$$

Consider first an autonomous ($\boldsymbol{u} = 0$) continuous time system governed by $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_\mathrm{c}\boldsymbol{x}(t)$. We know that the evolution of this system is governed by the state transition matrix $\boldsymbol{\Phi}(t)$. Thus if we know the state at time t_1, then we can predict the state t_s later using $\boldsymbol{\Phi}$.

$$\begin{aligned}
\boldsymbol{x}(t_1 + t_\mathrm{s}) &= \boldsymbol{\Phi}(t_1 + t_\mathrm{s}, t_1)\boldsymbol{x}(t_1) \\
&= \mathrm{e}^{(t_1+t_\mathrm{s}-t_1)\boldsymbol{A}_\mathrm{c}}\boldsymbol{x}(t_1) \\
&= \mathrm{e}^{t_\mathrm{s}\boldsymbol{A}_\mathrm{c}}\boldsymbol{x}(t_1) \\
\boldsymbol{x}_\mathrm{d}(k+1) &= \mathrm{e}^{t_\mathrm{s}\boldsymbol{A}_\mathrm{c}}\boldsymbol{x}_\mathrm{d}(k) \\
\text{but by definition, } \boldsymbol{x}_\mathrm{d}(k+1) &= \boldsymbol{A}_\mathrm{d}\boldsymbol{x}_\mathrm{d}(k) \\
\implies \boldsymbol{A}_\mathrm{d} &= \mathrm{e}^{t_\mathrm{s}\boldsymbol{A}_\mathrm{c}}
\end{aligned}$$

Now let's consider the term arising from an input. Recall that this is a convolution between the state transition matrix and the input, which we have previously written as

$$\int_0^{t_\mathrm{s}} \mathrm{e}^{(t-\tau)\boldsymbol{A}_\mathrm{c}}\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(\tau)\,\mathrm{d}\tau\,.$$

We can equivalently write this convolution as

$$\int_0^{t_\mathrm{s}} \mathrm{e}^{\tau\boldsymbol{A}_\mathrm{c}}\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau\,.$$

But we know that $\boldsymbol{B}_\mathrm{c}\boldsymbol{u} = \boldsymbol{B}_\mathrm{c}\boldsymbol{u}(k)$, which is assumed to be constant for $t \in [0, t_\mathrm{s})$. We can therefore move it outside the integral.

$$\left(\int_0^{t_\mathrm{s}} \mathrm{e}^{\tau\boldsymbol{A}_\mathrm{c}}\,\mathrm{d}\tau\right)\boldsymbol{B}_\mathrm{c}\boldsymbol{u}(k)\,.$$

We now have an equation for the input's contribution, which is of the form $\boldsymbol{B}_\mathrm{d}\boldsymbol{u}(k)$ with

$$\boldsymbol{B}_\mathrm{d} = \left(\int_0^{t_\mathrm{s}} \mathrm{e}^{\tau\boldsymbol{A}_\mathrm{c}}\,\mathrm{d}\tau\right)\boldsymbol{B}_\mathrm{c}$$

If we replaced $\boldsymbol{A}_\mathrm{c}$ in this equation with a scalar a then you could solve the integral:

$$\int_0^{t_\mathrm{s}} \mathrm{e}^{a\tau}\,\mathrm{d}\tau = \left[\frac{1}{a}\mathrm{e}^{a\tau}\right]_0^{t_\mathrm{s}} = \frac{1}{a}\left(\mathrm{e}^{at_\mathrm{s}} - 1\right)$$

It may not surprise you that iff $\boldsymbol{A}_\mathrm{c}$ is invertible, then we can write

$$\boldsymbol{B}_\mathrm{d} = \boldsymbol{A}_\mathrm{c}^{-1}\left(\mathrm{e}^{t_\mathrm{s}\boldsymbol{A}_\mathrm{c}} - \boldsymbol{I}\right)\boldsymbol{B}_\mathrm{c}$$

We can therefore use one of these equations to form our discrete time $\boldsymbol{B}$ matrix from the continuous time model and the sampling time.

There is no change to the output equations when doing the discrete time conversion. Thus, we can convert a continuous time system to discrete time by replacing the system matrices as follows:

$$\begin{aligned}
\boldsymbol{A}_\mathrm{d} &= \mathrm{e}^{t_\mathrm{s}\boldsymbol{A}_\mathrm{c}} \\
\boldsymbol{B}_\mathrm{d} &= \left(\int_0^{t_\mathrm{s}} \mathrm{e}^{\tau\boldsymbol{A}_\mathrm{c}}\,\mathrm{d}\tau\right)\boldsymbol{B}_\mathrm{c} \\
\boldsymbol{C}_\mathrm{d} &= \boldsymbol{C}_\mathrm{c} \\
\boldsymbol{D}_\mathrm{d} &= \boldsymbol{D}_\mathrm{c}
\end{aligned}$$

The quality of the approximation will depend on the choice of t_s. Although our model depends on t_s, once we have a model we don't need to worry about it any more. The discrete time model simply tells

us how things evolve from one sampling interval to another. Similarly we will dispense with k and use t to describe time, with the understanding that t is an integer.

Matlab's `c2d` command can perform the continuous to discrete time conversion for you. It can also move in the other direction with `d2c` should that be required. Confusing a continuous and a discrete time model is probably the most frequent error in the course. Always double check that you have the right type of time domain when you enter a system model.

6.6 Time Response of Discrete Time Systems

We can calculate the response of an autonomous discrete time system $\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t)$. As we are now interested in the discrete time response of our discrete time system we will drop the d subscripts.

$$\begin{aligned}
\boldsymbol{x}(1) &= \boldsymbol{A}\boldsymbol{x}(0) \\
\boldsymbol{x}(2) &= \boldsymbol{A}\boldsymbol{x}(1) = \boldsymbol{A}^2\boldsymbol{x}(0) \\
\boldsymbol{x}(3) &= \boldsymbol{A}\boldsymbol{x}(2) = \boldsymbol{A}^3\boldsymbol{x}(0) \\
&\vdots \\
\boldsymbol{x}(t) &= \boldsymbol{A}^t\boldsymbol{x}(0) \\
\\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) \\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{A}^t\boldsymbol{x}(0)
\end{aligned}$$

Now consider a non-autonomous system $\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$.

$$\begin{aligned}
\boldsymbol{x}(1) &= \boldsymbol{A}\boldsymbol{x}(0) + \boldsymbol{B}\boldsymbol{u}(0) \\
\boldsymbol{x}(2) &= \boldsymbol{A}^2\boldsymbol{x}(0) + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{B}\boldsymbol{u}(1) \\
\boldsymbol{x}(3) &= \boldsymbol{A}^3\boldsymbol{x}(0) + \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(1) + \boldsymbol{B}\boldsymbol{u}(2) \\
&\vdots \\
\boldsymbol{x}(t) &= \boldsymbol{A}^t\boldsymbol{x}(0) + \sum_{\tau=0}^{t-1} \boldsymbol{A}^{t-1-\tau}\boldsymbol{B}\boldsymbol{u}(\tau) \\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{A}^t\boldsymbol{x}(0) + \boldsymbol{C}\sum_{\tau=0}^{t-1} \boldsymbol{A}^{t-1-\tau}\boldsymbol{B}\boldsymbol{u}(\tau) + \boldsymbol{D}\boldsymbol{u}(t)
\end{aligned}$$

You can also use Matlab's `lsim` command to plot this response.

In continuous time we defined the state evolution matrix $\boldsymbol{\Phi}(t)$ so that $\dot{\boldsymbol{x}} = \boldsymbol{\Phi}\boldsymbol{x}$. In that case we found $\boldsymbol{\Phi} = \mathrm{e}^{t\boldsymbol{A}}$. The definition of $\boldsymbol{\Phi}(t)$ in discrete time can be found directly from the previous argument, where we see (for example) that $\boldsymbol{x}(3) = \boldsymbol{A}^3\boldsymbol{x}(0)$. By inspection we can see that *in discrete time*,

$$\boldsymbol{\Phi}(t) = \boldsymbol{A}^t$$

Or more generally,

$$\boldsymbol{\Phi}(t_2, t_1) = \boldsymbol{A}^{t_2 - t_1}$$

Notice that the discrete time response equations are only defined for integer time steps. It makes no sense to ask what is happening *between* sampling intervals. However, when we consider sampled data systems we are using a discrete time model as an approximation to a real system that does have a continuous time response. It therefore will be doing things between sampling intervals. If you should want to know what is happening at those times you need to resort to using the continuous time model. If the system has nonzero input signals ($\boldsymbol{u}$) then your continuous time simulation can often assume that they remain constant over each sampling interval.

7 State Transformations

We have previously seen how to move from a differential or difference equation description of a system through to a state space model. Often however we might already know a transfer function model of a system that we would like to convert to state space. Similarly we might have a state space model that we would like to convert back to transfer functions so that we can use the tools of classical control to investigate system properties. In this chapter we will begin by finding how to perform the conversions between the transfer function and state space worlds.

The transfer function representation of a system is unique. That is, for any system there is only one transfer function that provides a correct description of the underlying mechanisms at work. However, we will shortly see that this is not the case for state space models. Indeed, we have considerable freedom in formulating the model in a way that is convenient. Indeed, it can be shown that any system has an infinite number of possible state space model formulations. We will therefore need to know the general process that allows us to convert between different realisations.

There are several so-called *canonical forms* that express identical system structure, but yield a form that is particularly well suited for some task or other. We will discuss how to convert an arbitrary model into these special forms.

7.1 State Space to Transfer Function Conversion

We would like a way to find a transfer function representation of a system expressed in state space. We know that $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$. If we take the Laplace transform of this equation and ignore initial conditions we obtain

$$\begin{aligned} s\boldsymbol{X}(s) &= \boldsymbol{A}\boldsymbol{X}(s) + \boldsymbol{B}\boldsymbol{U}(s) \\ \implies (s\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{X}(s) &= \boldsymbol{B}\boldsymbol{U}(s) \\ \boldsymbol{X}(s) &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{B}\boldsymbol{U}(s) \end{aligned}$$

We also know that $\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X} + \boldsymbol{D}\boldsymbol{U}$, so substituting for $\boldsymbol{X}$, we get

$$\boldsymbol{Y} = \boldsymbol{C}\left((s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{B}\boldsymbol{U}(s)\right) + \boldsymbol{D}\boldsymbol{U} = \left[\boldsymbol{C}\left(s\boldsymbol{I} - \boldsymbol{A}^{-1}\right)\boldsymbol{B} + \boldsymbol{D}\right]\boldsymbol{U}$$

But the transfer function $\boldsymbol{G}$ is defined as $\boldsymbol{Y}/\boldsymbol{U}$, so

$$\boxed{\boldsymbol{G}(s) = \boldsymbol{C}\,(s\boldsymbol{I} - \boldsymbol{A})^{-1}\,\boldsymbol{B} + \boldsymbol{D}}$$

Notice that $\boldsymbol{G}$ is a matrix of transfer functions. That is, we have a set of transfer functions relating each input to each output of our system. For a system with m inputs and p outputs, $\boldsymbol{G}$ has dimension $p \times m$. This result makes sense if you consider the structure of the state space model as shown in figure

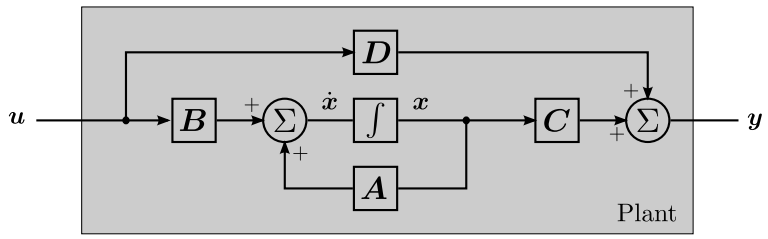


Figure 7.1: Graphical representation of the state space model of a plant.

7.1 and work through the block reduction algebra for a state x_i.

7.1.1 Example One

Find the transfer function for a system described in state space by

$$\boldsymbol{A} = \begin{bmatrix} -7 & -12 \\ 1 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 2 \end{bmatrix}, \boldsymbol{D} = 0.$$

$$\begin{aligned} s\boldsymbol{I} - \boldsymbol{A} &= \begin{bmatrix} s+7 & 12 \\ -1 & s \end{bmatrix} \\ \text{so, } (s\boldsymbol{I} - \boldsymbol{A})^{-1} &= \frac{1}{s(s+7)+12} \begin{bmatrix} s & -12 \\ 1 & s+7 \end{bmatrix} \\ \text{thus, } \boldsymbol{G}(s) &= \frac{\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} s & -12 \\ 1 & s+7 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{s(s+7)+12} \\ &= \frac{s+2}{s^2+7s+12} = \frac{s+2}{(s+3)(s+4)} \end{aligned}$$

7.1.2 Example Two

Consider a second example, taken from Stefani et al. "Design of Feedback Control Systems" 4th ed, p552.

$$\boldsymbol{A} = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 4 & 6 \\ -5 & 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & -1 \\ 8 & 1 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{aligned} \implies s\boldsymbol{I} - \boldsymbol{A} &= \begin{bmatrix} s+3 & -1 \\ 2 & s \end{bmatrix} \\ \text{so, } (s\boldsymbol{I} - \boldsymbol{A})^{-1} &= \frac{1}{s(s+3)+2} \begin{bmatrix} s & 1 \\ -2 & s+3 \end{bmatrix} \\ \text{thus, } \boldsymbol{G}(s) &= \frac{\begin{bmatrix} 1 & -1 \\ 8 & 1 \end{bmatrix} \begin{bmatrix} s & -1 \\ -2 & s+3 \end{bmatrix} \begin{bmatrix} 4 & 6 \\ -5 & 0 \end{bmatrix}}{s^2+3s+2} \\ &= \begin{bmatrix} \dfrac{9s+18}{s^2+3s+2} & \dfrac{6s+12}{s^2+3s+2} \\ \dfrac{27s-63}{s^2+3s+2} & \dfrac{48s-12}{s^2+3s+2} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \boldsymbol{G}(s) &= \begin{bmatrix} \dfrac{9(s+2)}{(s+1)(s+2)} & \dfrac{6(s+2)}{(s+1)(s+2)} \\ \dfrac{9(3s-7)}{(s+1)(s+2)} & \dfrac{12(4s-1)}{(s+1)(s+2)} \end{bmatrix} \\ \text{Now, } \boldsymbol{Y} &= \boldsymbol{GU} \\ \text{So, } \begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} &= \begin{bmatrix} \dfrac{9}{(s+1)} & \dfrac{6}{(s+1)} \\ \dfrac{9(3s-7)}{(s+1)(s+2)} & \dfrac{12(4s-1)}{(s+1)(s+2)} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \end{aligned}$$

$$\text{So, for example, } \frac{Y_2}{U_1} = 9\frac{3s-7}{(s+1)(s+2)}$$

7.2 State Transformations

While a LTI system is described by a unique transfer function, in general there will be *no* unique state space model.

- At a trivial level we could shuffle or scale the state variables.
- We could choose different quantities to represent the system in state space.

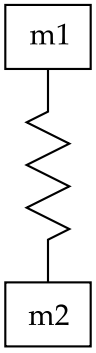


Figure 7.2: Two masses separated by a spring.

For example, for the system shown in figure 7.2 we could specify the state variables as:

1. The position and velocities of the two masses.
2. The position and rate of change of the centre of mass of the system and the separation between the objects.

These two options lead to quite different state space representations.

In fact, there are infinitely many ways we can represent a system in state space. Consider again the image of state space as a higher dimensional abstract space. We can describe the position of a point in that space by *any* set of coordinate axes that cover the entire space. That is, we are free to choose any appropriate set of basis vectors we like. For simplicity we will restrict ourselves to new state variables (ie a choice of new basis vectors) that are linear combinations of the old state variables. Thus, we can choose a state space representation that is convenient to the task at hand. Generally we will formulate our problem using physical insight, after which we we can transform the system to a convenient form.

Imagine a new state space realization that has a state vector $\boldsymbol{z}$, which is a linear transformation of $\boldsymbol{x}$. So for some non-singular (invertible) matrix $\boldsymbol{T}$, we can write

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{T}\boldsymbol{z} \\ \text{Now, } \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \implies \boldsymbol{T}\dot{\boldsymbol{z}} &= \boldsymbol{A}\boldsymbol{T}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u} \\ \dot{\boldsymbol{z}} &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}\boldsymbol{z} + \boldsymbol{T}^{-1}\boldsymbol{B}\boldsymbol{u} \end{aligned}$$

We can also rewrite the output equation in terms of the new state:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \\ &= \boldsymbol{C}\boldsymbol{T}\boldsymbol{z} + \boldsymbol{D}\boldsymbol{u} \end{aligned}$$

We have therefore formed a new state space realization $[\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \bar{\boldsymbol{D}}]$ with

$$\boxed{\bar{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \quad \bar{\boldsymbol{B}} = \boldsymbol{T}^{-1}\boldsymbol{B}, \quad \bar{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{T}, \quad \bar{\boldsymbol{D}} = \boldsymbol{D}}$$

7.2.1 State Transformation Example

Let's consider an example inspired by Stefani et.al. pp555-6. Consider a system having

$$\begin{aligned} \dot{x_1} &= x_1 + x_2 \\ \dot{x_2} &= x_1 + x_2 \\ y &= x_1 + x_2 \end{aligned}$$

$$\implies \boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

We want to find the new system matrices if we change the state variables to $z_1 = x_1 + x_2$ and $z_2 = x_1 - x_2$. Now,

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{T}\boldsymbol{z} \\ \boldsymbol{T}^{-1}\boldsymbol{x} &= \boldsymbol{z} \end{aligned}$$

As we know both $\boldsymbol{x}$ and $\boldsymbol{z}$ we can determine an appropriate $\boldsymbol{T}$.

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{T}^{-1}\boldsymbol{x} \\ \begin{bmatrix} x_1 + x_2 \\ x_1 - x_2 \end{bmatrix} &= \begin{bmatrix} (\boldsymbol{T}^{-1})_{11} & (\boldsymbol{T}^{-1})_{12} \\ (\boldsymbol{T}^{-1})_{21} & (\boldsymbol{T}^{-1})_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

$$\text{Recall that if } \boldsymbol{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \text{ then } \boldsymbol{M}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \implies \boldsymbol{T} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Thereby we find, $\bar{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ and $\bar{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{T} = \begin{bmatrix} 1 & 0 \end{bmatrix}$

In summary, we have two equivalent representations.
Original:

$$\begin{aligned} \dot{x_1} &= x_1 + x_2 \\ \dot{x_2} &= x_1 + x_2 \\ y &= x_1 + x_2 \end{aligned}$$

$$\rightarrow \boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Transformed:

$$\begin{aligned} \dot{z_1} &= 2z_1 \\ \dot{z_2} &= 0 \\ y &= z_1 \end{aligned}$$

$$\rightarrow \boldsymbol{A} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

7.2.2 Transfer Function of Transformed State.

Let's confirm that the change in representation has not resulted in a change in the transfer function. For algebraic simplicity let's assume that $\boldsymbol{D} = 0$. Recall that the system $[\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}]$ has a corresponding transfer function given by

$$\begin{aligned}
\bar{\boldsymbol{G}} &= \bar{\boldsymbol{C}} \left(s\boldsymbol{I} - \bar{\boldsymbol{A}}\right)^{-1} \bar{\boldsymbol{B}} \\
&= \boldsymbol{C}\boldsymbol{T} \left(s\boldsymbol{I} - \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^{-1}\boldsymbol{B} \\
&= \boldsymbol{C}\boldsymbol{T} \left(s\boldsymbol{T}^{-1}\boldsymbol{T} - \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^{-1}\boldsymbol{B} \\
&= \boldsymbol{C}\boldsymbol{T} \left[\boldsymbol{T}^{-1}\left(s\boldsymbol{I} - \boldsymbol{A}\right)\boldsymbol{T}\right]^{-1} \boldsymbol{T}^{-1}\boldsymbol{B} \\
\text{Now, } \left(\boldsymbol{X}\boldsymbol{Y}\boldsymbol{Z}\right)^{-1} &= \boldsymbol{Z}^{-1}\boldsymbol{Y}^{-1}\boldsymbol{X}^{-1} \text{ , so} \\
\bar{\boldsymbol{G}} &= \boldsymbol{C}\boldsymbol{T} \left[\boldsymbol{T}^{-1}\left(s\boldsymbol{I} - \boldsymbol{A}\right)^{-1}\boldsymbol{T}\right] \boldsymbol{T}^{-1}\boldsymbol{B} \\
&= \boldsymbol{C}\left(s\boldsymbol{I} - \boldsymbol{A}\right)^{-1}\boldsymbol{B} \\
\implies \bar{\boldsymbol{G}} &= \boldsymbol{G}
\end{aligned}$$

7.3 Canonical Forms

A transfer function is simply a description of the way in which a system's output(s) depend on its input(s). It has no information about the internal structure of a system, so it is not tied to any particular state representation. Because there is no privileged state representation for a system we have some choices to make when we do a conversion from a transfer function. There are three so-called *canonical* forms that we find useful for various tasks.

- Control canonical form,
- Observer canonical form,
- Modal (or diagonal) form and the related Jordan form.

Normally none of these forms will be the same as would be obtained by direct state-space modelling of a system.

7.3.1 Modal Canonical Form

A SISO transfer function $G(s) = \dfrac{n_n s^n + n_{n-1}s^{n-1} + \cdots + n_1 s + n_0}{s^n + d_{n-1}s^{n-1} + \cdots + d_1 s + d_0}$ which has no repeated poles can be expanded using partial fractions to get

$$G(s) = n_n + \frac{c_1}{s - \lambda_1} + \frac{c_2}{s - \lambda_2} + \cdots + \frac{c_n}{s - \lambda_n}$$

For some $c_i, \lambda_i \in \mathbb{C}$. This is represented by the system $\dot{\boldsymbol{x}} = \boldsymbol{A}_{\mathrm{m}}\boldsymbol{x} + \boldsymbol{B}_{\mathrm{m}}\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}_{\mathrm{m}}\boldsymbol{x} + \boldsymbol{D}_{\mathrm{m}}\boldsymbol{u}$, where

$$\boldsymbol{A}_{\mathrm{m}} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}, \boldsymbol{B}_{\mathrm{m}} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$\boldsymbol{C}_{\mathrm{m}} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \text{ and } \boldsymbol{D}_{\mathrm{m}} = n_n.$$

As an example, consider the transfer function

$$G(s) = \frac{s+3}{s^2 + 3s + 2} = \frac{s+3}{(s+1)(s+2)} = \frac{2}{s+1} + \frac{-1}{s+2}$$

Thus we have $c_1 = 2$, $\lambda_1 = -1$, $c_2 = -1$ and $\lambda_2 = -2$. Hence modal canonical form:

$$\begin{aligned}\boldsymbol{\Lambda} = \boldsymbol{A}_{\mathrm{m}} &= \begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix} = \begin{bmatrix}-1 & 0\\ 0 & -2\end{bmatrix}\\ \boldsymbol{B}_{\mathrm{m}} &= \begin{bmatrix}1\\ 1\end{bmatrix}\\ \boldsymbol{C}_{\mathrm{m}} &= \begin{bmatrix}c_1 & c_2\end{bmatrix} = \begin{bmatrix}2 & -1\end{bmatrix}\\ \boldsymbol{D}_{\mathrm{m}} &= 0\end{aligned}$$

Modal canonical form is extremely useful, because we can determine the modes of the system by inspection. It is often useful to convert a system to modal canonical form (or the related Jordan form) to gain insight into the system behaviour. Because the $\boldsymbol{A}_{\mathrm{m}}$ matrix has the eigenvalues arranged along its diagonal, you will often see it denoted $\boldsymbol{\Lambda}$. That is

$$\boldsymbol{\Lambda} := \mathrm{diag}(\lambda_i) \equiv \begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{bmatrix} \equiv \boldsymbol{A}_{\mathrm{m}}$$

7.3.2 Modal canonical form with complex poles

Transfer functions often include pairs of complex conjugate poles. If we strictly carried out the preceding method we obtain complex eigenvalues, which are a little awkward to deal with. There are a number of ways to avoid complex matrix entries, by relaxing the requirement that the $\boldsymbol{A}$ matrix be diagonal. The basic approach is to allow a two by two element section of $\boldsymbol{A}$ to represent a second order system and have a corresponding two element blocks in and $\boldsymbol{B}$ and $\boldsymbol{C}$.

One approach with complex poles represents each complex pole pair $\sigma_i \pm \mathrm{j}\omega_i$ as a 2 by 2 block of $\left[\begin{smallmatrix}\sigma & \omega\\ -\omega & \sigma\end{smallmatrix}\right]$ in the $\boldsymbol{A}$ matrix. This form is used by matlab for example. For example, let's write a state space version of the transfer function

$$\begin{aligned}G(s) &= 3 + \frac{6}{s-3} + \frac{-8}{s+4} + \frac{-5s+1}{s^2+2s+17} + \frac{7s}{s^2-3s+10}\\ &= 3 + \frac{6}{s-3} + \frac{-8}{s+4} + \frac{-5s+1}{(s-1.5+\mathrm{j}2.78)(s-1.5-\mathrm{j}2.78)}\\ &\qquad + \frac{7s}{(s+1+\mathrm{j}4)(s+1-\mathrm{j}4)}\end{aligned}$$

This form would have

$$\boldsymbol{A}_{\mathrm{m}} = \begin{bmatrix}1.5 & 2.78 & 0 & 0 & 0 & 0\\ -2.78 & 1.5 & 0 & 0 & 0 & 0\\ 0 & 0 & -1 & 4 & 0 & 0\\ 0 & 0 & -4 & -1 & 0 & 0\\ 0 & 0 & 0 & 0 & 3 & 0\\ 0 & 0 & 0 & 0 & 0 & -4\end{bmatrix}$$

This form of the $\boldsymbol{A}$ matrix is very useful, because you can directly read λ, σ and ω for all of the system modes. (You can use `canon(<System>,'modal');` in Matlab to get this matrix.)

7.4 Autonomous System Evolution using Modal Form

The modal canonical form has more than just intuitive appeal. It is very simple to calculate the autonomous evolution of a system that is expressed in modal canonical form. When we looked at the time evolution of state space systems, we saw that the evolution of initial states that were multiples of a system eigenvector were very simple, exponential responses. When we put a system into modal canonical

form, we are choosing the system eigenvectors to be the basis vectors for our representation. This is the representation for which the dynamics will as simple as possible.

Consider a discrete time system described by the matrix $\boldsymbol{A}$, which we will assume that we can diagonalise.

$$\begin{aligned}
\boldsymbol{\Lambda} = \boldsymbol{A}_{\mathrm{m}} &= \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} \text{ for some } \boldsymbol{V} \\
\implies \boldsymbol{A} &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} \\
\text{So, } \boldsymbol{\Phi}(t) \equiv \boldsymbol{A}^t &= \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)^t \\
&= \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)\cdots\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right) \\
&= \boldsymbol{V}\boldsymbol{\Lambda}^t\boldsymbol{V}^{-1} \\
\boldsymbol{\Phi}(t) &= \boldsymbol{V}\,\mathrm{diag}(\lambda_i^t)\boldsymbol{V}^{-1}
\end{aligned}$$

This equation does three things:

1. Multiply by $\boldsymbol{V}^{-1}$ to put a state into a modal representation;
2. Multiply by $\mathrm{diag}(\lambda_i^t)$ to evolve the state;
3. Multiply by $\boldsymbol{V}$ to get back to the representation we started in.

This is easy to solve, as we only need to find powers of scalars.

Similarly in continuous time,

$$\begin{aligned}
\boldsymbol{\Phi}(t) \equiv \mathrm{e}^{t\boldsymbol{A}} &= \boldsymbol{I} + t\boldsymbol{A} + \frac{1}{2!}(t\boldsymbol{A})^2 + \cdots \\
&= \boldsymbol{I} + t\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} + \frac{t^2}{2!}\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)^2 + \cdots \\
&= \boldsymbol{I} + t\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} + \frac{t^2}{2!}\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right) + \cdots \\
&= \boldsymbol{I} + t\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} + \frac{t^2}{2!}\left(\boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^{-1}\right) + \cdots \\
&= \boldsymbol{V}\left(\boldsymbol{I} + t\boldsymbol{\Lambda} + \frac{t^2}{2!}\boldsymbol{\Lambda}^2 + \cdots\right)\boldsymbol{V}^{-1} \\
\boldsymbol{\Phi}(t) &= \boldsymbol{V}\mathrm{e}^{t\boldsymbol{\Lambda}}\boldsymbol{V}^{-1}
\end{aligned}$$

This equation doesn't appear to gain us much, as it still requires a matrix exponential. However, unlike in the general case, for diagonal matrices like $\boldsymbol{\Lambda}$, the matrix exponential is equal to the elementwise scalar exponential.

7.5 Jordan Canonical form

Transfer functions that have no repeated poles can always be diagonalised to find a modal canonical form. The same is not generally true for systems with repeated poles. In some cases the best that can be done is to put the $\boldsymbol{A}$ matrix into Jordan canonical form, which includes some 1's in the superdiagonal. Recall from classical controls studies that a repeated pole at λ with multiplicity k leads to a mode of $t^{k-1}\mathrm{e}^{\lambda t}$, as well as a cascade of modes with lower powers of t, but the same decay (or growth) rate. We might expect to see these modes emerge from non-trivial Jordan blocks. That is, we will see that the presence of a $k \times k$ Jordan block in the $\boldsymbol{A}$ matrix of a system in Jordan canonical form implies that the time response of the system will contain terms like $t^{k-1}\mathrm{e}^{\lambda t}$.

7.5.1 Evolution of Systems in Jordan Form ($t \in \mathbb{Z}_+$)

For discrete time systems in Jordan form we find that calculation of the state transition matrix $\boldsymbol{\Phi}$ is relatively simple. This is because we can find the powers of each Jordan block separately. For example,

the t-th powers of a 3×3 Jordan block can be written as follows.

$$\text{If } \boldsymbol{A_J} = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} \text{ then, } \boldsymbol{A_J^t} = \begin{bmatrix} \lambda^t & t\lambda^{t-1} & \frac{t(t-1)}{2}\lambda^{t-2} \\ 0 & \lambda^t & t\lambda^{t-1} \\ 0 & 0 & \lambda^t \end{bmatrix}$$
$$= \begin{bmatrix} \lambda^t & \binom{t}{1}\lambda^{t-1} & \binom{t}{2}\lambda^{t-2} \\ 0 & \lambda^t & \binom{t}{1}\lambda^{t-1} \\ 0 & 0 & \lambda^t \end{bmatrix}$$

Extension to larger Jordan blocks should be self-evident.

Typically a system will have some eigenvalues that can be diagonalised, so we will have part of the Jordan form have trivial structure. Consider a discrete time dynamical system that is in the Jordan canonical form

$$\boldsymbol{A} = \boldsymbol{T}\boldsymbol{A_J}\boldsymbol{T}^{-1} \text{ where, } \boldsymbol{A_J} = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 & 0 \\ 0 & \lambda_1 & 1 & 0 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3 \end{bmatrix}$$

The state transition matrix for this system will be

$$\boldsymbol{\Phi}(t) = \boldsymbol{T} \begin{bmatrix} \lambda_1^t & t\lambda_1^{t-1} & \frac{t(t-1)}{2}\lambda_1^{t-2} & 0 & 0 \\ 0 & \lambda_1^t & t\lambda_1^{t-1} & 0 & 0 \\ 0 & 0 & \lambda_1^t & 0 & 0 \\ 0 & 0 & 0 & \lambda_2^t & 0 \\ 0 & 0 & 0 & 0 & \lambda_3^t \end{bmatrix} \boldsymbol{T}^{-1}$$

Here $\boldsymbol{T}$ represents the transformation required to place the system into Jordan canonical form. It plays the same role that $\boldsymbol{V}$ did for systems in modal form, but we change the notation back to the general $\boldsymbol{T}$ to emphasise that the matrix is not made up of eigenvectors as it was in the modal canonical form case.

7.5.2 Autonomous Evolution in Jordan Formula ($t \in \mathbb{R}_+$)

Let's also consider the evolution of a continuous time system represented by the $k \times k$ non-trivial Jordan block $\boldsymbol{J}$ associated with eigenvalue λ.

$$\boldsymbol{\Phi_J}(t) = \mathcal{L}^{-1}\left\{(s\boldsymbol{I} - \boldsymbol{A_J})^{-1}\right\}$$
$$= \mathcal{L}^{-1}\left\{\begin{bmatrix} s-\lambda & -1 & & & \\ & s-\lambda & -1 & & \\ & & \ddots & \ddots & \\ & & & s-\lambda & -1 \\ & & & & s-\lambda \end{bmatrix}^{-1}\right\}$$
$$= \mathcal{L}^{-1}\left\{\begin{bmatrix} (s-\lambda)^{-1} & (s-\lambda)^{-2} & \cdots & (s-\lambda)^{-k+1} & (s-\lambda)^{-k} \\ & (s-\lambda)^{-1} & (s-\lambda)^{-2} & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & (s-\lambda)^{-1} & (s-\lambda)^{-2} \\ & & & & (s-\lambda)^{-1} \end{bmatrix}\right\}$$

$$= \mathcal{L}^{-1}\left\{(sI-\lambda)^{-1}\boldsymbol{I} + (sI-\lambda)^{-2}\boldsymbol{U}_1 + \cdots + (sI-\lambda)^{-k}\boldsymbol{U}_{k-1}\right\}$$

where $\boldsymbol{U}_1$ is a zero matrix with the elements one above the diagonal set to zero. Similarly $\boldsymbol{U}_i$ has the elements i above the diagonal set to one.

$$= \mathrm{e}^{-\lambda t}\boldsymbol{I} + t\mathrm{e}^{-\lambda t}\boldsymbol{U}_1 + \frac{1}{2}t^2\mathrm{e}^{-\lambda t}\boldsymbol{U}_2 + \cdots + \frac{1}{k-1}t^{k-1}\mathrm{e}^{-\lambda t}\boldsymbol{U}_{k-1}$$

$$\boldsymbol{\Phi_J}(t) = \begin{bmatrix} \mathrm{e}^{-\lambda t} & \frac{1}{2}t^2\mathrm{e}^{-\lambda t} & \cdots & \cdots & \frac{1}{k-1}t^{k-1}\mathrm{e}^{-\lambda t} \\ & \mathrm{e}^{-\lambda t} & \frac{1}{2}t^2\mathrm{e}^{-\lambda t} & \cdots & \frac{1}{k-2}t^{k-2}\mathrm{e}^{-\lambda t} \\ & & \ddots & \ddots & \vdots \\ & & & & \mathrm{e}^{-\lambda t} \end{bmatrix}$$

Here we have used the notation $\boldsymbol{\Phi_J}$ as a reminder that we have placed the system in to Jordan canonical form. If we started in some other representation then we need $\boldsymbol{\Phi}(t) = \boldsymbol{T}\boldsymbol{\Phi_J}(t)\boldsymbol{T}^{-1}$ where $\boldsymbol{T}$ is the mapping used to place the original system into Jordan form.

A system having repeated poles (or eigenvalues) is a necessary, but not sufficient condition for the system's $\boldsymbol{A}$ matrix being non-diagonalisable. A matrix can have repeated eigenvalues and still be diagonalisable. Indeed, some of the repeated eigenvalues might be "part" of a non-trivial Jordan block and others not. For example, a matrix $\boldsymbol{A} \in \mathbb{R}^{4\times 4}$ with four repeated eigenvalues at -2 could have any of the following Jordan forms.

$$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$

The modal structures associated with each of these possibilities will be quite different.

In each case we have four state variable associated with the eigenvalue of -2, but the coupling between those state variables is different in the five cases. For example, a complex system might have a group of independent signals filtered by a back of filters with the same time constant, in which case they would have a trivial Jordan form. Conversely, there might be a complicated mechanical structure in which multiple degrees of freedom are coupled.

Figures 7.3-7.7 show the impulse responses of the various systems when each state is probed with an impulse in turn. Note that the systems with larger Jordan blocks exhibit state variable modes with ever more complex behaviour. This complexity arises from the number of state variables that are coupled together by the ones in the superdiagonal of the Jordan form.

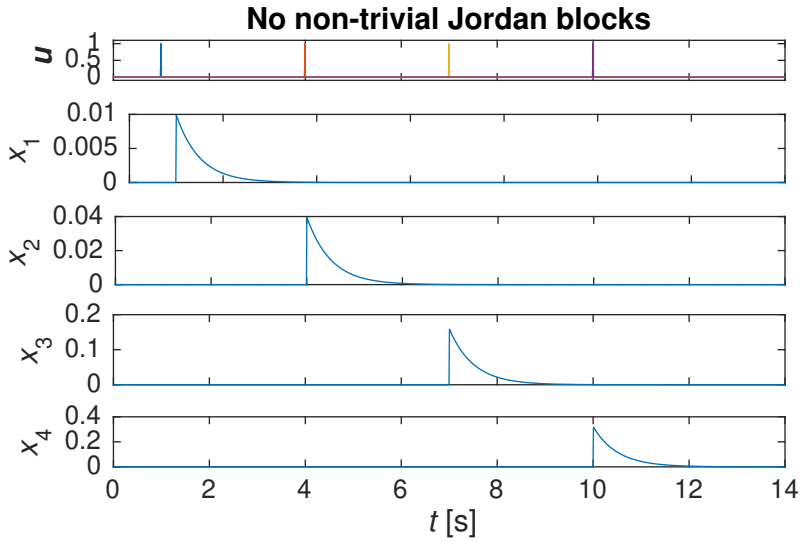


Figure 7.3: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains no non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

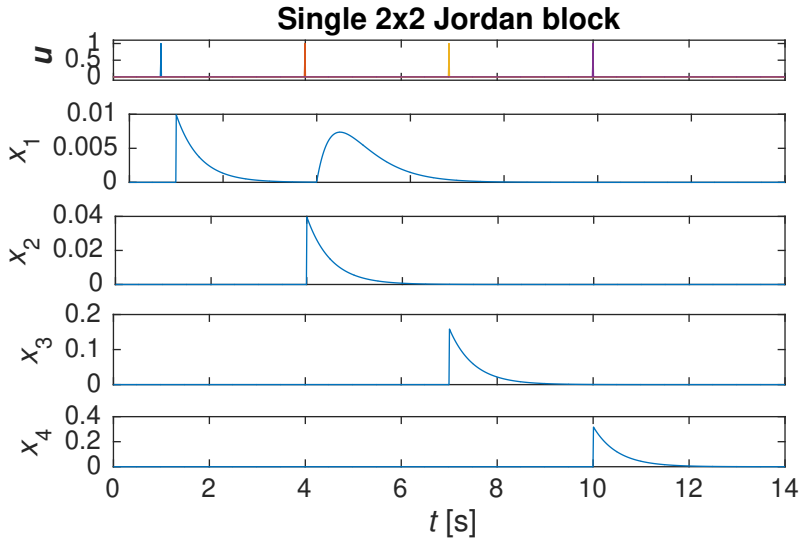


Figure 7.4: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single two-by-two non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

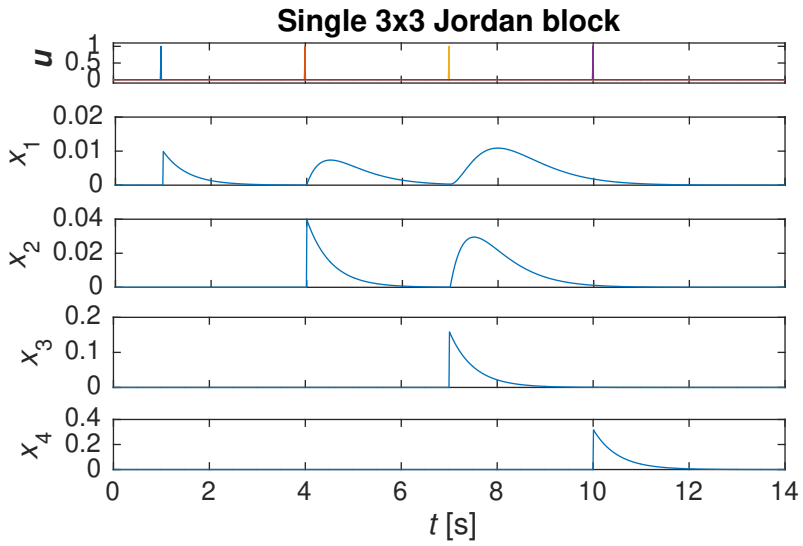


Figure 7.5: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single three-by-three non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

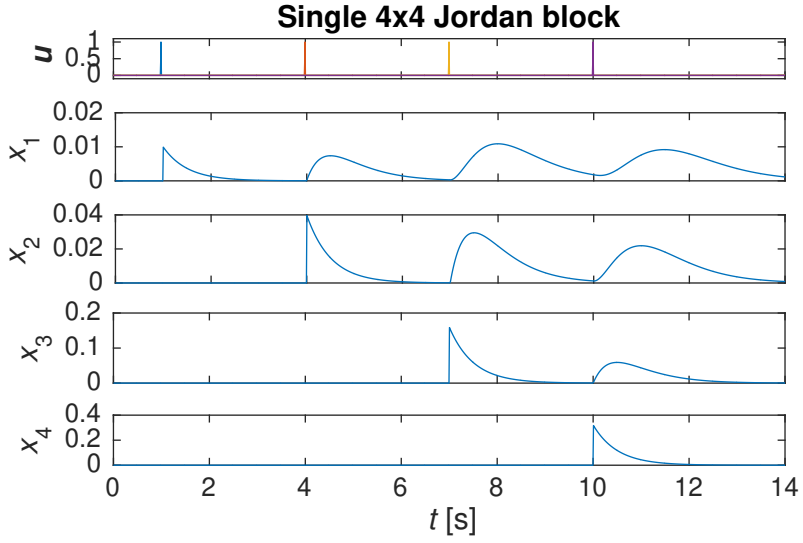


Figure 7.6: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single four-by-four non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

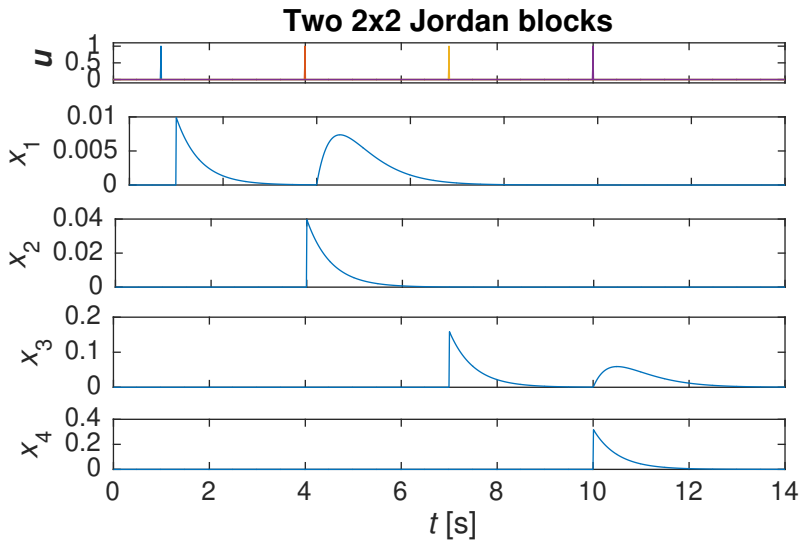


Figure 7.7: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains two two-by-two non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

7.6 Canonical Forms in Matlab

Matlab can conveniently convert to a modal canonical form and will also return the transformation matrix required to perform the conversion. `[Am,Bm,Cm,Dm,P] = canon(A,B,C,D,'modal')`

Matlab can also convert to a so-called *companion* form, which is similar to our observer canonical form (see later). This form is less useful for our purposes. Unfortunately the $\boldsymbol{T}$ matrix in matlab is defined as the inverse of that used in the notes (and almost everywhere else). For us, the state vector $\boldsymbol{x}$ is transformed to $\boldsymbol{z}$ via $\boldsymbol{x} = \boldsymbol{T}\boldsymbol{z}$, but in matlab it is $\boldsymbol{z} = \boldsymbol{T}\boldsymbol{x}$. Matlab's approach seems more intuitive if working on control, but the typical way makes more sense from a linear algebra point of view. Bear this in mind when working on problems.

If you examine the models formed by matlab, you will see that the $\boldsymbol{B}$ and $\boldsymbol{C}$ matrices will not have the very simple forms that we expect for a canonical form. This is because Matlab scales the state variables so to ensure that the model has good numerical properties. By changing $\boldsymbol{B}$ and $\boldsymbol{C}$ the "units" of the state variables can be changed so that the entries in the $\boldsymbol{A}$ matrices do not cover too wide a range. These sorts of numerical stability issues are important when dealing with large systems, but will not be too much of an issue for the systems that we will encounter in this course.

7.7 Appendix – Other Canonical Forms

7.7.1 Converting to modal canonical form

We would like a method to convert from an arbitrary system $\left[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}\right]$ to the modal canonical realisation $\left[\boldsymbol{A}_\mathrm{m}, \boldsymbol{B}_\mathrm{m}, \boldsymbol{C}_\mathrm{m}, \boldsymbol{D}_\mathrm{m}\right]$. As before, we have $\boldsymbol{A}_\mathrm{m} = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} \implies \boldsymbol{V}\boldsymbol{A}_\mathrm{m} = \boldsymbol{A}\boldsymbol{V}$, where $\boldsymbol{V}$ is the transformation matrix that we wish to find. We first let $\boldsymbol{v}_i$ be the i-th column of $\boldsymbol{V}$.

$$\begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{bmatrix} = \boldsymbol{A}\begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3\end{bmatrix}$$
$$\lambda_i\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{v}_i$$

You hopefully recognise this as an eigenvalue problem, with λ_i as the eigenvalues and $\boldsymbol{v}_i$ the corresponding eigenvectors. That is finding the transformation matrix $\boldsymbol{V}$ just requires us to find the eigenvectors of $\boldsymbol{A}$.

As an example, consider $\boldsymbol{A} = \begin{bmatrix}-7 & -12\\ 1 & 0\end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix}1\\ 0\end{bmatrix}$, $\boldsymbol{C} = \begin{bmatrix}1 & 2\end{bmatrix}$ and $\boldsymbol{D} = 0$. Let's find the $\boldsymbol{V}$ matrix that can be used to convert the system to modal canonical form. Start by finding the eigenvalues and eigenvectors of $\boldsymbol{A}$. Let λ_i be the eigenvalues and $\boldsymbol{v}_i = \begin{bmatrix}v_1 & v_2\end{bmatrix}^\mathsf{T}$ be the corresponding eigenvectors. So,

$$\left|\lambda\boldsymbol{I} - \boldsymbol{A}\right| = 0 \implies \begin{vmatrix}\lambda + 7 & 12\\ -1 & \lambda\end{vmatrix} = 0$$
$$\lambda(\lambda + 7) + 12 = 0$$
$$\lambda^2 + 7\lambda + 12 = 0$$
$$(\lambda + 3)(\lambda + 4) = 0$$
$$\lambda = -3, -4$$

To find the eigenvectors we substitute each eigenvalue into $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ and solve.

$$\begin{bmatrix}-7 & -12\\ 1 & 0\end{bmatrix}\boldsymbol{v}_i = \lambda_i\boldsymbol{v}_i$$
$$\text{So, } \begin{cases}-7v_1 - 12v_2 = \lambda v_1\\ v_1 = \lambda v_2\end{cases}$$

The second of these equations tells us that the two eigenvectors are of form $\begin{bmatrix}-3\alpha\\ \alpha\end{bmatrix}$ and $\begin{bmatrix}-4\beta\\ \beta\end{bmatrix}$, with α and β being as yet undetermined scalars. That is, to avoid confusion we have set $v_2 = \alpha$ for the

first eigenvector and $v_2 = \beta$ for the second. Recall that the eigenvectors form the columns of our transformation matrix $\boldsymbol{V}$, so we can write $\boldsymbol{V} = \begin{bmatrix} -3\alpha & -4\beta \\ \alpha & \beta \end{bmatrix}$

We must now choose appropriate values for α and β. Recall that in modal canonical form we want all elements of $\boldsymbol{B}_\mathrm{m}$ equal to one. We also know that $\boldsymbol{V}\boldsymbol{B}_\mathrm{m} = \boldsymbol{B}$, so we have

$$\begin{bmatrix} -3\alpha & -4\beta \\ \alpha & \beta \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$\implies -3\alpha - 4\beta = 1 \text{ and } \alpha + \beta = 0$$

Substituting $-\alpha$ for β, we thus get

$$\begin{aligned} -3\alpha + 4\alpha &= 1 \\ \alpha &= 1 \\ \implies \beta &= -1 \end{aligned}$$

Thus we have our transformation matrix $\boldsymbol{V} = \begin{bmatrix} -3 & 4 \\ 1 & -1 \end{bmatrix}$.

If we use the fact that $\boldsymbol{V}^{-1} = \begin{bmatrix} 1 & 4 \\ 1 & 3 \end{bmatrix}$ we can calculate the complete set of transformed matrices:

$$\begin{aligned} \boldsymbol{A}_\mathrm{m} = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} &= \begin{bmatrix} 1 & 4 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -7 & -12 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -3 & 4 \\ 1 & -1 \end{bmatrix} &&= \begin{bmatrix} -3 & 0 \\ 0 & -4 \end{bmatrix} \\ \boldsymbol{B}_\mathrm{m} = \boldsymbol{V}^{-1}\boldsymbol{B} &= \begin{bmatrix} 1 & 4 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} &&= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \boldsymbol{C}_\mathrm{m} = \boldsymbol{C}\boldsymbol{V} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} -3 & 4 \\ 1 & -1 \end{bmatrix} &&= \begin{bmatrix} -1 & -2 \end{bmatrix} \\ \boldsymbol{D}_\mathrm{m} = \boldsymbol{D} &&&= \begin{bmatrix} 0 \end{bmatrix} \end{aligned}$$

As expected the result is now in modal canonical form.

In addition to the modal and Jordan canonical forms discussed above, there are a couple of other canonical forms that are sometimes useful for particular purposes. Two forms that are encountered on occasions are

- Control canonical form,
- Observer canonical form.

These forms are very convenient when designing compensators or observers respectively. We will see this later, but given that most design is now done with computer tools, the forms are not so important for us.

7.7.2 From transfer function to control canonical form

A SISO transfer function

$$G = \frac{n_n s^n + n_{n-1}s^{n-1} + \cdots + n_1 s + n_0}{s^n + d_{n-1}s^{n-1} + \cdots + d_1 s + d_0}$$

is represented by the system with form $\dot{\boldsymbol{x}} = \boldsymbol{A}_\mathrm{c}\boldsymbol{x} + \boldsymbol{B}_\mathrm{c}\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}_\mathrm{c}\boldsymbol{x} + \boldsymbol{D}_\mathrm{c}\boldsymbol{u}$, where

$$\boldsymbol{A}_\mathrm{c} = \begin{bmatrix} -d_{n-1} & -d_{n-2} & \cdots & -d_1 & -d_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \boldsymbol{B}_\mathrm{c} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\boldsymbol{C}_\mathrm{c} = \begin{bmatrix} n_{n-1} - d_{n-1}n_n \vdots n_{n-2} - d_{n-2}n_n \vdots \cdots \vdots n_1 - d_1 n_n \vdots n_0 - d_0 n_n \end{bmatrix}$$

$$\text{and } \boldsymbol{D}_\text{c} = n_n.$$

In the common case where $n_n = 0$, then

$$G = \frac{n_{n-1}s^{n-1} + \cdots + n_1 s + n_0}{s^n + d_{n-1}s^{n-1} + \cdots + d_1 s + d_0}$$

and we can simplify our matrices to get

$$\boldsymbol{A}_\text{c} = \begin{bmatrix} -d_{n-1} & -d_{n-2} & \cdots & -d_1 & -d_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \boldsymbol{B}_\text{c} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\boldsymbol{C}_\text{c} = \begin{bmatrix} n_{n-1} & n_{n-2} & \cdots & n_1 & n_0 \end{bmatrix}$$

$$\text{and } \boldsymbol{D}_\text{c} = 0.$$

Convert the transfer function $G(s) = \dfrac{s+3}{s^2+3s+2}$ to state space.

Now $G = \dfrac{n_2 s^2 + n_1 s + n_0}{s^2 + d_1 s + d_0}$, with $n_0 = 3$, $n_1 = 1$, $n_2 = 0$, $d_1 = 3$ and $d_0 = 2$.

Hence controllable canonical form:

$$\begin{aligned} \boldsymbol{A}_\text{c} &= \begin{bmatrix} -d_1 & -d_0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -3 & -2 \\ 1 & 0 \end{bmatrix} \\ \boldsymbol{B}_\text{c} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \boldsymbol{C}_\text{c} &= \begin{bmatrix} n_1 - d_1 n_2 & n_0 - d_0 n_2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \end{bmatrix} \\ \boldsymbol{D}_\text{c} &= 0 \end{aligned}$$

7.7.3 Converting to control canonical form

We would like to have a general method for converting an arbitrary state space system into control canonical form. To do this we must find an appropriate transformation matrix, $\boldsymbol{T}$, to do the basis conversion.

Let $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ describe the system that we wish to transform into the control canonical form $(\boldsymbol{A}_\text{c}, \boldsymbol{B}_\text{c}, \boldsymbol{C}_\text{c}, \boldsymbol{D}_\text{c})$.

We know that $\boldsymbol{A}_\text{c} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} \implies \boldsymbol{A}_\text{c}\boldsymbol{T}^{-1} = \boldsymbol{T}^{-1}\boldsymbol{A}$

Now, we know something about the structure of $\boldsymbol{A}_\text{c}$ because it is in control canonical form. That allows us to simplify the equation.

Without loss of generality, let's assume a third order system. Let $\boldsymbol{T}^{-1}$ be a matrix having rows $\boldsymbol{p}_1^\mathsf{T}$,$\boldsymbol{p}_2^\mathsf{T}$ and $\boldsymbol{p}_3^\mathsf{T}$ (where in this case $\boldsymbol{p}_i^\mathsf{T}$ are 1×3 vectors). That is,

$$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T} \\ \boldsymbol{p}_2^\mathsf{T} \\ \boldsymbol{p}_3^\mathsf{T} \end{bmatrix}$$

Then

$$\begin{bmatrix} -d_2 & -d_1 & -d_0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T} \\ \boldsymbol{p}_2^\mathsf{T} \\ \boldsymbol{p}_3^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T}\boldsymbol{A} \\ \boldsymbol{p}_2^\mathsf{T}\boldsymbol{A} \\ \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A} \end{bmatrix}$$

We can calculate the third then second rows to find that

$$\begin{aligned}\boldsymbol{p}_2^\mathsf{T} &= \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}\\ \boldsymbol{p}_1^\mathsf{T} &= \boldsymbol{p}_2^\mathsf{T}\boldsymbol{A} = \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}^2\end{aligned}$$

We also know that $\boldsymbol{T}^{-1}\boldsymbol{B} = \boldsymbol{B_c}$, so

$$\begin{bmatrix}\boldsymbol{p}_1^\mathsf{T}\boldsymbol{B}\\ \boldsymbol{p}_2^\mathsf{T}\boldsymbol{B}\\ \boldsymbol{p}_3^\mathsf{T}\boldsymbol{B}\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix} \implies \boldsymbol{p}_1^\mathsf{T}\boldsymbol{B} = 1, \boldsymbol{p}_2^\mathsf{T}\boldsymbol{B} = 0 \text{ and } \boldsymbol{p}_3^\mathsf{T}\boldsymbol{B} = 0$$

Substituting for $\boldsymbol{p}_1^\mathsf{T}$ and $\boldsymbol{p}_2^\mathsf{T}$ gives us the following equations:

$$\begin{aligned}\boldsymbol{p}_3^\mathsf{T}\boldsymbol{B} &= 0\\ \boldsymbol{p}_2^\mathsf{T}\boldsymbol{B} = \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}\boldsymbol{B} &= 0\\ \boldsymbol{p}_1^\mathsf{T}\boldsymbol{B} = \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}^2\boldsymbol{B} &= 1\end{aligned}$$

Or alternatively,

$$\begin{aligned}\boldsymbol{p}_3^\mathsf{T}\begin{bmatrix}\boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B}\end{bmatrix} &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}\\ \boldsymbol{p}_3^\mathsf{T} &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}\boldsymbol{\mathcal{C}}^{-1}\end{aligned}$$

where $\boldsymbol{\mathcal{M}}_\mathrm{c} = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B}\end{bmatrix}$ is known as the *controllability matrix* for reasons that we will see later.

But recall that we already have equations for $\boldsymbol{p}_1^\mathsf{T}$ and $\boldsymbol{p}_2^\mathsf{T}$ in terms of $\boldsymbol{p}_3^\mathsf{T}$, so we now know all of $\boldsymbol{T}$.

$$\boldsymbol{T}^{-1} = \begin{bmatrix}\boldsymbol{p}_1^\mathsf{T}\\ \boldsymbol{p}_2^\mathsf{T}\\ \boldsymbol{p}_3^\mathsf{T}\end{bmatrix} = \begin{bmatrix}\boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}^2\\ \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}\\ \boldsymbol{p}_3^\mathsf{T}\end{bmatrix}$$

where $\boldsymbol{p}_3^\mathsf{T}$ is given by the last row of the inverse of the controllability matrix.

We can now use $\boldsymbol{T}^{-1}$ to convert to control canonical form.

1. Calculate the controllability matrix,

$$\boxed{\boldsymbol{\mathcal{M}}_\mathrm{c} = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B}\cdots & \boldsymbol{A}^{n-1}\boldsymbol{B}\end{bmatrix}}$$

   where $\boldsymbol{A}^2 = \boldsymbol{A}\boldsymbol{A}$ etc. and n is the number of states.

2. Find the inverse of $\boldsymbol{\mathcal{M}}_\mathrm{c}$ using any appropriate method.

3. Calculate the last row of $\boldsymbol{T}$ as $\boldsymbol{p}_n^\mathsf{T} = \begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}\boldsymbol{\mathcal{M}}_\mathrm{c}^{-1}$

4. Construct the transformation matrix $\boldsymbol{T}^{-1} = \begin{bmatrix}\boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}^{n-1}\\ \boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}^{n-2}\\ \vdots\\ \boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}\\ \boldsymbol{p}_n^\mathsf{T}\end{bmatrix}$

5. Use $\boldsymbol{T}^{-1}$ to find the system matrices in control canonical form.

7.7.4 From transfer function to observer canonical form

A SISO transfer function

$$G = \frac{n_n s^n + n_{n-1}s^{n-1} + \cdots + n_1 s + b_0}{s^n + d_{n-1}s^{n-1} + \cdots + d_1 s + d_0}$$

is represented by the system with form $\dot{\boldsymbol{x}} = \boldsymbol{A}_\text{o}\boldsymbol{x} + \boldsymbol{B}_\text{o}\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}_\text{o}\boldsymbol{x} + \boldsymbol{D}_\text{o}\boldsymbol{u}$, where

$$\boldsymbol{A}_\text{o} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -d_0 \\ 1 & 0 & \cdots & 0 & -d_1 \\ 0 & 1 & \cdots & 0 & -d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -d_{n-1} \end{bmatrix}, \boldsymbol{B}_\text{o} = \begin{bmatrix} n_0 - d_0 n_n \\ n_1 - d_1 n_n \\ n_2 - d_2 n_n \\ \vdots \\ n_{n-1} - d_{n-1} n_n \end{bmatrix}$$

$$\boldsymbol{C}_\text{o} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{D}_\text{o} = n_n.$$

In the common case where $n_n = 0$, then

$$G = \frac{n_{n-1}s^{n-1} + \cdots + n_1 s + n_0}{s^n + d_{n-1}s^{n-1} + \cdots + d_1 s + d_0}$$

and we can simplify our matrices to get

$$\boldsymbol{A}_\text{o} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -d_0 \\ 1 & 0 & \cdots & 0 & -d_1 \\ 0 & 1 & \cdots & 0 & -d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -d_{n-1} \end{bmatrix}, \boldsymbol{B}_\text{o} = \begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ \vdots \\ n_{n-1} \end{bmatrix}$$

$$\boldsymbol{C}_\text{o} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{D}_\text{o} = 0.$$

$$G(s) = \frac{s+3}{s^2+3s+2} = \frac{n_2 s^2 + n_1 s + n_0}{s^2 + d_1 s + d_0}$$

with $n_0 = 3$, $n_1 = 1$, $n_2 = 0$, $d_1 = 3$ and $d_0 = 2$. Hence observer canonical form:

$$\begin{aligned} \boldsymbol{A}_\text{o} &= \begin{bmatrix} 0 & -d_0 \\ 1 & -d_1 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix} \\ \boldsymbol{B}_\text{o} &= \begin{bmatrix} n_0 - d_0 n_2 \\ n_1 - d_1 n_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\ \boldsymbol{C}_\text{o} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ \boldsymbol{D}_\text{o} &= 0 \end{aligned}$$

7.7.5 A comment on $n_n \neq 0$

In the above discussion we have allowed for a non-zero value of n_n. This gives us a nice general form for the state space matrices, but makes them a little messy. In reality we often have $n_n = 0$, so the degree of the numerator polynomial is less than that of the denominator. We call such transfer functions "strictly proper". In such systems we have $\boldsymbol{D} = 0$ and the terms in the state space matrices are simplified. If have a transfer function that is proper (the degree of the numerator and denominator polynomials is the same), then you can always break it into two parts; a constant plus a strictly proper part.

For example, we can break the proper transfer function $\dfrac{s^2+3s+3}{s^2+2s+1}$ into the sum of a constant and a strictly proper transfer function:

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

We can then form any of our state space representations by setting $\boldsymbol{D}$ equal to the constant (1 in this case) and proceeding as normal to extract $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ from the strictly proper part. So for this example, we could write the a control canonical form representation as

$$\boldsymbol{A} = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 2 \end{bmatrix}, \text{ and } \boldsymbol{D} = 1.$$