ECEN415
Advanced Control Engineering

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

Trimester 2, 2017
Revised July 24, 2017

Contents

1 Introduction

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. We will also generalise our approach to consider problems that exist in discrete rather than continuous time. In particular we will look at

- Systems having multiple outputs that we can control with multiple inputs;
- Systems that are affected by disturbances and measurement noise;
- *Optimal* controllers, which allow us to be confident that we have designed the best possible controller.

Conceptually this follows from ECEN315, 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.

1.1 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 $\{u_1, u_2\}$ and two outputs $\{y_1, y_2\}$ as shown in figure 1.1. We could describe the input-output relations of this system 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. It is sometimes the case that only a single input affects each output. These systems

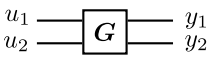


Figure 1.1: A system described by a matrix of transfer functions ($\boldsymbol{G}$) which has two inputs (u_1, u_2) and two outputs (y_1, y_2).

are essentially a set of uncoupled SISO systems and we could proceed to design multiple compensators using classical control techniques. See for example figure 1.2, 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 1.3 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.

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

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 1.6, which shows the time response and figure 1.7 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 1.8. This has the virtue of simplicity, and will sometimes work fine.

In the example shown in figure 1.8 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

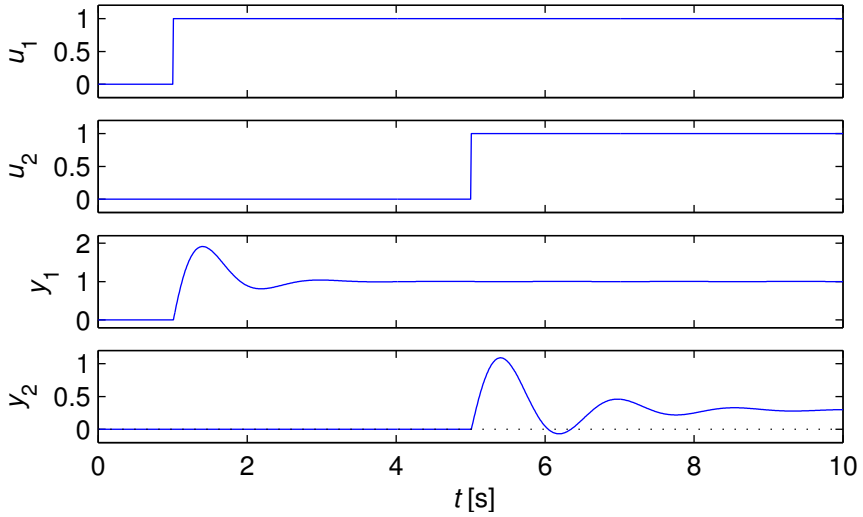


Figure 1.2: Step response of a two-input two-output system with uncoupled dynamics.

$C_{22}(s)$ 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 1.9.

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.

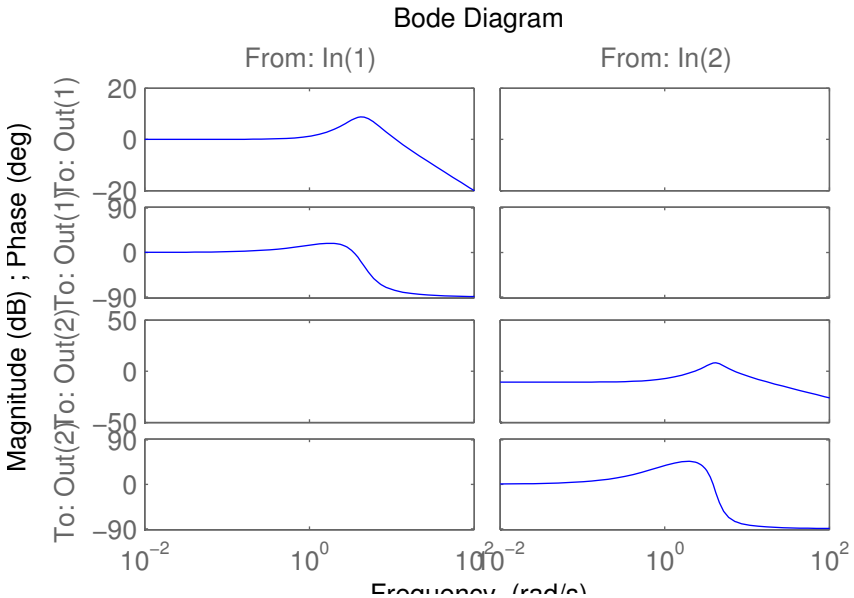


Figure 1.3: Bode plots of a two-input two-output system with uncoupled dynamics. Each Bode plot shows a frequency response that indicates the effect that a particular input has on a particular output.

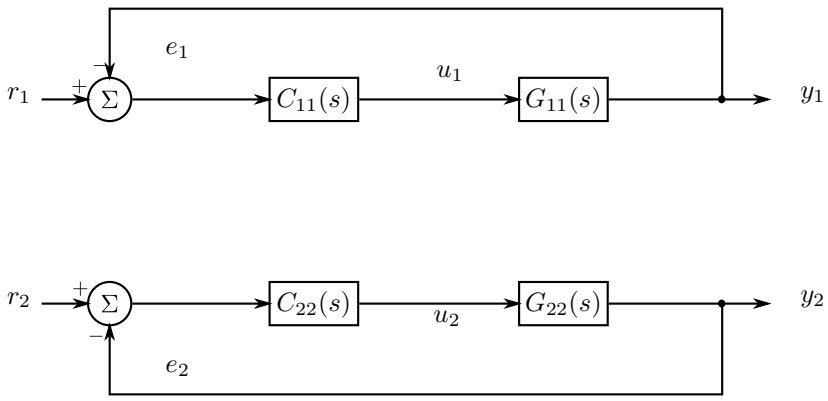


Figure 1.4: Schematic of a two-input, two-output system with feedback control that ignores the cross coupling.

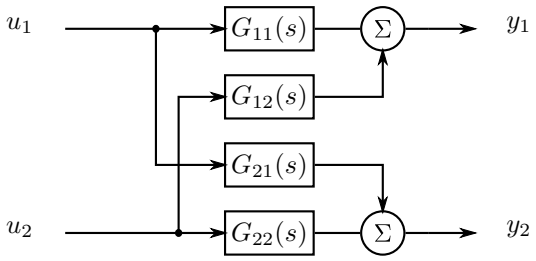


Figure 1.5: Schematic of a two-input, two-output coupled system.

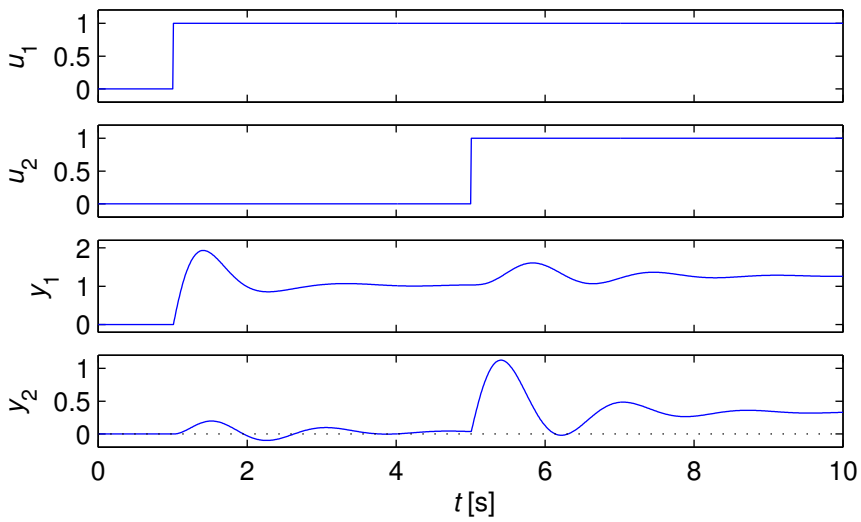


Figure 1.6: Step response of a two-input two-output system with coupled dynamics.

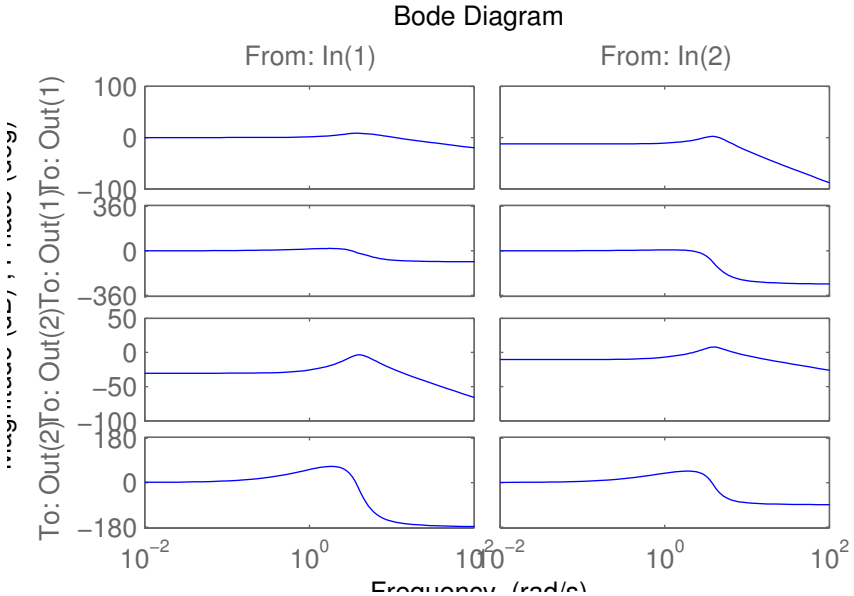


Figure 1.7: Bode plots of a two-input two-output system with coupled dynamics.

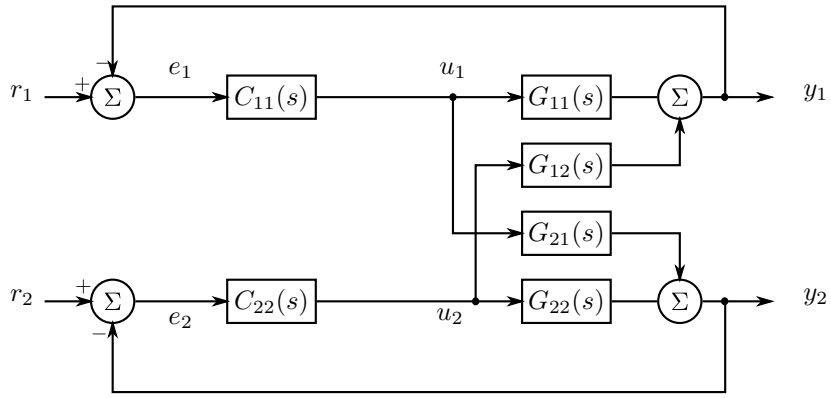


Figure 1.8: Schematic of a two-input, two-output system with feedback control that ignores the cross coupling.

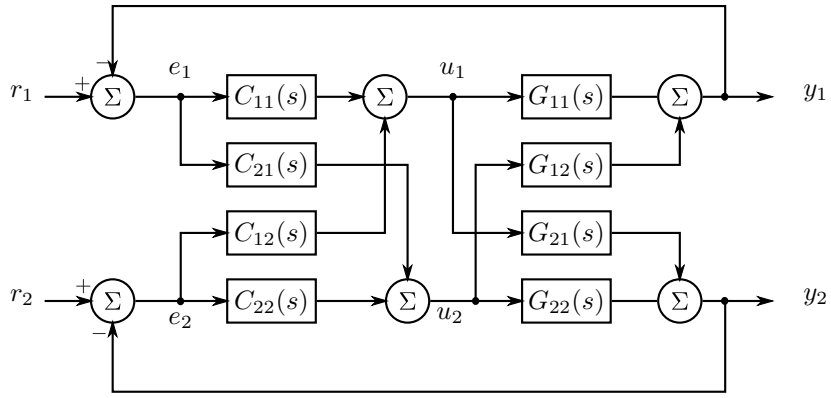


Figure 1.9: Schematic of a two-input, two-output system with complete feedback control.

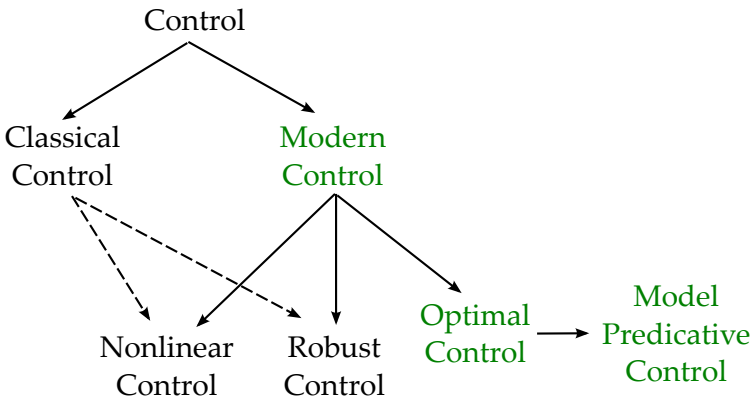


Figure 1.10: Dependencies in subfields of control engineering.

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 1.10 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.

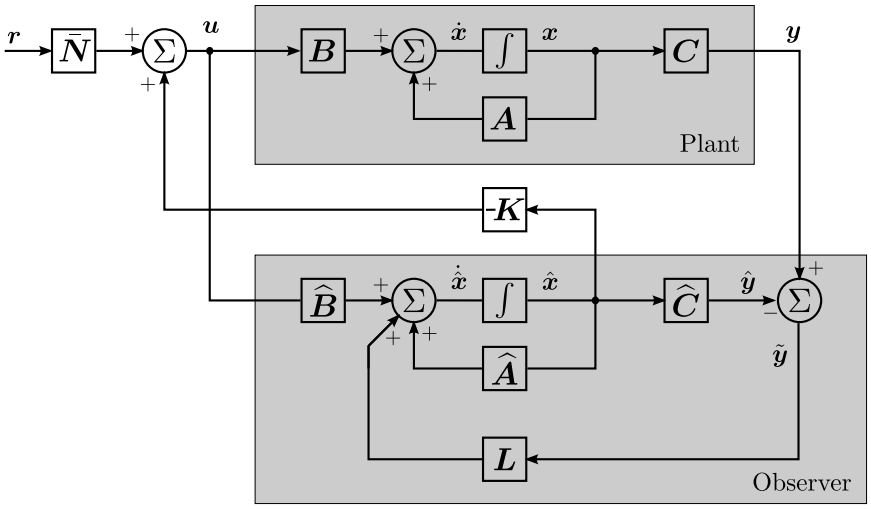


Figure 1.11: A state space system, including the plant model, estimated state feedback taken from a Luenberger observer and a prefilter $\bar{N}$ to set the desired dc gain.

- 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 and Kalman filters.
9. Nonlinear systems or MPC or digital controller implementation or robust control or system identification or loop shaping or ...

2 Classical Control

2.1 Classical Control in Continuous Time

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

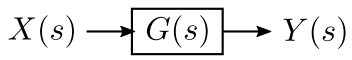


Figure 2.1: A transfer function representation of a system.

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

We have dealt exclusively with transfer functions that are rational functions. 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. 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.

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

Remember that there will be exactly n poles, though they may not be distinct.

Three main types of poles arise when solving the characteristic equation: real poles, conjugate pairs of complex poles and repeated poles. Recall that different pole configurations in a transfer functions correspond to a familiar set of possible system modes.

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

The general properties of the modes can be inferred from the position of the corresponding position of the poles in the s-plane. Figure 2.2 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.

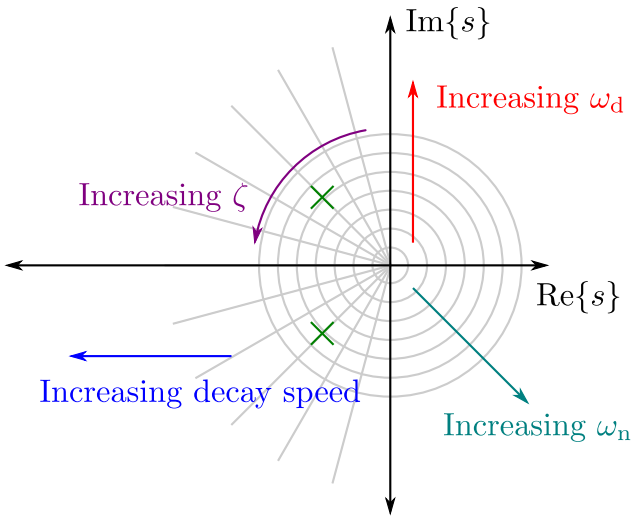


Figure 2.2: Qualitative behaviour of modes corresponding to a pair of complex conjugate poles in the s-plane.

We are often given specifications on both the settling time and damping that a system must exhibit. We can therefore designate a region of the s-plane within which we need to place the system poles. A typical example of this is shown in figure 2.3.

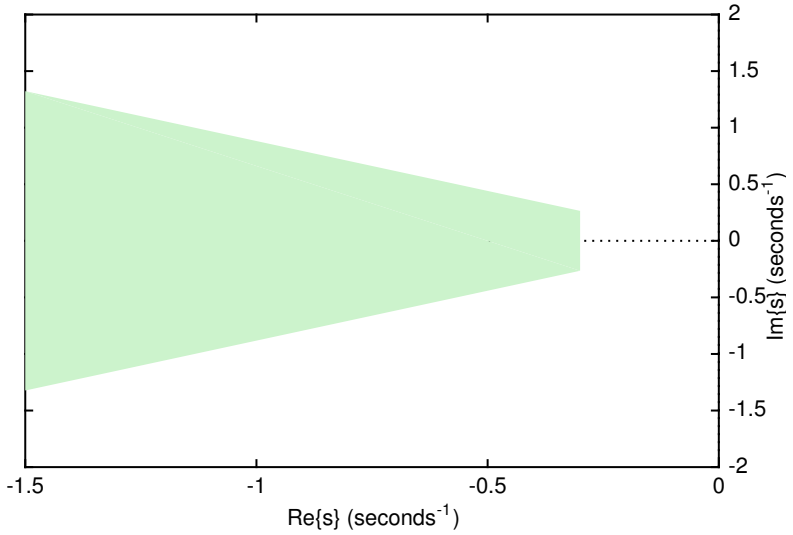


Figure 2.3: Acceptable dominant pole locations for a system having both a settling time and a damping constraint.

The mechanism we use to alter the position of the system poles is *feedback*. Figure 2.4 shows the simplest arrangement of a feedback control system. We can then calculate the loop transfer function of the closed loop system, $T(s)$.

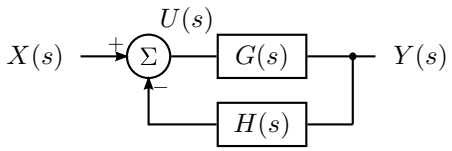


Figure 2.4: A simple feedback system, using a feedback compensator with transfer function $H(s)$.

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

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

For example, we can move the poles of a dc motor having $G(s) = \dfrac{1}{s(s+1)}$ and proportional feedback as shown in figure 2.5. The path taken by the closed loop pole locations as we vary the feedback gain

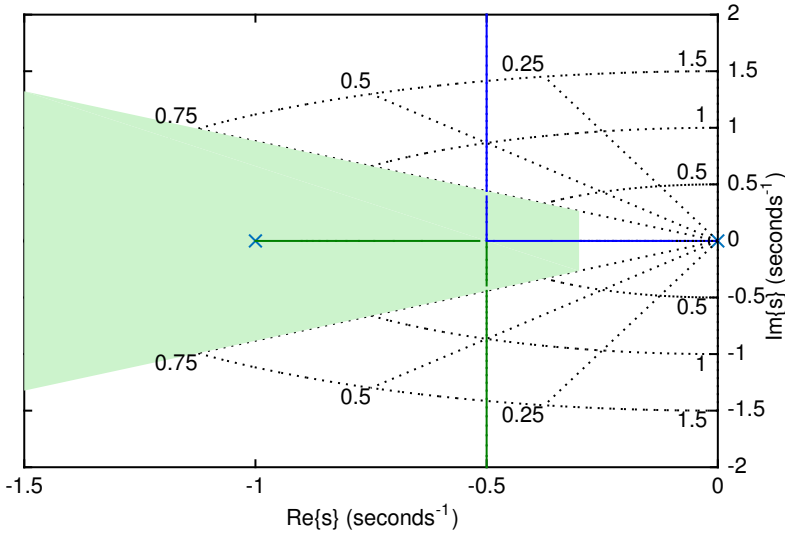


Figure 2.5: Root Locus diagram for a system $G(s) = \dfrac{1}{s(s+1)}$

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 (ie, use a compensator more complex than a proportional gain).

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.

2.2 Classical Control in Discrete Time

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 not do any significant work with the z-transform directly. For the purposes of this course, the important thing to understand is the nature of the z-plane, and particularly the meaning of various pole locations within it. 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.

2.2.1 The z transform

Recall that we made extensive use of the Laplace transform in continous time control system work becuase it allowed us to deal with derivatives.

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

Imagine now that we approximate this differential equation by a difference equation, where we look at how things change between two moments that are separated by one unit of time.

$$y(t+1) - y(t) = 2x(t)$$

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

$$\begin{aligned}&\overset{\mathcal{Z}}{\Longleftrightarrow} zY(z) - Y(z) = 2X(z) \\ G(z) := \frac{Y(z)}{X(z)} &= \frac{2}{z-1}\end{aligned}$$

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.

2.2.2 The z plane

Recall that there is a relationship between locations on the s and z planes, specifically $z = \mathrm{e}^{st_\mathrm{s}}$, 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 this would be at

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

This is shown schematically in figure 2.6.

We know that CT systems are stable iff all of their poles are on the left side 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 $|\lambda_z| > 1$. The correspondence between the regions of stability (and otherwise) in the s and z planes are shown in figure 2.7.

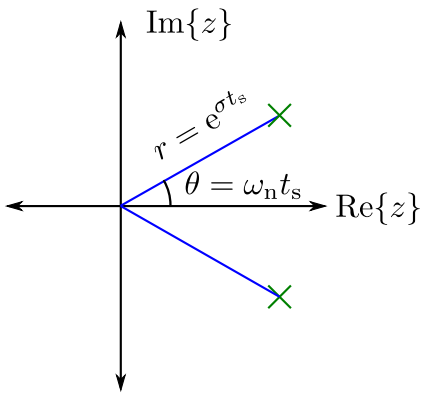


Figure 2.6: Pole locations on the z-plane.

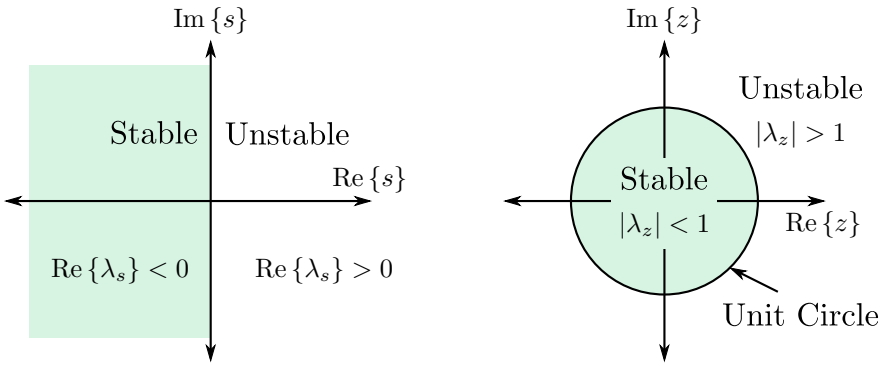


Figure 2.7: Regions of stability on the s and z planes.

Examples for the step responses of both stable and unstable first order discrete time systems can be seen in figures 2.8 and 2.9. 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 2.8, resulting in step responses that settle exponentially to one. In contrast, in figure 2.9 the poles lie outside the unit circle, so the resulting modes grow exponentially.

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

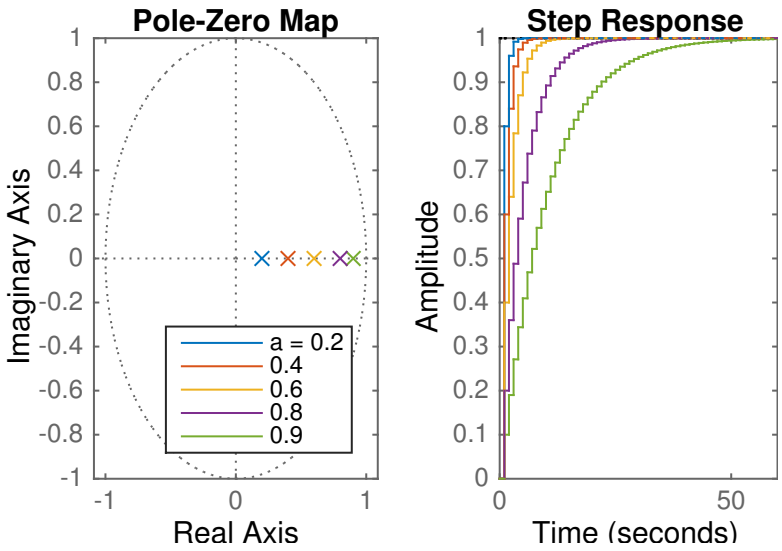


Figure 2.8: Stable modes corresponding to real positive pole locations in the z-plane.

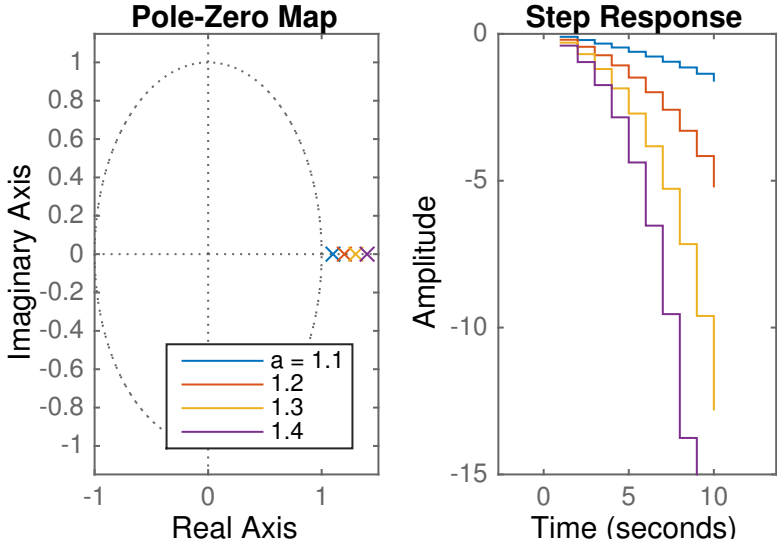


Figure 2.9: Unstable modes corresponding to real positive pole locations in the z-plane.

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 2.10.

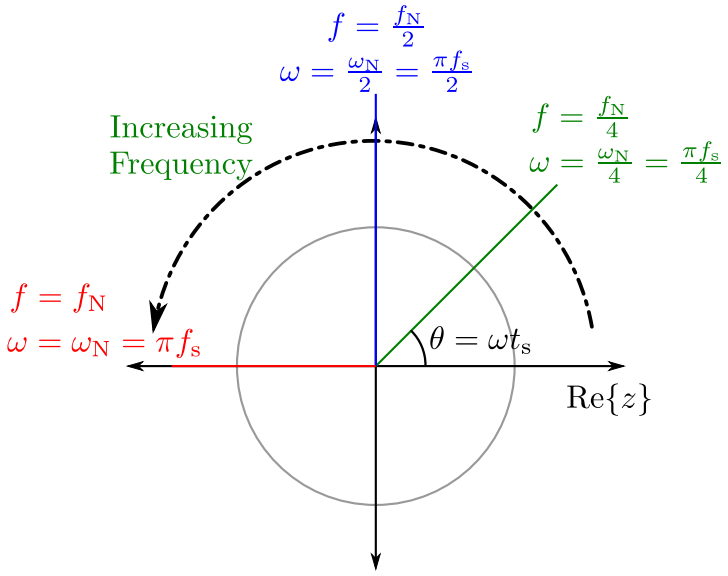


Figure 2.10: Frequency on the z-plane.

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 2.11 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 $G(z) = \dfrac{1}{(z-a)(z-a^T)}$ with $a \in \mathbb{C}$ as given in the figure legend. In all cases $|a| = 0.9$ to ensure stability.

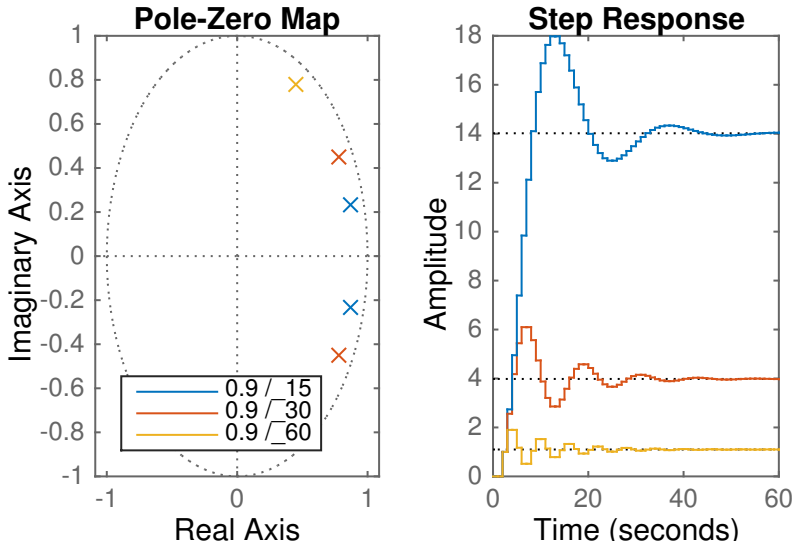


Figure 2.11: Stable modes corresponding to complex pole locations in the z-plane.

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 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 poles live in a narrow wedge around the positive real axis of the z-plane as shown in figure 2.12.

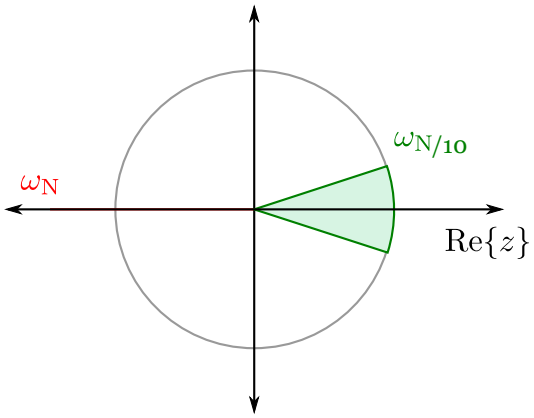


Figure 2.12: Region of the z-plane likely to contain most poles in a closed loop control system.

2.2.3 Aliasing

As should be evident from figure 2.10, 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.

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.

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

This is indistinguishable from a pair of poles lying at $\mathrm{e}^{\pm \mathrm{j}\frac{3}{8}}$. 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 2.13 which illustrates that the mapping from the s to the z-plane is multivauled. 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.

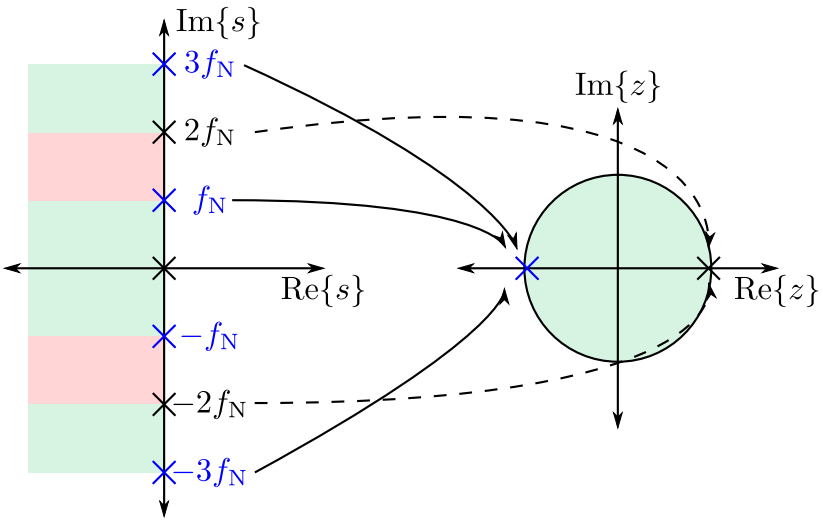


Figure 2.13: Aliasing of the mapping from the s to the z-plane.

2.2.4 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 $s = -\zeta\omega_\mathrm{n} \pm \mathrm{j}\omega_\mathrm{n}\sqrt{1-\zeta^2}$, or equivalently at $s = -\zeta\omega_\mathrm{n} \pm \mathrm{j}\omega_\mathrm{d}$ We can again find the corresponding locations of the poles in the z-plane.

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

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

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

If we express the pole location as $r\angle\theta$ then we have

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

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

Equating the two, we find

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

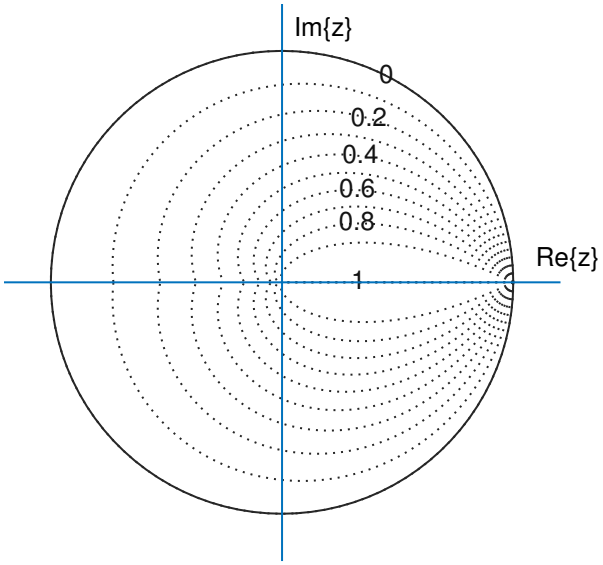


Figure 2.14: Lines of constant damping ratio.

Figure 2.14 shows the complicated curves traced by lines of constant damping ratio.

Figure 2.15 shows the an example of such a region.

As was the case in continuous time, the presence of damping perturbs the frequency of a mode. This again results in a damped frequency that is lower than the natural frequency. Figure 2.16 shows the resulting lines of constant damped frequency.

Because the mapping from the s-plane to the z-plane is conformal, the lines of constant damping ratio and constant damped frequency remain perpendicular everywhere. Figure 2.17 shows the relationship between the two sets of curves.

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 of the z-plane will allow you to find appropriate locations in the z-plane. An example is shown in figure 2.18.

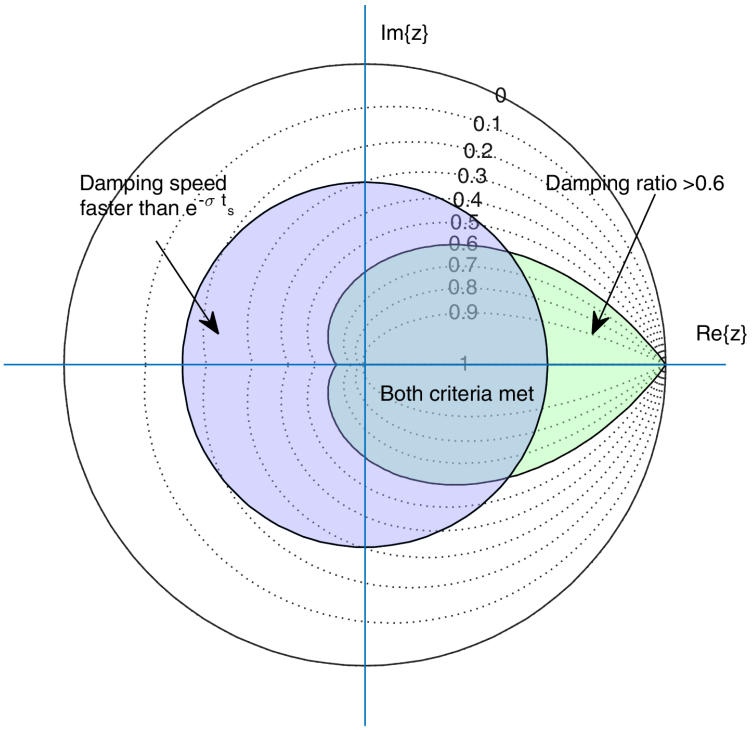


Figure 2.15: A hypothetical example of an acceptable region for the locations of closed loop poles.

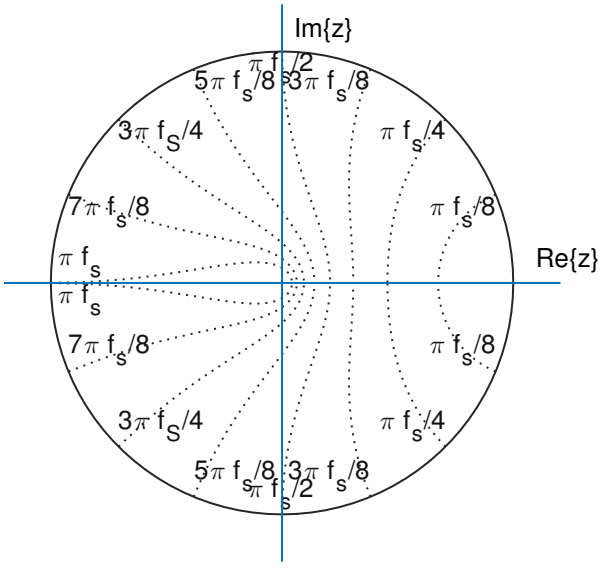


Figure 2.16: Lines of constant natural frequency are shown dotted on the diagram.

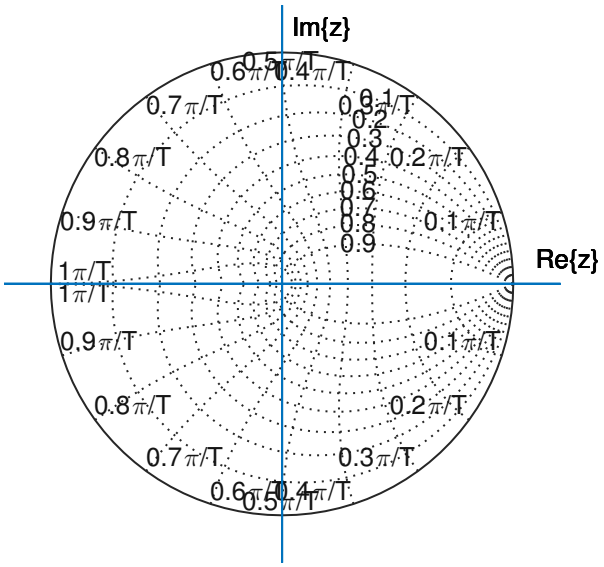


Figure 2.17: Lines of constant damping ratio and constant natural frequency are shown dotted on the diagram.

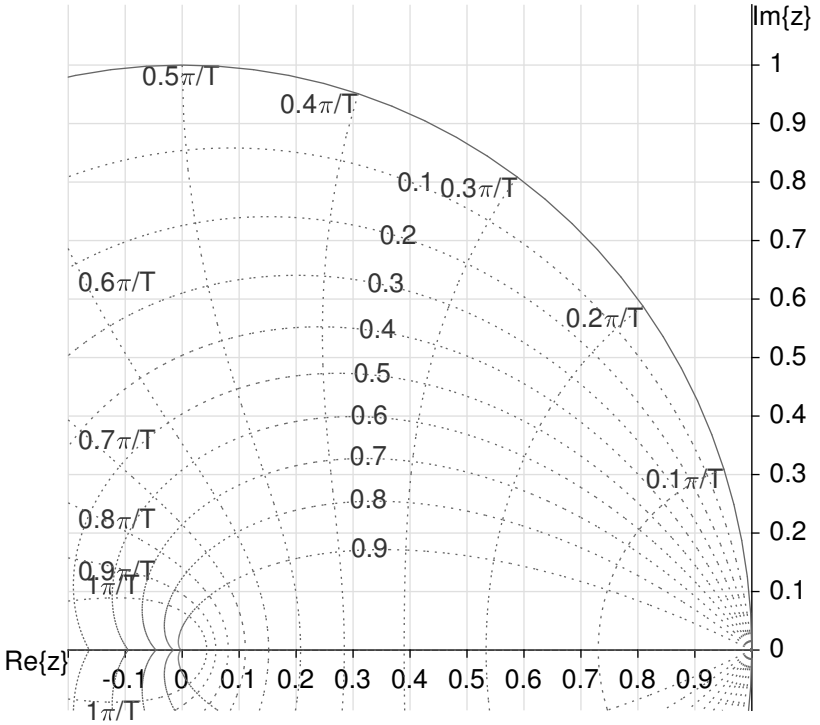


Figure 2.18: Closeup of lines of constant damping ratio and constant natural frequency are shown dotted on the diagram. The solid lines show the real and imaginary components of the z-plane locations.

2.2.5 Special points in the z plane

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

2.2.6 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 things. 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 notice.

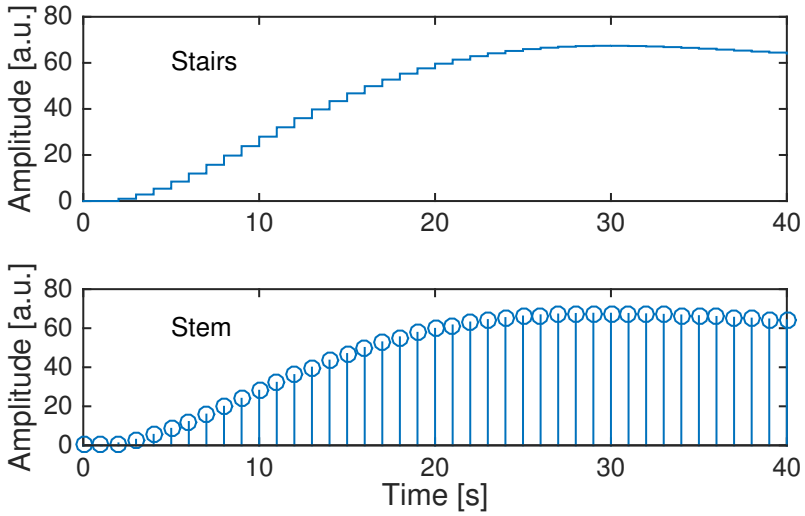


Figure 2.19: Two representations of the time responses of a discrete time system produced by Matlab.

Matlab will conventionally show the response of discrete time systems using a stair plot, as shown in figure 2.19. 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 ($t \in \mathbb{Z}$). Strictly speaking we should use `stem` plots for such systems. However, stem plots often look cluttered which is probably why Matlab's control toolbox doesn't produce them by default. We will therefore tend to use stair plots in practice. 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.

3 A Review of Linear Algebra

3.1 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 $\boldsymbol{x} \in \mathbb{R}^2$.

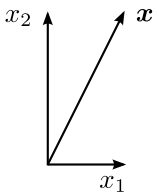


Figure 3.1: A vector $\boldsymbol{x} \in \mathbb{R}^2$.

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;

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

3.2 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 also see matrices with complex numbers and matrices with rational functions (transfer functions) 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.

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

3.2.1 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 3.2.

3.2.2 The Identity matrix

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

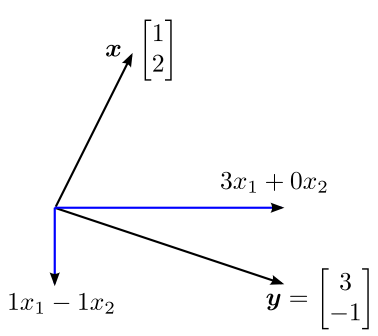
$$\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x}, \quad \forall \boldsymbol{x}$$

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

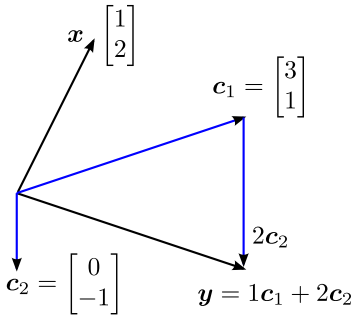
$$\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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 3.2: 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.

3.2.3 Scaling matrices

When the matrix is a multiple of $\boldsymbol{I}$ the scaling is isotropic. The effect of such mappings is illustrated in figure 3.3.

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 3.4.

A negative element on the diagonal of the matrix results in reflection. Figure 3.5 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 3.6.

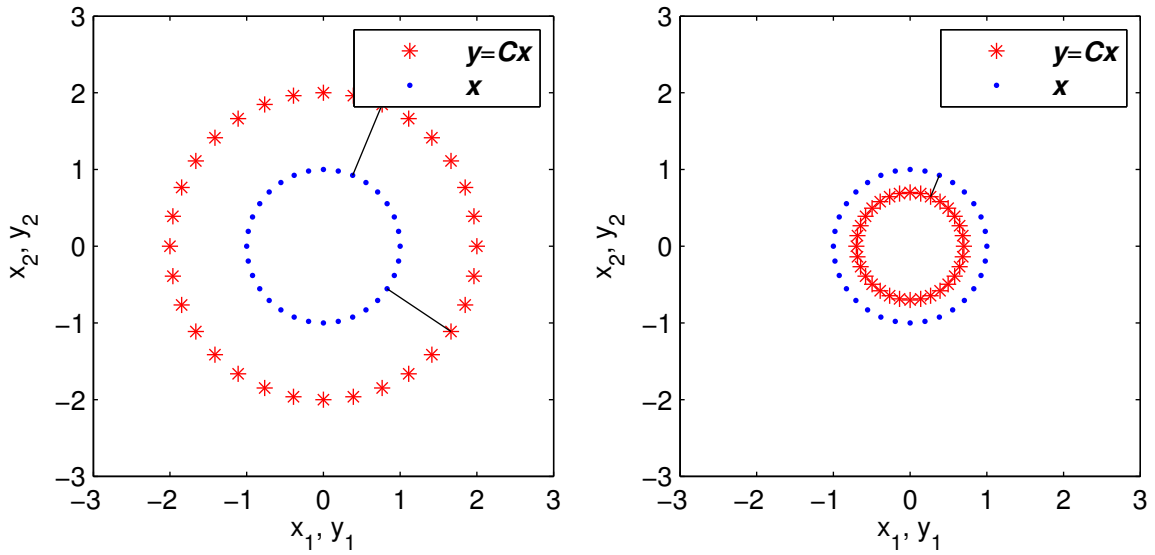
3.2.4 Rotation matrices

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 3.7 shows an example with $\theta = 5°$.

3.2.5 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. Example:

$$\boldsymbol{y} = \begin{bmatrix}1 & 0 & 2\\3 & 1 & 2\end{bmatrix}\begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}1+2\\3+2+2\end{bmatrix} = \begin{bmatrix}3\\7\end{bmatrix}$$



(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 3.3: 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 $\mathbb{R}^n$. A fat matrix produces a lower dimensional output vector, a thin matrix produces a higher dimensional vector.

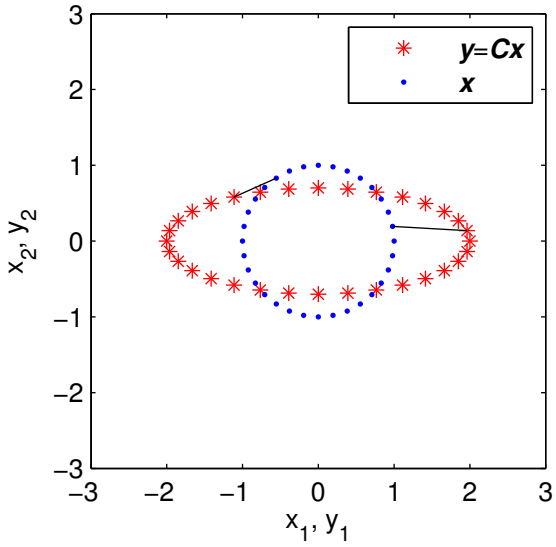
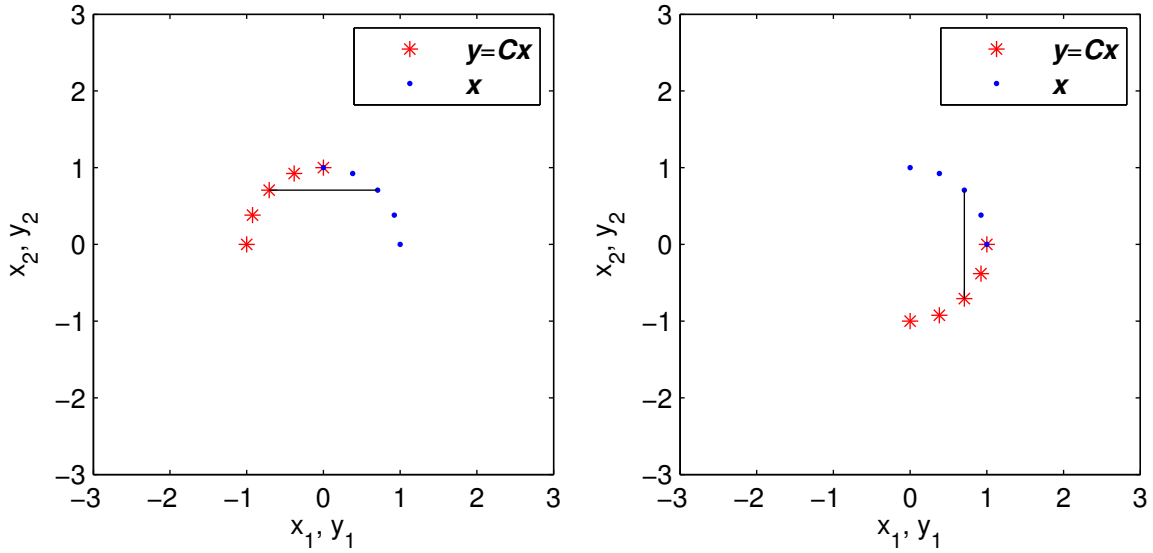


Figure 3.4: Effect of the anisotropic mapping arising from $\boldsymbol{y} = \begin{bmatrix} 2 & 0 \\ 0 & 0.7 \end{bmatrix} \boldsymbol{x}$.



(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 3.5: 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 $\boldsymbol{y}$.

3.3 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 3.8. $\boldsymbol{y}$ 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

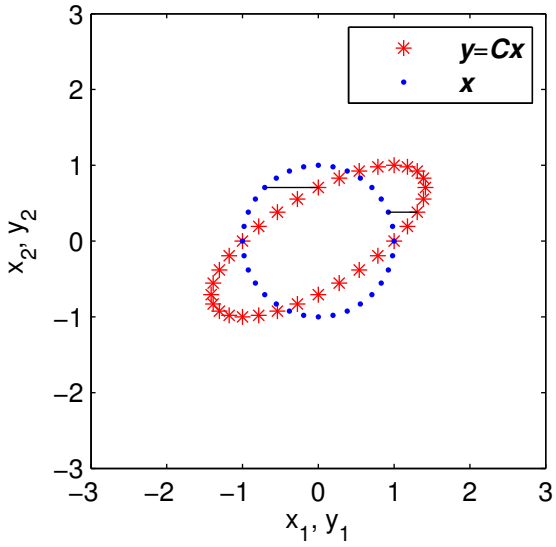


Figure 3.6: Effect of a shear mapping with $\boldsymbol{y} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}$

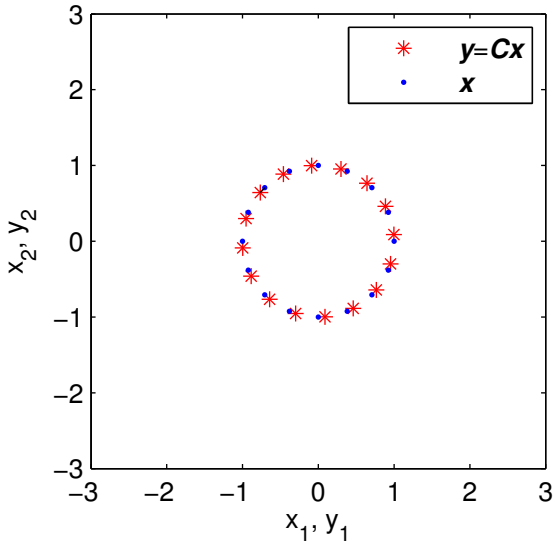


Figure 3.7: Effect of a rotation mapping with $\boldsymbol{y} = \begin{bmatrix} \cos 5^\circ & -\sin 5^\circ \\ \sin 5^\circ & \cos 5^\circ \end{bmatrix} \boldsymbol{x}$

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 original 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 its range. We denote the range of a matrix by $\mathcal{R}$. So, for example, if

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

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

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

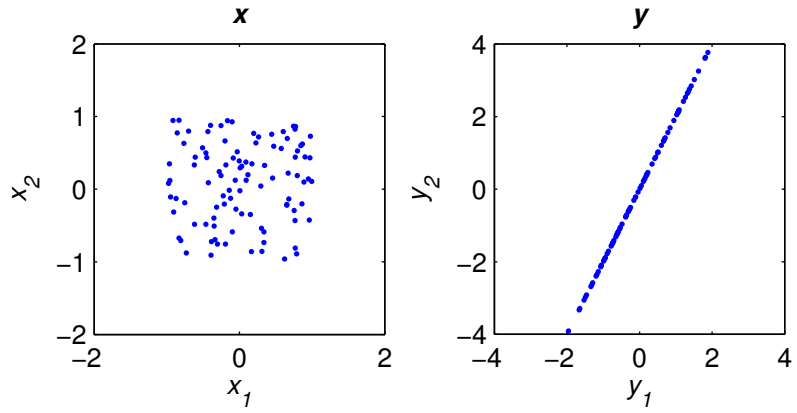


Figure 3.8: Mapping with a rank deficient matrix. A total of 100 random vectors are chosen for $\boldsymbol{x}$ such that x_1 and x_2 are iid and each is $\sim \mathcal{U}(-1, 1)$. Each is mapped to a corresponding $\boldsymbol{y}$.

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.

3.3.1 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*.
- 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.

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

- If $\operatorname{rank} \boldsymbol{C} = n$ then the columns of $\boldsymbol{C}$ are linearly independent and we describe $\boldsymbol{C}$ as *full rank*.
- If $\operatorname{rank} \boldsymbol{C} < n$ 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 $\operatorname{rank} \boldsymbol{C} \leq n$. 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 $\operatorname{rank} \boldsymbol{C} \leq m$. Thus, for any matrix $\boldsymbol{C} \in \mathbb{R}^{n \times m}$, we have

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

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.

3.4 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 $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{y}$.

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

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

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

For example,

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

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 $\boldsymbol{A} \in \mathbb{K}^{n \times p}$ and $\boldsymbol{A} \in \mathbb{K}^{p \times m}$. Then the answer $\boldsymbol{C} \in \mathbb{K}^{n \times m}$.
- When composing multiple mappings, the matrix for the first mapping is placed furthest to the right.

3.5 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 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*.

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

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

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

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.

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

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,

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

(As an aside, $(\boldsymbol{AB})^{\mathsf{T}} = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{A}^{\mathsf{T}}$ also, and this reversal pattern comes up elsewhere too.)

3.6 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 3.9.

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.

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

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

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

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

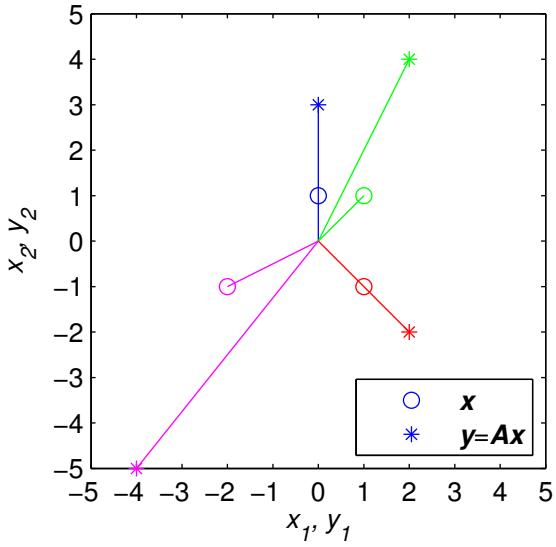


Figure 3.9: Rotation of points about the points when undergoing mapping by a matrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$.

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 3.9 and 3.10 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 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, $\boldsymbol{v} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$.

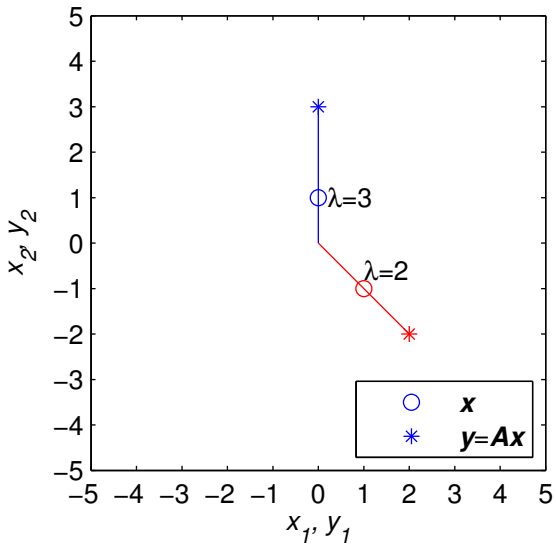


Figure 3.10: Eigenvectors of the matrix $\boldsymbol{A}$ with associated eigenvalues shown.

Similarly,

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

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

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 3.10.

3.6.1 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*.

3.7 The Cayley-Hamilton theorem

Theorem 1. *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 it 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 ration of the adjugate and determinant. For our previous example we have

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

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

Corollary 2. *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.

4 State Space Modelling

4.1 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 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 4.1. 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.

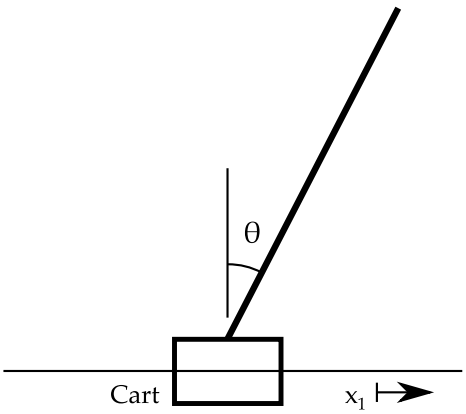


Figure 4.1: An inverted pendulum mounted on a cart.

4.1.1 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. In the general case $\boldsymbol{f}$ can be non-linear and

vary with time. If we restrict ourselves to systems that can be modelled as time invariant then we can make a slight simplification to the state equation.

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

Note that the time derivative of $\boldsymbol{x}$ is a just a vector of the derivatives of each state variable,

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

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

4.1.2 Reduction to Systems of First Order DEs

Notice that the state space form only contains first derivatives. However, this does not stop us using it to describe systems that are governed by differential equations including higher derivatives, as we can

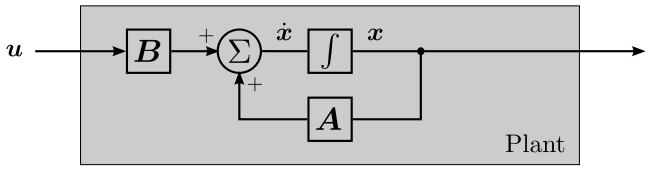


Figure 4.2: Graphical representation of the equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$

express higher order differential equations as systems of first order equations. For example, consider a second order DE:

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

We can define two state variables x_1 and x_2 such that $x_1 = r$ and $x_2 = \dot{r}$. 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 4.3. First write a differential equation describing the motion of the mass: $m\ddot{x} = \sum F_{\text{ext}} =$

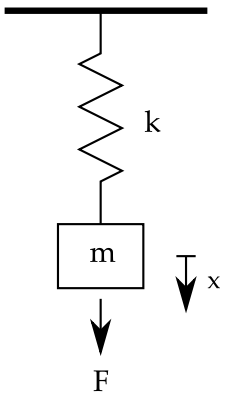


Figure 4.3: A mass spring system with an external applied force F.

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

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

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:

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

4.1.3 Exercise: Add a dashpot to the previous example

We can add a dashpot to the system and see how that changes the state space model. Figure 4.4 shows a schematic of the modified system.

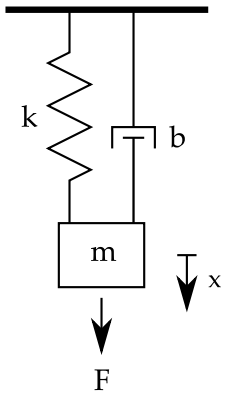


Figure 4.4: A mass spring damper system with an external applied force F.

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

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

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

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

4.1.4 An example that begins with multiple DEs

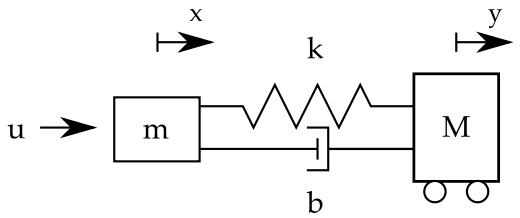


Figure 4.5: Two carts separated with a spring damper system.

We can also consider more complex examples. For example, the two mass system showed in figure 4.5 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 $[x_1, x_2, x_3, x_4]^\mathsf{T} = [x, \dot{x}, y, \dot{y}]^\mathsf{T}$

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

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

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

4.1.5 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 space. This space, the state 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 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 often known as *phase space* in fields outside engineering.) A simple pendulum of length L that is undergoing small 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 4.6. 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.

4.2 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. 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 $d_{ij} = 0 \quad \forall(i,j)$.

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

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

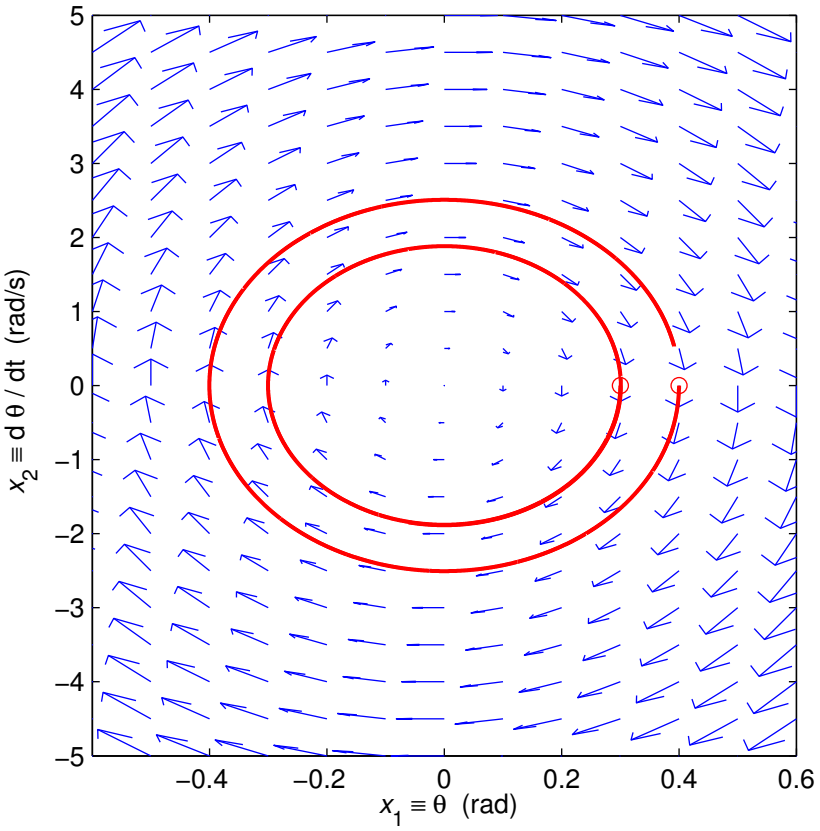


Figure 4.6: Phase portrait for a planar pendulum undergoing small oscillations. The blue arrows show the local direction and magnitude of $\dot{\boldsymbol{x}}$. The red tracks indicate the path in state space that will be taken by two different systems that begin at the red circles.

If we 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 $\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$.
- 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
$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$
- 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 $\boldsymbol{D} = \begin{bmatrix} \frac{1}{m} \end{bmatrix}$.

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 $\begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}$. 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 4.7.

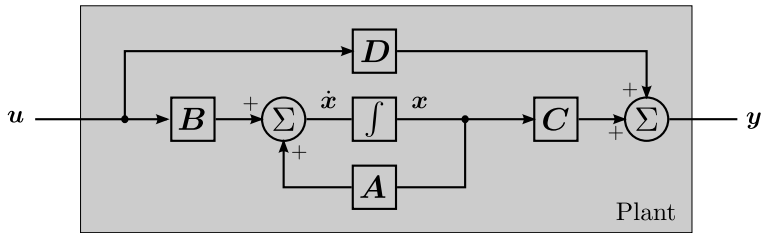


Figure 4.7: Graphical representation of the complete state space model for a system.

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

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

4.3 Examples of State Space models for Electronic Circuits

4.3.1 A passive circuit

Consider a state space model for the circuit shown in 4.8. 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.

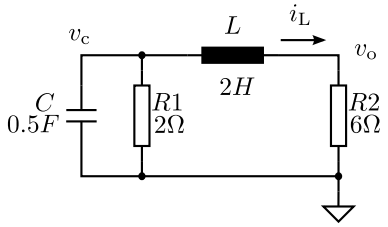


Figure 4.8: A passive circuit with no input.The output of this circuit is the node v_o.

$$\begin{aligned} C\frac{\mathrm{d}v_\mathrm{c}}{\mathrm{d}t} + \frac{v_\mathrm{c}}{R_1} + i_\mathrm{L} &= 0 \\ -i_\mathrm{L} + \frac{v_\mathrm{o}}{R_2} &= 0 \end{aligned} \tag{4.1}$$

But $v_\mathrm{o} = v_\mathrm{c} - L\dfrac{\mathrm{d}i_\mathrm{L}}{\mathrm{d}t}$, so we can substitute into (4.1) 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

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

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.

4.3.2 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 4.9.

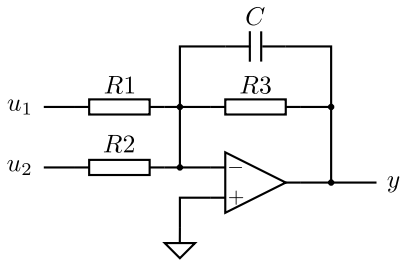


Figure 4.9: An opamp circuit with multiple inputs.

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

$$\frac{u_1}{R_1} + \frac{u_2}{R_2} + \frac{v_c}{R_3} + C\frac{\mathrm{d}v_c}{\mathrm{d}t} = 0 \tag{4.2}$$

We choose our state variable $x_1 = v_c$.

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

$$= -\frac{x_1}{R_3 C} - \frac{u_1}{R_1 C} - \frac{u_2}{R_2 C}$$

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

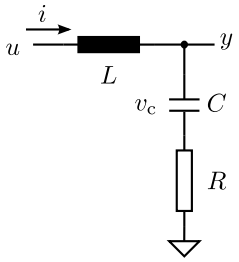


Figure 4.10: An RLC circuit.

4.3.3 An RLC circuit

Consider the RLC filter shown in figure 4.10. 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

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

4.3.4 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

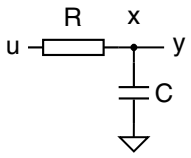


Figure 4.11: An RC low pass filter.

$\boldsymbol{x}$ 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,	$\boldsymbol{D} \in \mathbb{R}^{p \times m}$

4.4 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 find the 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 4.11. 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.

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

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

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.

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

4.5 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.
- 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

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

4.6 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 4.12, 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 $\boldsymbol{x}_\mathrm{o}$.

Now, we know that

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

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

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

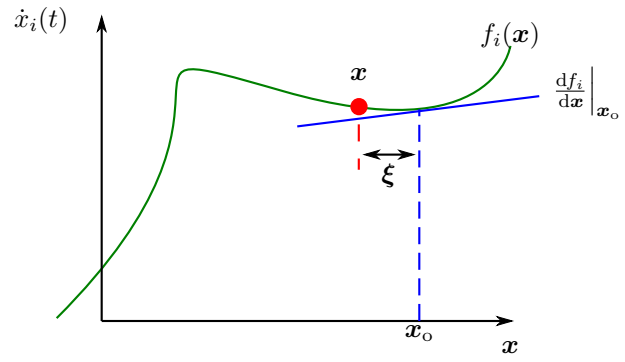


Figure 4.12: Linearisation of a system in the x_i direction. The system is operating near the operating point $\boldsymbol{x}_\mathrm{o}$.

Where $\mathrm{D}\boldsymbol{f}$ is the Jacobian of $\boldsymbol{f}$.

Because $\boldsymbol{x}_\mathrm{o}$ is a fixed operating point, we have $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\mathrm{o} = \boldsymbol{f}(\boldsymbol{x}_\mathrm{o}) = 0$, so we can cancel those terms from each side of our expansion.

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

Because we assume that $\boldsymbol{\xi}$ is small, we neglect the higher order terms.

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

If you remember that the Jacobian of $\boldsymbol{f}$ at $\boldsymbol{x}_\mathrm{o}$ 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

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

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 $\boldsymbol{x}_\mathrm{o}$.

We also need to also consider inputs $\boldsymbol{u}$. We will consider the input to be $\boldsymbol{u} = \boldsymbol{u}_\mathrm{o} + \boldsymbol{v}$. If we 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

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

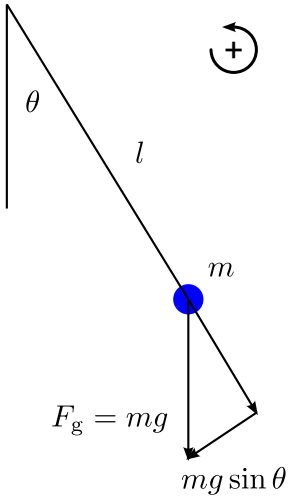


Figure 4.13: A simple pendulum

As an example, consider a simple pendulum with length l and a bob with mass m as shown in figure 4.13. We can use Newton's second law to determine the motion of the bob.

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

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

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

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

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

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. 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 more dramatically. 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 particular operting 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 Time Response of State Space Systems

5.1 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. That is, we would like to 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

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

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

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

where $\boldsymbol{\Phi}(t) := \mathcal{L}^{-1}\left\{(s\boldsymbol{I} - \boldsymbol{A})^{-1}\right\}$ is the *state-transition* matrix. To find $\boldsymbol{\Phi}$ one must 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.

5.1.1 $\boldsymbol{\Phi}$ and the Matrix Exponential

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

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

Now we can calculating 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,

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

Calculating this by hand is often not as intractable as the series often terminated. In any case the Cayley-Hamilton theorem ensures that we do not need to calculate matrix powers beyond A^{n-1}. 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 found 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 also convenient to extend the notation for $\boldsymbol{\Phi}$ to include two time instants. $\boldsymbol{\Phi}(t_1, t_0) := \mathrm{e}^{(t_1-t_0)\boldsymbol{A}}$.

5.1.2 State Transition example

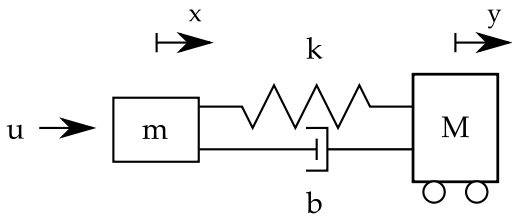


Figure 5.1: A system of two carts separated by a damped spring.

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 5.2 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.

5.2 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

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

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 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.

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

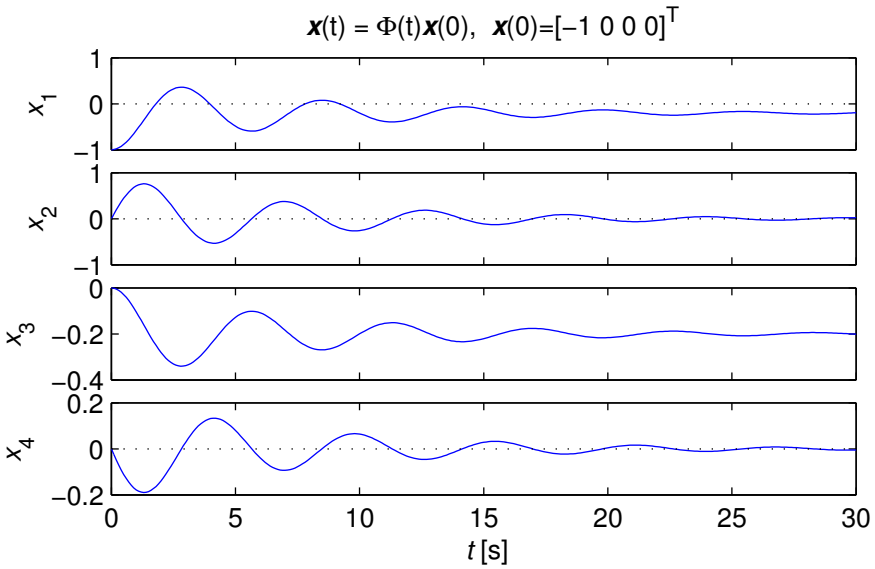


Figure 5.2: Response of the state variables of the system in figure 5.1 as it relaxes from the initial condition $\boldsymbol{x}(0) = [-1\ 0\ 0\ 0]^\mathsf{T}$.

where $\chi(s) = \det(s\boldsymbol{I} - \boldsymbol{A})$ is a polynomial of s and each of the $n_{ij}(s)$ 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.

5.3 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.

Recall that different pole configurations in a transfer functions correspond to a familiar set of possible system modes.

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

The new formulation of the problem will lead to the identical family of modes. Indeed it would be troubling if it did not.

5.3.1 Modal Responses - example

Let's construct a random system with 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 5.3 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.

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

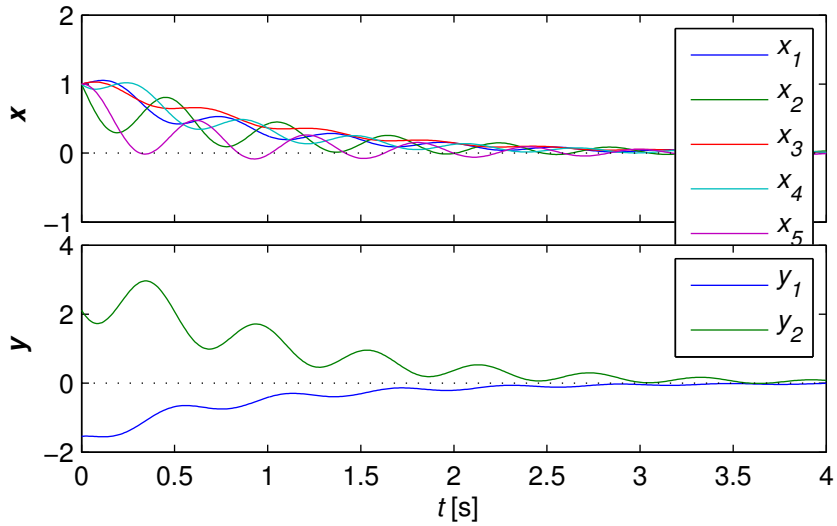


Figure 5.3: Response of a five-state, two-output state space model to an arbitrary initial condition $\boldsymbol{x}(0) = [1\ 1\ 1\ 1\ 1]^{\mathsf{T}}$. Notice that the response of each state variable and of each output is a combination of different system modes.

to one of the system's eigenvector and allow the system to evolve with time.

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

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

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

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

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 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 that the eigenvectors of $\boldsymbol{A}$ are

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

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 5.4. 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 $c_i e^{-1.13t}$ for an appropriately chosen constant c_i.

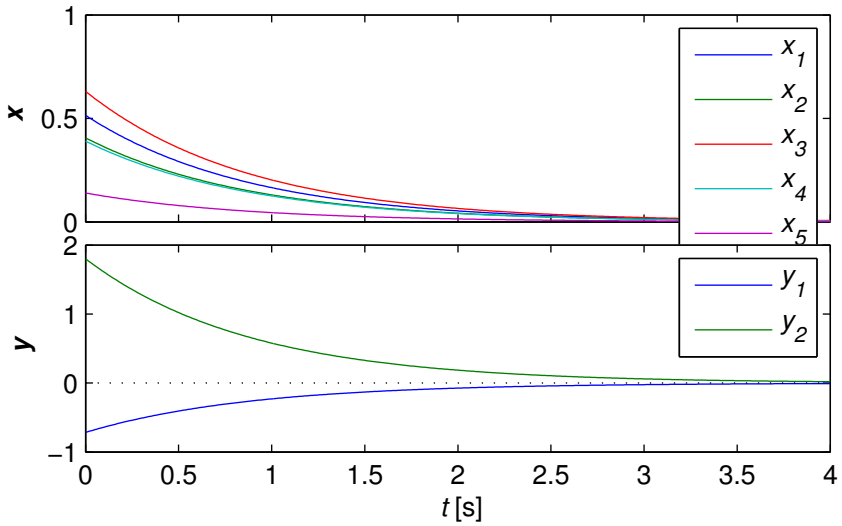


Figure 5.4: Response of the system when the initial condition is set to the eigenvector associated with eigenvalue -1.1354.

The same pattern is apparent if we examine the response to an initial condition corresponding to the other eigenvectors, as shown in figures 5.5 to 5.8. We again see that in each case all of the state variables respond as $c_i e^{\lambda t}$. 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.

5.3.2 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 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 rotation.

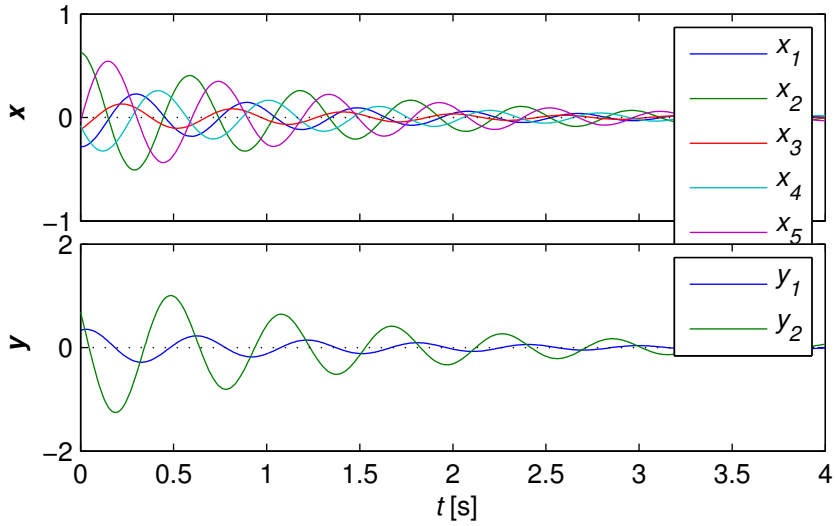


Figure 5.5: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue $-0.7483 + 10.5927\text{j}$.

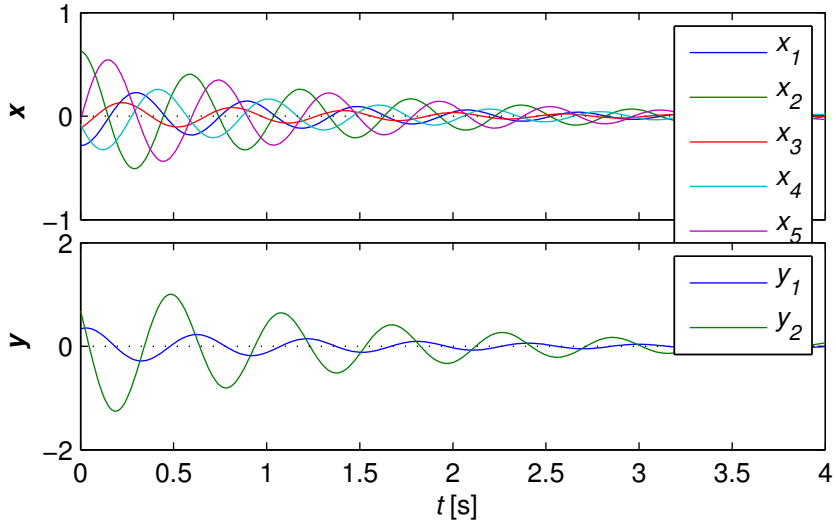


Figure 5.6: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue $-0.7483 - 10.5927\text{j}$.

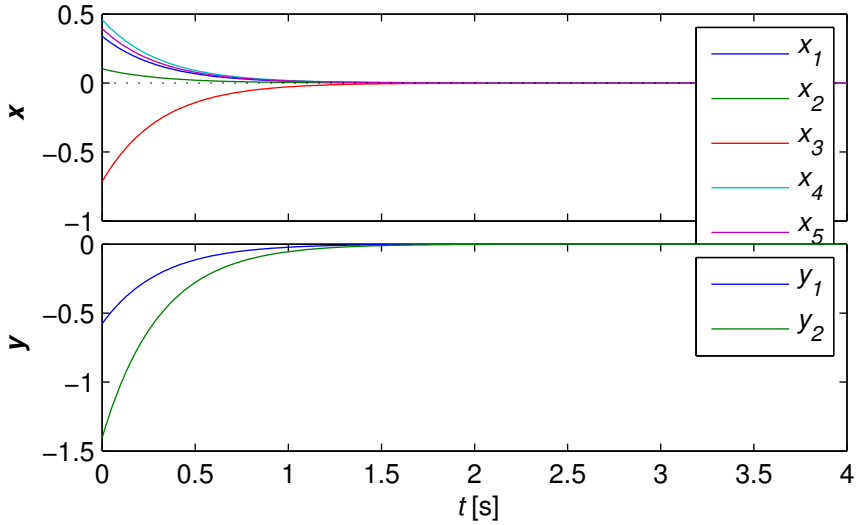


Figure 5.7: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue -3.2353.

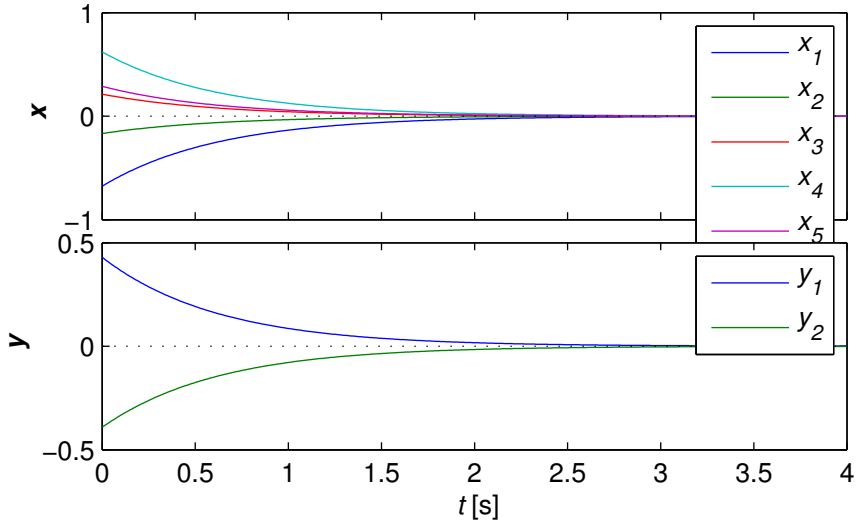


Figure 5.8: Response of the system when the initial condition is set to the eigenvector associated with the eigenvalue -1.6087.

5.4 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.

5.5 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 constant for $t \in [0, t_\mathrm{s})$, so we can move it outside of 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.

5.5.1 Continuous to discrete time conversion – Summary

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.

5.6 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 defiend 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 $\boldsymbol{\Phi}(t)$ 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,

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

6 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.

6.1 State Space to Transfer Function Conversion

We would like a way to find a transfer function representation of a system expressed in state space. Now $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$. 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

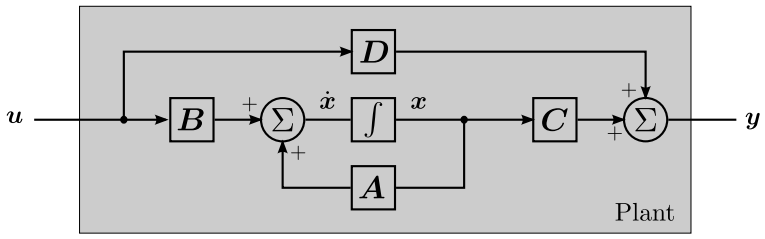


Figure 6.1: Graphical representation of the state space model of a plant.

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

6.1.1 Example One

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

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

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

6.1.2 Example Two

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

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

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

$$\boldsymbol{G}(s) = \begin{bmatrix} \frac{9(s+2)}{(s+1)(s+2)} & \frac{6(s+2)}{(s+1)(s+2)} \\ \frac{9(3s-7)}{(s+1)(s+2)} & \frac{12(4s-1)}{(s+1)(s+2)} \end{bmatrix}$$

Now, $\boldsymbol{Y} = \boldsymbol{G}\boldsymbol{U}$

$$\text{So, } \begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{9}{(s+1)} & \frac{6}{(s+1)} \\ \frac{9(3s-7)}{(s+1)(s+2)} & \frac{12(4s-1)}{(s+1)(s+2)} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}$$

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

6.2 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.

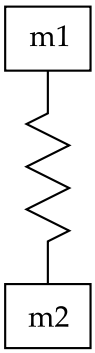


Figure 6.2: Two masses separated by a spring.

For example, for the system shown in figure 6.2 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, as long as the new set is a linear mapping of the old set. That is, we are free to choose any appropriate set of basis vectors we like. The new state variables will be linear combinations of the old 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 $\boldsymbol{P}$, we can write

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

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

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

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

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

6.2.1 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,

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

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

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

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

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

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

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:

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

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

6.2.2 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

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

6.3 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.

- 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.

6.3.1 Modal Canonical Form

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

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

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 modifications to it that we will see shortly) 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

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

6.3.2 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

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

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.)

6.3.3 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,

$$\begin{aligned}\left|\lambda\boldsymbol{I} - \boldsymbol{A}\right| = 0 \implies \begin{vmatrix}\lambda+7 & 12\\ -1 & \lambda\end{vmatrix} &= 0\\ \lambda(\lambda+7)+12 &= 0\\ \lambda^2+7\lambda+12 &= 0\\ (\lambda+3)(\lambda+4) &= 0\\ \lambda &= -3,-4\end{aligned}$$

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.

6.4 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. Consider a discrete time system described by the matrix $\boldsymbol{A}$, which we will assume that we can diagonalise. This implies

$$\begin{aligned}
\boldsymbol{\Lambda} = \boldsymbol{A}_\mathrm{m} &= \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} \text{ for some } \boldsymbol{V} \\
\implies \boldsymbol{A} &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} \\
\text{So, } \boldsymbol{\Phi}(t) \equiv \boldsymbol{A}^t &= \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)^t \\
&= \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right)\cdots\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\right) \\
&= \boldsymbol{V}\boldsymbol{\Lambda}^t\boldsymbol{V}^{-1} \\
\boldsymbol{\Phi}(t) &= \boldsymbol{V}\,\mathrm{diag}(\lambda_i^t)\boldsymbol{V}^{-1}
\end{aligned}$$

This 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, for a diagonal matrix (like $\boldsymbol{\Lambda}$), the matrix exponential *is* equal to the elementwise scalar exponential.

6.5 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 that a repeated pole at λ with multiplicity k leads to a mode of $t^{k-1}\mathrm{e}^{\lambda t}$. Therefore the presence of a $k \times k$ Jordan block in the $\boldsymbol{A}$ matrix of a modal canonical form implies that the time response of the system will contain terms like $t^{k-1}\mathrm{e}^{\lambda t}$. Note that a diagonal form is a special case of the Jordan form, where all of the Jordan blocks have size 1×1.

6.5.1 Evolution of Systems in Jordan Form

For discrete time systems 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.

$$\text{If } \boldsymbol{J} = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} \text{ then, } \boldsymbol{J}^t = \begin{bmatrix} \lambda^t & t\lambda^{t-1} & \frac{t(t-1)}{2}\lambda^{t-2} \\ 0 & \lambda^t & t\lambda^{t-1} \\ 0 & 0 & \lambda^t \end{bmatrix}$$
$$= \begin{bmatrix} \lambda^t & \binom{t}{1}\lambda^{t-1} & \binom{t}{2}\lambda^{t-2} \\ 0 & \lambda^t & \binom{t}{1}\lambda^{t-1} \\ 0 & 0 & \lambda^t \end{bmatrix}$$

Extension to larger Jordan blocks should be self-evident.

6.5.2 Functions of a Matrix in Jordan Form

As an aside, this is an example of a more general feature of matrices in Jordan normal form. Consider $\boldsymbol{J}_{\lambda,n}$ a Jordan block corresponding to an eigenvalue λ with multiplicity n. We can write a general form for the result of applying any differentiable analytic function f to $\boldsymbol{J}$ as

$$f(\boldsymbol{J}_{\lambda,n}) = \begin{bmatrix} f(\lambda) & \frac{\mathrm{d}f(\lambda)}{\mathrm{d}\lambda} & \frac{1}{2!}\frac{\mathrm{d}f(\lambda)}{\mathrm{d}\lambda} & \cdots & \frac{1}{(n-2)!}\frac{\mathrm{d}^{n-2}f(\lambda)}{\mathrm{d}\lambda^{n-2}} & \frac{1}{(n-1)!}\frac{\mathrm{d}^{n-1}f(\lambda)}{\mathrm{d}\lambda^{n-1}} \\ 0 & f(\lambda) & \frac{\mathrm{d}f(\lambda)}{\mathrm{d}\lambda} & \cdots & \frac{1}{(n-3)!}\frac{\mathrm{d}^{n-3}f(\lambda)}{\mathrm{d}\lambda^{n-3}} & \frac{1}{(n-2)!}\frac{\mathrm{d}^{n-2}f(\lambda)}{\mathrm{d}\lambda^{n-2}} \\ 0 & 0 & f(\lambda) & \cdots & \frac{1}{(n-4)!}\frac{\mathrm{d}^{n-4}f(\lambda)}{\mathrm{d}\lambda^{n-4}} & \frac{1}{(n-3)!}\frac{\mathrm{d}^{n-3}f(\lambda)}{\mathrm{d}\lambda^{n-3}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & f(\lambda) & \frac{\mathrm{d}f(\lambda)}{\mathrm{d}\lambda}) \\ 0 & 0 & 0 & \cdots & 0 & f(\lambda) \end{bmatrix}$$

6.5.3 Example - Evolution of a System

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 system that can be converted to a Jordan canonical form

$$\boldsymbol{J} = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 & 0 \\ 0 & \lambda_1 & 1 & 0 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3 \end{bmatrix}$$

The state transition matrix for this system will be

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} \lambda_1^t & t\lambda_1^{t-1} & \frac{t(t-1)}{2}\lambda_1^{t-2} & 0 & 0 \\ 0 & \lambda_1^t & t\lambda_1^{t-1} & 0 & 0 \\ 0 & 0 & \lambda_1^t & 0 & 0 \\ 0 & 0 & 0 & \lambda_2^t & 0 \\ 0 & 0 & 0 & 0 & \lambda_3^t \end{bmatrix}$$

A system having repeated poles (or eigenvalues) is a necessary, but not sufficient condition for the system's 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 $A \in \mathbb{R}^{4\times 4}$ with four repeated eigenvalues at -2 could have any of the following Jordan forms.

$$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$

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 6.3-6.7 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.

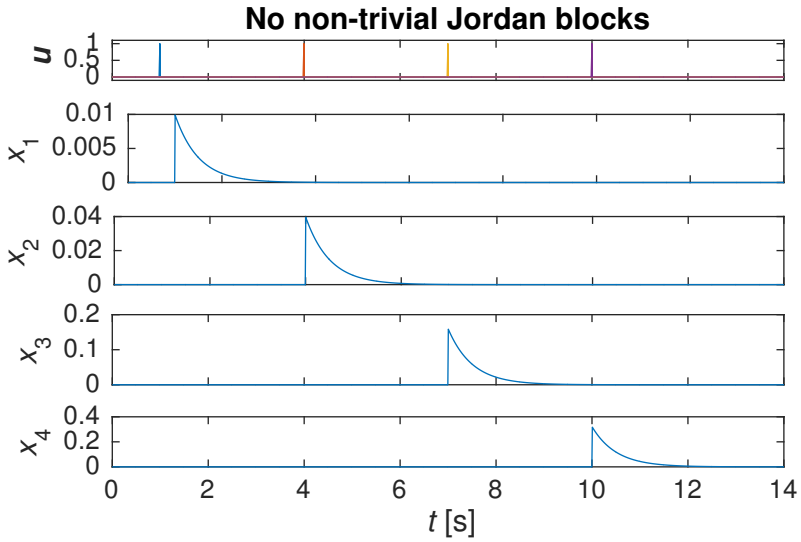


Figure 6.3: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains no non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

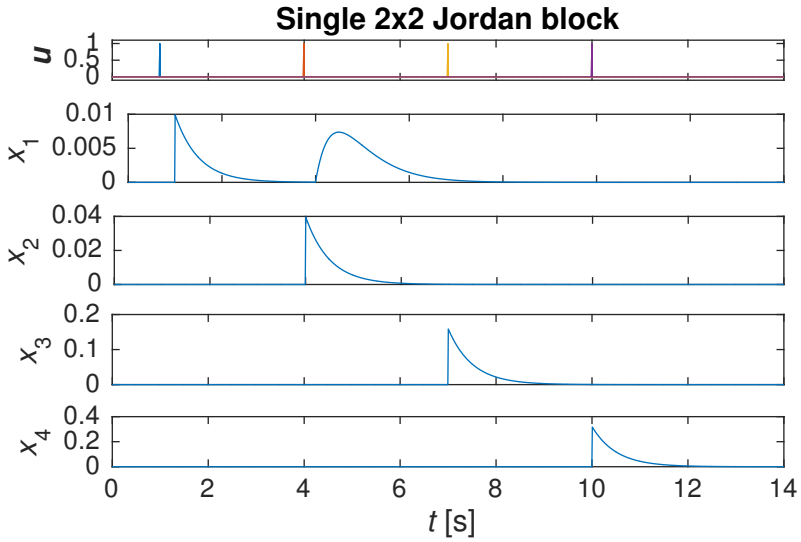


Figure 6.4: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single two-by-two non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

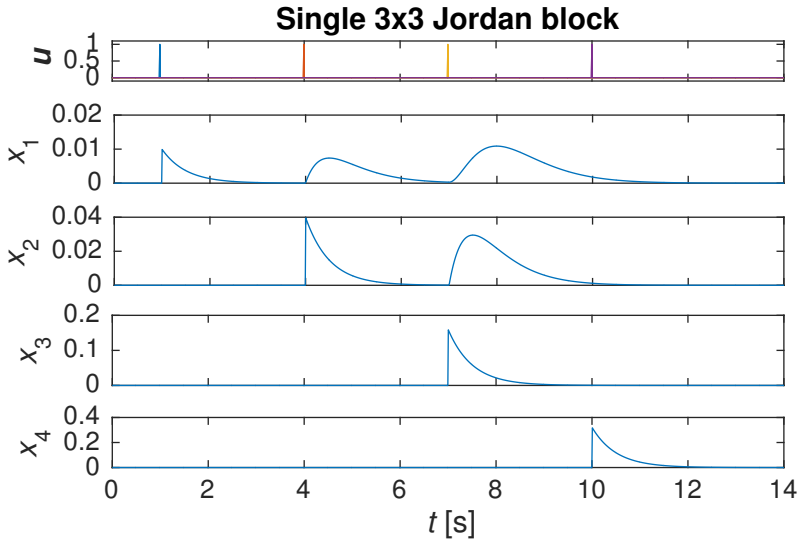


Figure 6.5: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single three-by-three non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

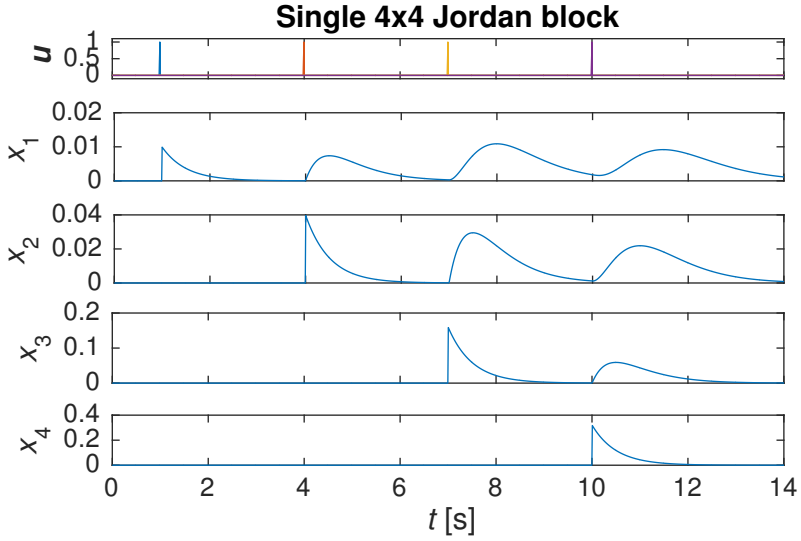


Figure 6.6: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains a single four-by-four non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

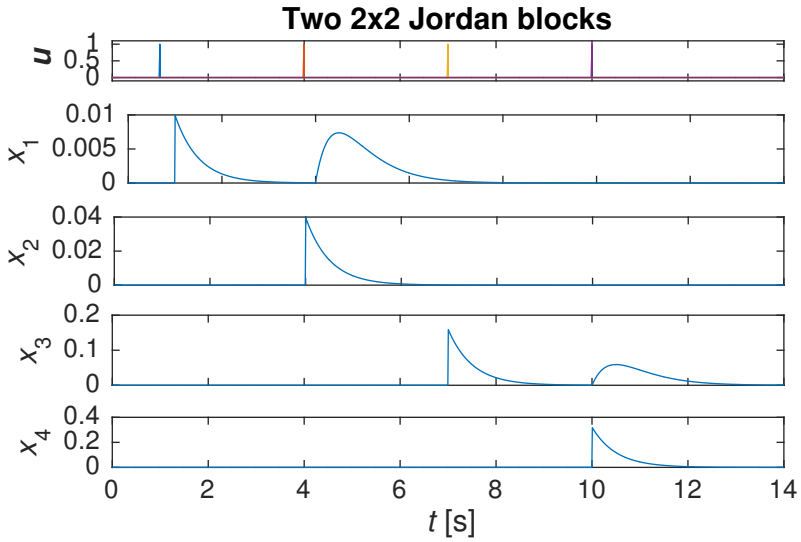


Figure 6.7: Impulse response of a four-state system having four repeated eigenvalues. The matrix contains two two-by-two non-trivial Jordan blocks. $J = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$.

6.6 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 $\boldsymbol{P}$ 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 $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{z}$, but in matlab it is $\boldsymbol{z} = \boldsymbol{P}\boldsymbol{x}$. Matlab's approach seems more intuitive, but it is not the way the field has developed. 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 numbers are similar. This is 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.

6.7 Appendix – Other Canonical Forms

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.

6.7.1 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

$$\boldsymbol{A}_\mathrm{c} = \begin{bmatrix} -d_{n-1} & -d_{n-2} & \cdots & -d_1 & -d_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \boldsymbol{B}_\mathrm{c} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\boldsymbol{C}_\mathrm{c} = \begin{bmatrix} n_{n-1} - d_{n-1}n_n \vdots n_{n-2} - d_{n-2}n_n \vdots \cdots \vdots n_1 - d_1 n_n \vdots n_0 - d_0 n_n \end{bmatrix}$$

$$\text{and } \boldsymbol{D}_\mathrm{c} = n_n.$$

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

$$\boldsymbol{A}_\mathrm{c} = \begin{bmatrix} -d_{n-1} & -d_{n-2} & \cdots & -d_1 & -d_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \boldsymbol{B}_\mathrm{c} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\boldsymbol{C}_\mathrm{c} = \begin{bmatrix} n_{n-1} & n_{n-2} & \cdots & n_1 & n_0 \end{bmatrix}$$
$$\text{and } \boldsymbol{D}_\mathrm{c} = 0.$$

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:

$$\begin{aligned}
\boldsymbol{A}_\mathrm{c} &= \begin{bmatrix} -d_1 & -d_0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -3 & -2 \\ 1 & 0 \end{bmatrix} \\
\boldsymbol{B}_\mathrm{c} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
\boldsymbol{C}_\mathrm{c} &= \begin{bmatrix} n_1 - d_1 n_2 & n_0 - d_0 n_2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \end{bmatrix} \\
\boldsymbol{D}_\mathrm{c} &= 0
\end{aligned}$$

6.7.2 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, $\boldsymbol{P}$, 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 $(\boldsymbol{A}_\mathrm{c}, \boldsymbol{B}_\mathrm{c}, \boldsymbol{C}_\mathrm{c}, \boldsymbol{D}_\mathrm{c})$.

We know that $\boldsymbol{A}_\mathrm{c} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P} \implies \boldsymbol{A}_\mathrm{c}\boldsymbol{P}^{-1} = \boldsymbol{P}^{-1}\boldsymbol{A}$

Now, we know something about the structure of $\boldsymbol{A}_\mathrm{c}$ 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 $\boldsymbol{P}^{-1}$ 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,

$$\boldsymbol{P}^{-1} = \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T} \\ \boldsymbol{p}_2^\mathsf{T} \\ \boldsymbol{p}_3^\mathsf{T} \end{bmatrix}$$

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 $\boldsymbol{P}^{-1}\boldsymbol{B} = \boldsymbol{B}_\boldsymbol{c}$, 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,

$$\begin{aligned}\boldsymbol{p}_3^\mathsf{T}\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B}\end{bmatrix} &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}\\ \boldsymbol{p}_3^\mathsf{T} &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}\boldsymbol{\mathcal{C}}^{-1}\end{aligned}$$

where $\boldsymbol{\mathcal{M}}_\mathrm{c} = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B}\end{bmatrix}$ 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 $\boldsymbol{P}$.

$$\boldsymbol{P}^{-1} = \begin{bmatrix}\boldsymbol{p}_1^\mathsf{T}\\ \boldsymbol{p}_2^\mathsf{T}\\ \boldsymbol{p}_3^\mathsf{T}\end{bmatrix} = \begin{bmatrix}\boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}^2\\ \boldsymbol{p}_3^\mathsf{T}\boldsymbol{A}\\ \boldsymbol{p}_3^\mathsf{T}\end{bmatrix}$$

where $\boldsymbol{p}_3^\mathsf{T}$ is given by the last row of the inverse of the controllability matrix.

We can now use $\boldsymbol{P}^{-1}$ to convert to control canonical form.

1. Calculate the controllability matrix,

$$\boxed{\boldsymbol{\mathcal{M}}_\mathrm{c} = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B}\cdots & \boldsymbol{A}^{n-1}\boldsymbol{B}\end{bmatrix}}$$

   where $\boldsymbol{A}^2 = \boldsymbol{AA}$ 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 $\boldsymbol{P}$ 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 $\boldsymbol{P}^{-1} = \begin{bmatrix}\boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}^{n-1}\\ \boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}^{n-2}\\ \vdots\\ \boldsymbol{p}_n^\mathsf{T}\boldsymbol{A}\\ \boldsymbol{p}_n^\mathsf{T}\end{bmatrix}$
5. Use $\boldsymbol{P}^{-1}$ to find the system matrices in control canonical form.

6.7.3 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 $\dot{\boldsymbol{x}} = \boldsymbol{A}_\mathrm{o}\boldsymbol{x} + \boldsymbol{B}_\mathrm{o}\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{C}_\mathrm{o}\boldsymbol{x} + \boldsymbol{D}_\mathrm{o}\boldsymbol{u}$, where

$$\boldsymbol{A}_\mathrm{o} = \begin{bmatrix}0 & 0 & \cdots & 0 & -d_0\\ 1 & 0 & \cdots & 0 & -d_1\\ 0 & 1 & \cdots & 0 & -d_2\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & -d_{n-1}\end{bmatrix}, \boldsymbol{B}_\mathrm{o} = \begin{bmatrix}n_0 - d_0 n_n\\ n_1 - d_1 n_n\\ n_2 - d_2 n_n\\ \vdots\\ n_{n-1} - d_{n-1}n_n\end{bmatrix}$$

$$\boldsymbol{C}_\mathrm{o} = \begin{bmatrix}0 & 0 & \cdots & 0 & 1\end{bmatrix} \text{ and } \boldsymbol{D}_\mathrm{o} = n_n.$$

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:

$$\begin{aligned} \boldsymbol{A}_{\text{o}} &= \begin{bmatrix} 0 & -d_0 \\ 1 & -d_1 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix} \\ \boldsymbol{B}_{\text{o}} &= \begin{bmatrix} n_0 - d_0 n_2 \\ n_1 - d_1 n_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\ \boldsymbol{C}_{\text{o}} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ \boldsymbol{D}_{\text{o}} &= 0 \end{aligned}$$

6.7.4 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

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

7 Regulator Design

We have now developed our state space theory sufficiently to examine the design of compensators (aka controllers). There are two main classes of control problems:

1. Regulator problems, where we seek to stabilise a system at some set point.
2. Servo (or tracking) problems, where we want our system to follow some command input(s).

Regulator problems are slightly simpler, as we do not have command (or reference) inputs entering the system. We will begin with regulator problems and will return to servo design later.

7.1 Operating Points

In a regulator problem we are interested in stabilising a system about some steady state operating point $\boldsymbol{x}_\mathrm{o}$, which we will assume corresponds to some steady state input $\boldsymbol{u}_\mathrm{o}$.

$$\begin{aligned} \text{Now,} \quad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \text{So,} \ \frac{\mathrm{d}(\boldsymbol{x} - \boldsymbol{x}_\mathrm{o})}{\mathrm{d}t} &= \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_\mathrm{o}) + \boldsymbol{B}(\boldsymbol{u} - \boldsymbol{u}_\mathrm{o}) \end{aligned}$$

Let's define $\boldsymbol{\xi} = \boldsymbol{x} - \boldsymbol{x}_\mathrm{o}$ and $\boldsymbol{v} = \boldsymbol{u} - \boldsymbol{u}_\mathrm{o}$ as deviations from the desired steady state values.

$$\implies \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{v}$$

The deviations are governed by the same state equation as the state. We typically drop the deviation variables and proceed without loss of generality.

In effect we will design a controller to regulate the system around $\boldsymbol{\xi} = 0$ and then use that *same* controller to regulate around some arbitrary $\boldsymbol{x}_\mathrm{o}$. Notice that the fact that the model around $\boldsymbol{x}_\mathrm{o}$ is unchanged from the general model is a consequence of the assumed linearity of the system. Once we have formed a model around the operating point $\boldsymbol{x}_\mathrm{o}$ we will then forget about $\boldsymbol{x}_\mathrm{o}$ while designing the controller. However, when it comes to implementation we *must* remember to include the $\boldsymbol{u}_\mathrm{o}$ signal required to produce the desired operating point. That is, we will develop control techniques to producing an input signal that will actually be $\boldsymbol{v}$. We will need to apply a real signal of $\boldsymbol{u}_\mathrm{o} + \boldsymbol{v}$ to our system.

We will need to know what steady state $\boldsymbol{u}_\mathrm{o}$ we need to apply to hold the system at some operating point $\boldsymbol{x}_\mathrm{o}$. Now,

$$\dot{\boldsymbol{x}}_\mathrm{o} = \boldsymbol{A}\boldsymbol{x}_\mathrm{o} + \boldsymbol{B}\boldsymbol{u}_\mathrm{o}$$

But at steady state $\dot{\boldsymbol{x}} = 0$, so

$$\begin{aligned} -\boldsymbol{A}\boldsymbol{x}_\mathrm{o} &= \boldsymbol{B}\boldsymbol{u}_\mathrm{o} \\ \implies \boldsymbol{u}_\mathrm{o} &= -\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}_\mathrm{o} \end{aligned}$$

This last equation makes sense *only* if $\boldsymbol{B}$ is invertible. This requires that there be as many actuators as there are system states ($m = n$) and that those actuators not be arranged in a way that yields a singular $\boldsymbol{B}$. In such a case we can solve uniquely for $\boldsymbol{u}_\mathrm{o}$

We can also identify the $\boldsymbol{x}_\mathrm{o}$ locations for which we can find some $\boldsymbol{u}_\mathrm{o}$ that results in the system remaining at $\boldsymbol{x}_\mathrm{o}$.

$$\begin{aligned} \dot{\boldsymbol{x}}_\mathrm{o} &= \boldsymbol{A}\boldsymbol{x}_\mathrm{o} + \boldsymbol{B}\boldsymbol{u}_\mathrm{o} \\ 0 &= \boldsymbol{A}\boldsymbol{x}_\mathrm{o} + \boldsymbol{B}\boldsymbol{u}_\mathrm{o} \quad \text{as } \boldsymbol{x} \text{ is not changing} \\ -\boldsymbol{A}\boldsymbol{x}_\mathrm{o} &= \boldsymbol{B}\boldsymbol{u}_\mathrm{o} \\ \boldsymbol{x}_\mathrm{o} &= -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{u}_\mathrm{o} \end{aligned}$$

Therefore the set of possible equilibrium points in $\boldsymbol{x}$ is

$$\begin{aligned} \mathcal{E}_{\boldsymbol{x}} &= \{\boldsymbol{x}_\mathrm{o} : \boldsymbol{A}\boldsymbol{x}_\mathrm{o} + \boldsymbol{B}\boldsymbol{u}_\mathrm{o} = 0, \boldsymbol{u}_\mathrm{o} \in \mathbb{R}^m\} \\ &= \{\boldsymbol{x}_\mathrm{o} : \boldsymbol{x}_\mathrm{o} = -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{u}_\mathrm{o}, \boldsymbol{u}_\mathrm{o} \in \mathbb{R}^m\} \\ \implies \mathcal{E}_{\boldsymbol{x}} &= \mathcal{R}(\boldsymbol{A}^{-1}\boldsymbol{B}) \end{aligned}$$

You should check that any proposed control scheme allows you to position the system at the desired operating point(s). Notice that $\dim(\mathcal{E}_{\boldsymbol{x}}) \leq \text{rank}(\boldsymbol{B})$, because $\boldsymbol{A}^{-1}\boldsymbol{B}$ can have no more independent columns that rank $\boldsymbol{B}$. For example, if a system has one input we can only choose where to put the system operating point along a one dimensional subspace (a line), no matter the dimensionality of the state space. Many real systems have rank $\boldsymbol{B} < n$, because in practice we don't always need to be able to force the position to arbitrary $\boldsymbol{x}$ (or $\boldsymbol{y}$). As long as $\mathcal{E}_{\boldsymbol{x}}$ includes the desired operating point we can minimise the number of actuators required.

We know that if $\boldsymbol{B}$ is invertible we can find the unique $\boldsymbol{u}_\text{o}$ to produce a desired $\boldsymbol{x}_\text{o}$. However, we also need to consider what happens when we cannot invert $\boldsymbol{B}$, either because it is singular, or because it is not square.

If $m \neq n$ then we cannot invert $\boldsymbol{B}$. There are two cases to consider.

- $m > n$ ($\boldsymbol{B}$ is fat) and rank $\boldsymbol{B} = n$. We have more actuators than states. In such a case there are typically multiple strategies that would produce a desired $\boldsymbol{x}_\text{o}$. That is, we have a *set* of possible $\boldsymbol{u}$ that would satisfy our requirements. In such a case we might select the smallest $\boldsymbol{u}$ that works. We could find this using the pseudoinverse (`pinv` in Matlab).

$$\boldsymbol{u}_\text{o} = -\boldsymbol{B}^\dagger \boldsymbol{A}\boldsymbol{x}_\text{o}$$

- $m < n$ ($\boldsymbol{B}$ is skinny) or rank $\boldsymbol{B} < n$. In this case there are fewer actuators than states. There is no guarantee that we will be able to find a $\boldsymbol{u}$ to achieve our desired $\boldsymbol{x}_\text{o}$. Where it is possible to find the desired input, it can again be done using the pseudo-inverse.

Let's consider an example with model $\boldsymbol{A} = \begin{bmatrix} 0 & 0.2 \\ -0.2 & 0 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The blue arrows show the

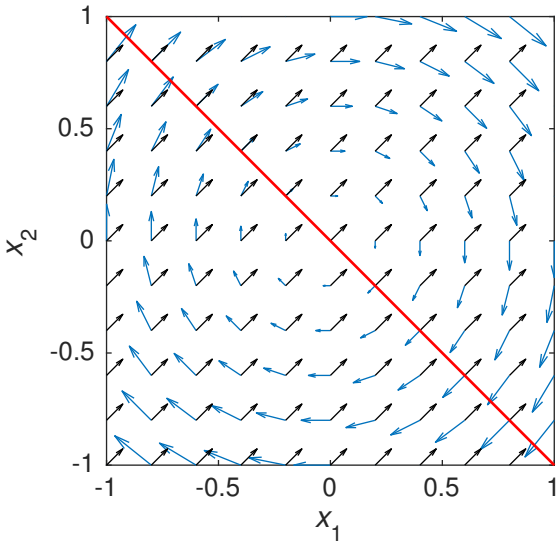


Figure 7.1: The blue vectors show that action of $\boldsymbol{A}\boldsymbol{x}$, and the black vectors indicate the direction of $\boldsymbol{B}\boldsymbol{u}$. The input can only balance the effect of $\boldsymbol{A}\boldsymbol{x}$ for locations along the red line.

contribution to $\dot{\boldsymbol{x}}$ from $\boldsymbol{A}\boldsymbol{x}$. The black arrows show the *direction* in which $\boldsymbol{B}\boldsymbol{u}$ pushes $\boldsymbol{x}$. Remember that it can be scaled. The two can only balance for values of $\boldsymbol{x}$ on the red line.

The subspace of accessible operating points is given by

$$\begin{aligned} \mathcal{E}_{\boldsymbol{x}} &= \mathcal{R}(\boldsymbol{A}^{-1}\boldsymbol{B}) \\ &= \mathcal{R}\begin{bmatrix} -5 \\ 5 \end{bmatrix} \end{aligned}$$

That is, the possible operating points are all multiples of the vector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. This is consistent with the red line in figure 7.1.

Interestingly, we will see later that we can drive this system to reach an arbitrary value of $\boldsymbol{x}$, but it can't *stay* at that arbitrary point.

Now let's consider $\boldsymbol{A} = \begin{bmatrix} 0 & 0.2 \\ -0.2 & 0 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

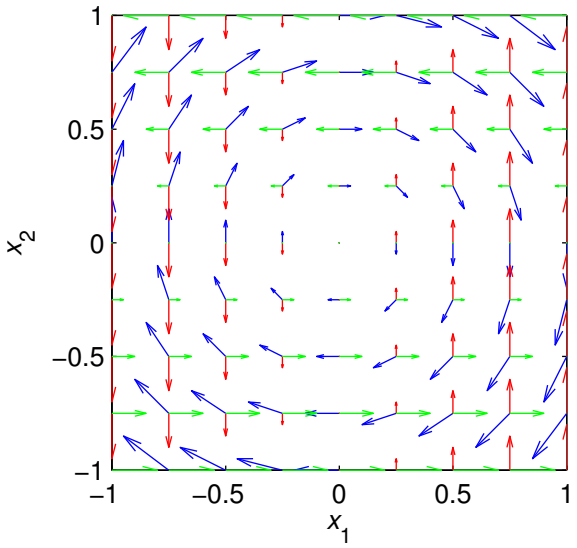


Figure 7.2: Illustration of the $\boldsymbol{u}_\mathrm{o}$ vector calculated for an invertible $\boldsymbol{B}$. the blue vectors show that action of $\boldsymbol{Ax}$, and the red and green vectors are actions of the first and second inputs respectively. That is, the red vector is $b_1 u_1$ and the green vector is $b_2 u_2$.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{Ax} + \boldsymbol{Bu} \\ &= \boldsymbol{Ax} + b_1 u_1 + b_2 u_2 \end{aligned}$$

u_1 acts along $b_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. u_2 acts along $b_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. The diagram shows the components of $\boldsymbol{u} = -\boldsymbol{B}^{-1}\boldsymbol{Ax}$ at each $\boldsymbol{x}$.

Finally, consider $\boldsymbol{A} = \begin{bmatrix} 0 & 0.2 \\ -0.2 & 0 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{Ax} + \boldsymbol{Bu} \\ &= \boldsymbol{Ax} + b_1 u_1 + b_2 u_2 + b_3 u_3 \end{aligned}$$

u_1 acts along $b_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. u_2 acts along $b_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. u_3 acts along $b_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The diagram shows the elements of $\boldsymbol{u} = -\boldsymbol{B}^\dagger \boldsymbol{Ax}$ at each $\boldsymbol{x}$.

Consider $\boldsymbol{A} = \begin{bmatrix} 0 & 0.2 \\ -0.2 & 0 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ as before. In this case the pseudoinverse gives us the control signal that does the best it can to counteract the changes due to $\boldsymbol{Ax}$. That is, it yields a $\boldsymbol{u}$ that counteracts $\boldsymbol{Ax}$ along the direction in which $\boldsymbol{u}$ can act ($[1\ 1]^\mathsf{T}$ in this case).

For a discrete time system we have

$$\boldsymbol{x}(t+1) = \boldsymbol{Ax}(t) + \boldsymbol{Bu}(t)$$

But at steady state we have $\boldsymbol{x}(t+1) = \boldsymbol{x}(t) = \boldsymbol{x}_\mathrm{o}$. So,

$$\begin{aligned} \boldsymbol{x}_\mathrm{o} &= \boldsymbol{Ax}_\mathrm{o} + \boldsymbol{Bu}_\mathrm{o} \\ (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x}_\mathrm{o} &= \boldsymbol{Bu}_\mathrm{o} \\ \text{So,} \quad \mathcal{E}_{\boldsymbol{x}} &= \left\{ (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{Bu}_\mathrm{o},\ \boldsymbol{u}_\mathrm{o} \in \mathbb{R}^m \right\} \\ \text{and if it exists, } \boldsymbol{u}_\mathrm{o} &= \boldsymbol{B}^{-1}(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x}_\mathrm{o} \end{aligned}$$

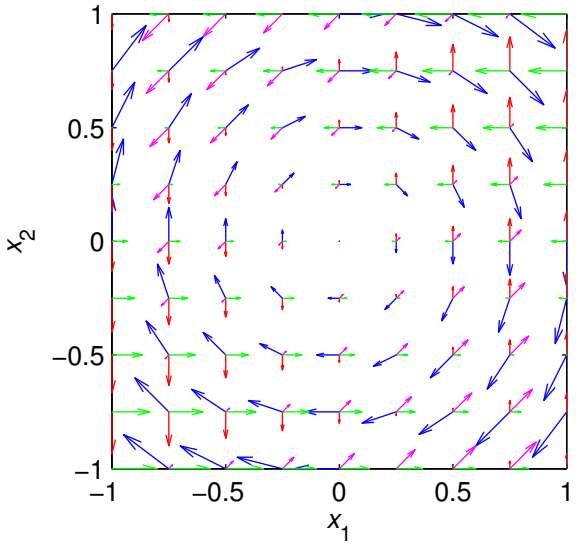


Figure 7.3: An illustration of the balance of changes calculated for a fat $\boldsymbol{B}$ using the pseudoinverse. This calculation yields the least norm $\boldsymbol{u}$ that will balance the motion of the autonomous system. The blue vectors indicate the effects of $\boldsymbol{Ax}$, red is b_1u_1, green is b_2u_2 and magenta is b_3u_3.

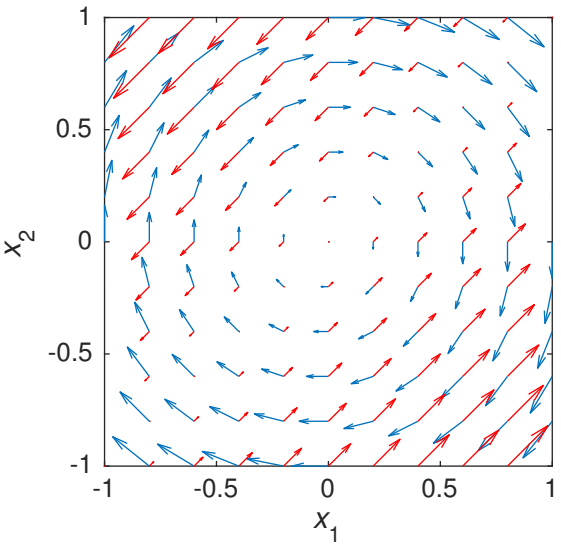


Figure 7.4: Calculation of a $\boldsymbol{u}$ using the pseudoinverse when $\boldsymbol{B}$ is skinny. Blue vectors show the action of $\boldsymbol{Ax}$ while the red vectors are the "best" possible values for $\boldsymbol{Bu}$.

We have the same considerations about invertibility of $\boldsymbol{B}$ as in the continuous time case.

We have previously discussed controlling nonlinear systems by using a linearising model that holds around the desired operating point. For these systems it is not generally true that the model at $\boldsymbol{x} = 0$ is the same as that at the desired operating point (or anywhere else for that matter). For nonlinear systems you need to use the complete nonlinear system equations to find an appropriate $\boldsymbol{u}_\mathrm{o}$. To complicate matters, depending on the particular nonlinearities that the system exhibits you may not end up at your desired $\boldsymbol{x}_\mathrm{o}$ if you apply $\boldsymbol{u}_\mathrm{o}$. You may need to carry the system along a particular trajectory to reach $\boldsymbol{x}_\mathrm{o}$, after which time your linearised controller can take over. For nonlinear systems the notion of an equilibrium subspace doesn't make any sense, so don't try to find that using the equations given above.

7.2 Compensator Topology

The basic structure of our system with an added compensator is as shown in figure 7.5.

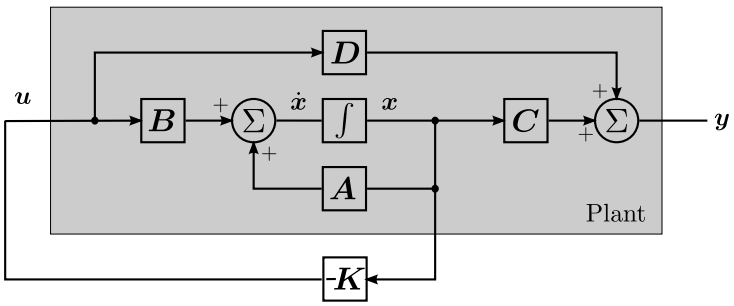


Figure 7.5: A regulated system using state feedback.

- The basic premise is that we use the current state vector to provide feedback to the system.
- The feedback matrix $\boldsymbol{K}$ describes *how* the state variables are fed back to the various inputs.
- $\boldsymbol{K} \in \mathbb{R}^{m \times n}$, where n is the number of states and m the number of system inputs.
- $\boldsymbol{K}$ is time-invariant for the systems we will consider, but it could vary in an adaptive control system.
- In a classical control problem we apply feedback from only the single output. It is not unreasonable to expect that we will have more flexibility with the state space structure.
- We assume that we have perfect knowledge of the current internal state of the system. This is an unrealistic assumption which we will address later.
- It is simply a convention that the feedback matrix is $-\boldsymbol{K}$ rather than $\boldsymbol{K}$.

7.2.1 An overview of state space compensator design

We have three main tasks in designing a state space compensator.

1. How do we determine whether it is possible to control a given system?
   - Can the system be driven to an arbitrary point in state space?
   - Can the system be stabilised at some operating point in state space?
2. How do we choose an appropriate $\boldsymbol{K}$ matrix to produce the desired system dynamic response? For now we will tackle this question by determining how to move the closed loop system poles to a desired location.
3. How do we choose where we should place the system poles?

7.3 Controllability

A system is uncontrollable when the input(s) have no effect on one or more of the state variables. That is, the system is uncontrollable because the input(s) are decoupled from at least one state variables. A system is also uncontrollable if we can't independently drive the different state variables to arbitrary locations (more on this shortly). As there is no guarantee that it is possible to control a given system, we need a mathematical test that tells us when a system is uncontrollable. You should *always* check whether a system is controllable before attempting to design a controller.

Here $\boldsymbol{A} := \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ 0 & \boldsymbol{A}_{22} \end{bmatrix}$, $\boldsymbol{B} := \begin{bmatrix} \boldsymbol{B}_1 \\ 0 \end{bmatrix}$, $\boldsymbol{x} := \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$. Notice that $\boldsymbol{u}$ cannot affect $\boldsymbol{x}_2$, so the plant is uncontrollable.

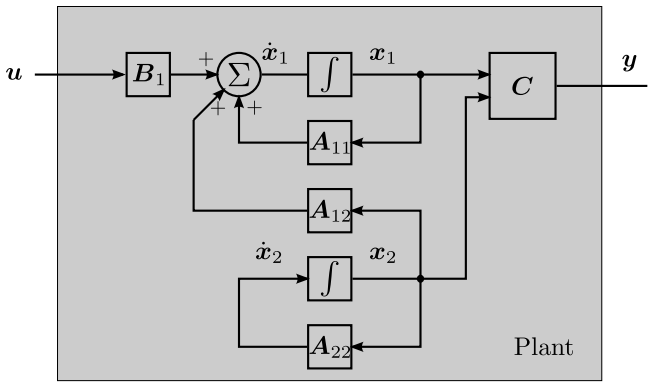


Figure 7.6: Internal structure of an uncontrollable system. The subset of state variables x_2 is not controllable from the inputs.

There are many different possible definitions for system controllability. Several are in common use, but for continuous-time systems they are happily equivalent! There are circumstances (eg discrete time) when the equivalence does not hold, but we will not be too worried about the details. We will consider a system controllable if we can drive the system from an initial state $\boldsymbol{x}(0) = 0$ to an arbitrary state $\boldsymbol{x}(\tau)$. Strictly speaking this is known as *reachability* and controllability is the ability to drive an arbitrary $\boldsymbol{x}(0)$ to $\boldsymbol{x}(\tau) = 0$. The two notions are equivalent for CT systems and also for most DT systems (other than pathological systems like $\boldsymbol{x}(t+1) = 0$). Controllability doesn't place any restrictions on the behaviour of the system outputs. It is purely about driving the system's internal state to a specified point.

7.3.1 Controllability of discrete-time systems

Consider a discrete time SISO system described by $\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t)$ with $\boldsymbol{x}(0) = 0$. We can write expressions for the state of the system at various times.

$$
\begin{aligned}
\boldsymbol{x}(1) &= \boldsymbol{B}u(0) \\
\boldsymbol{x}(2) &= \boldsymbol{A}\boldsymbol{x}(1) + \boldsymbol{B}u(1) \\
&= \boldsymbol{A}\big(\boldsymbol{B}u(0)\big) + \boldsymbol{B}u(1) \\
&= \boldsymbol{A}\boldsymbol{B}u(0) + \boldsymbol{B}u(1) \\
\boldsymbol{x}(3) &= \boldsymbol{A}\boldsymbol{x}(2) + \boldsymbol{B}u(2) \\
&= \boldsymbol{A}\big(\boldsymbol{A}\boldsymbol{B}u(0)\big) + \boldsymbol{A}\boldsymbol{B}u(1) + \boldsymbol{B}u(2) \\
&= \boldsymbol{A}^2\boldsymbol{B}u(0) + \boldsymbol{A}\boldsymbol{B}u(1) + \boldsymbol{B}u(2) \\
&\vdots \\
\boldsymbol{x}(k) &= \boldsymbol{A}^{k-1}\boldsymbol{B}u(0) + \boldsymbol{A}^{k-2}\boldsymbol{B}u(1) + \boldsymbol{A}^{k-3}\boldsymbol{B}u(2) + \cdots + \boldsymbol{B}u(k-1)
\end{aligned}
$$

$$
\begin{bmatrix} \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \boldsymbol{x}(3) \\ \vdots \\ \boldsymbol{x}(k) \end{bmatrix} = \begin{bmatrix} & & & & \boldsymbol{B} \\ & & & \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} \\ & & \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} \\ & \iddots & \iddots & \iddots & \vdots \\ \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^k\boldsymbol{B} \end{bmatrix} \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ \vdots \\ u(0) \end{bmatrix}
$$

If we concentrate simply on the expression for $\boldsymbol{x}(k)$ we get

$$\boldsymbol{x}(k) = \begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^k\boldsymbol{B}\end{bmatrix} \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ \vdots \\ u(0) \end{bmatrix} \equiv \boldsymbol{\mathcal{M}}_{\mathrm{c}} \begin{bmatrix} u(k-1) \\ u(k-2) \\ u(k-3) \\ \vdots \\ u(0) \end{bmatrix}$$

where $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$, the *controllability matrix*, is defined as $\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots\end{bmatrix}$.

If we want to choose a sequence of u values to achieve a certain $\boldsymbol{x}(k)$, then we could invert this expression.

$$\begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(1) \\ u(0) \end{bmatrix} = \boldsymbol{\mathcal{M}}_{\mathrm{c}}^{-1}\boldsymbol{x}(k)$$

That is, iff $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ is invertible then we can choose $\boldsymbol{u}$ to bring about our desired $\boldsymbol{x}(k)$. Note that $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ can only be invertible for a SISO system, as this is the only situation in which it can be square. In the general MIMO case, we will see that the structure of this special matrix $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ is fundamental to discussions of the set of states that can be reached by a state space system.

Consider a discrete-time LTI system described by

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{x}(0) &= 0 \\ \boldsymbol{x} \in \mathbb{R}^n,\, t &\in \mathbb{Z}_+ \end{aligned}$$

We want to determine the complete set of possible values for $\boldsymbol{x}(t)$, as a function of t. If $\boldsymbol{x}(t)$ can (eventually) take any value in $\mathbb{R}^n$ then the system is controllable. We can write an expression for $\boldsymbol{x}(1)$:

$$\begin{aligned} \boldsymbol{x}(1) &= \boldsymbol{A}\boldsymbol{x}(0) + \boldsymbol{B}\boldsymbol{u}(0) \\ &= \boldsymbol{B}\boldsymbol{u}(0) \end{aligned}$$

At $t = 1$ the system can have any value that is in the range of $\boldsymbol{B}$.

$\boldsymbol{x}(1) \in \mathcal{R}\boldsymbol{B}$, so if there are fewer inputs than system states then we will not be able to reach every possible state in $\mathbb{R}^n$ in one time step. If we do have enough inputs then we can reach every state only if $\boldsymbol{B}$ is full rank. Recall the example for $\boldsymbol{x} \in \mathbb{R}^2$ with $\boldsymbol{B} = \begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix}$. If we generate random samples for $\boldsymbol{u}(0)$ we find that the effect on $\boldsymbol{x}(1)$ is one dimensional, namely $\mathcal{R}\boldsymbol{B}$. This ought not surprise us, as $\boldsymbol{B}$ is clearly not full rank because its columns are not independent. Figure 7.7 shows how the random samples for $\boldsymbol{u}(0)$ map to $\boldsymbol{x}(1)$.

After an additional time step we have the following:

$$\begin{aligned} \boldsymbol{x}(2) &= \boldsymbol{A}\boldsymbol{x}(1) + \boldsymbol{B}\boldsymbol{u}(1) \\ &= \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{B}\boldsymbol{u}(1) \end{aligned}$$

We chose a $\boldsymbol{u}(0)$, which resulted in $\boldsymbol{x}(1) \in \mathcal{R}\boldsymbol{B}$. The effect of $\boldsymbol{u}(0)$ is now in $\mathcal{R}(\boldsymbol{AB})$. In addition we have a $\boldsymbol{u}(1)$, the effect of which is in $\mathcal{R}\boldsymbol{B}$. That is,

$$\begin{aligned} \boldsymbol{x}(2) &\in \mathcal{R}(\boldsymbol{B}) \cup \mathcal{R}(\boldsymbol{AB}) \\ &\in \mathcal{R}\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB}\end{bmatrix} \end{aligned}$$

That is, $\boldsymbol{x}(2)$ is in a space spanned by the columns of $\boldsymbol{B}$ plus the columns of $\boldsymbol{AB}$. This is equivalent to saying that it is spanned by a matrix made by concatenating $\boldsymbol{B}$ and $\boldsymbol{AB}$.

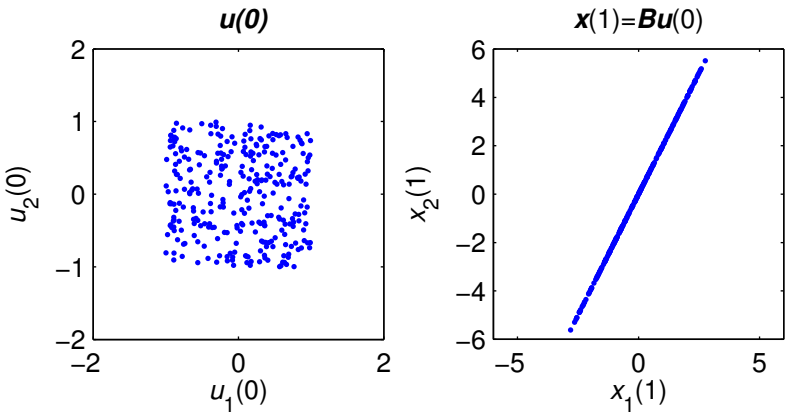


Figure 7.7: Effect of control with a rank deficient $\boldsymbol{B}$ after one time step. The left subfigure shows a random sample of 300 possible $\boldsymbol{u}(0)$ drawn from a uniform distribution between -1 and 1. The right subfigure shows how the various $\boldsymbol{u}$ values are mapped into $\boldsymbol{x}(1)$ by the action of $\boldsymbol{B}$. In particular notice that the result spans a one dimensional subspace of $\mathbb{R}^2$.

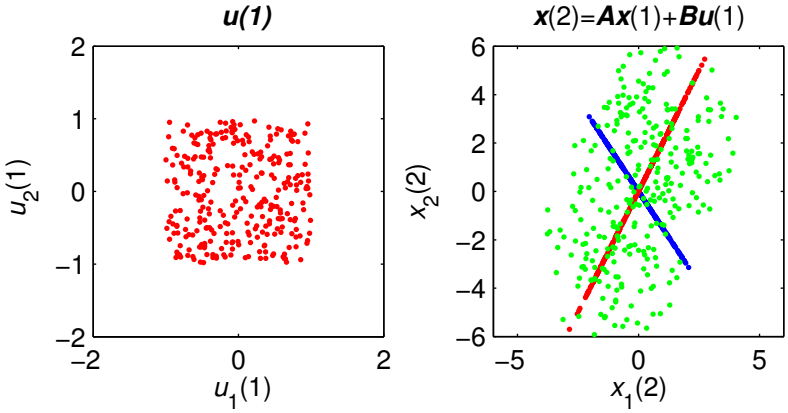


Figure 7.8: The left subfigure shows 300 random samples for $\boldsymbol{u}(1) \sim \mathcal{U}(-1, 1)$. The right subfigure shows the effect of $\boldsymbol{u}(1)$ on $\boldsymbol{x}(2)$ in red and the effect of $\boldsymbol{u}(0)$ and blue. The green points are produced by summing one of the red points with one of the blue. That is, the green points show the $\boldsymbol{x}(2)$ values reached by using a random $\boldsymbol{u}(0)$ followed by another random $\boldsymbol{u}(1)$.

Returning to our example in $\mathbb{R}^2$, at $t = 2$ we can see in figure 7.8 that the effect of $u(0)$ has now been swept along by the action of $\boldsymbol{A} = \begin{bmatrix} 0.6\cos(\pi/3) & -0.6\sin(\pi/3) \\ 0.6\sin(\pi/3) & 0.6\cos(\pi/3) \end{bmatrix}$. We can see that this is still a one dimensional subspace, $\mathcal{R}(\boldsymbol{AB})$. The effect of the random samples for $\boldsymbol{u}(1)$ spans the same subspace $\mathcal{R}\boldsymbol{B}$ as before. However, the sum of a point from each now spans $\mathbb{R}^2$.

$$
\begin{aligned}
\boldsymbol{x}(3) &= \boldsymbol{A}\boldsymbol{x}(2) + \boldsymbol{B}\boldsymbol{u}(2) \\
&= \boldsymbol{A}\big(\boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{B}\boldsymbol{u}(1)\big) + \boldsymbol{B}\boldsymbol{u}(2) \\
&= \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(1) + \boldsymbol{B}\boldsymbol{u}(2) \\
\text{So,} \quad \boldsymbol{x}(3) &\in \mathcal{R}\begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} \end{bmatrix}
\end{aligned}
$$

We can write a general equation for the state at time t

$$
\begin{aligned}
\boldsymbol{x}(t) = \boldsymbol{A}^{t-1}\boldsymbol{B}\boldsymbol{u}(0) + \boldsymbol{A}^{t-2}\boldsymbol{B}\boldsymbol{u}(1) + \boldsymbol{A}^{t-3}\boldsymbol{B}\boldsymbol{u}(2) \\
+ \cdots + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(t-2) + \boldsymbol{B}\boldsymbol{u}(t-1)\,,
\end{aligned}
$$

for which the reachable subspace is

$$\mathcal{R}\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^{t-1}\boldsymbol{B}\end{bmatrix} .$$

It would appear that the range of the matrix

$$\mathcal{R}\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^{t-1}\boldsymbol{B}\end{bmatrix}$$

continues to grow as t grows. However, the Cayley-Hamilton theorem allows us to write powers of a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ at or beyond the n-th as a linear combination of lower powers of $\boldsymbol{M}$. That is, from $t = n$ the $\boldsymbol{A}^t$ matrices are linearly dependent on lower powers of $\boldsymbol{A}$. The range of the matrix can therefore not increase after $t = n - 1$.

We define the controllability matrix as

$$\boldsymbol{\mathcal{M}}_{\mathrm{c}} := \begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B}\cdots & \boldsymbol{A}^{n-1}\boldsymbol{B}\end{bmatrix} .$$

$\mathcal{R}(\boldsymbol{\mathcal{M}}_{\mathrm{c}})$ is the entire set of possible $\boldsymbol{x}$ that the system can eventually be driven to with *some* choice of $\boldsymbol{u}$. If the system is controllable then we can drive $\boldsymbol{x}$ to an arbitrary point in $\mathbb{R}^n$. That is, we require $\mathcal{R}(\boldsymbol{\mathcal{M}}_{\mathrm{c}}) = \mathbb{R}^n$. As $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ has n rows, a system is controllable iff $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ is full rank.

7.3.2 Reachability in time t

It is sometimes useful to consider the subspace that can be reached (or controlled) in some given t. That is, we can define the reachable subspace in time t as

$$\mathcal{R}_t := \mathcal{R}(\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \cdots & \boldsymbol{A}^{t-1}\boldsymbol{B}\end{bmatrix})$$

Notice that the reachable/controllable subspace can never shrink as time increases, though eventually it stops growing.

$$\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \cdots \subseteq \mathcal{R}_{n-1} = \mathcal{R}\boldsymbol{\mathcal{M}}_{\mathrm{c}}$$

7.3.3 Controllability of continuous-time systems

For a continuous time system with $\boldsymbol{x}(0) = 0$ we have

$$\begin{aligned}\boldsymbol{x}(t) &= \int_0^t \boldsymbol{\Phi}(\tau)\boldsymbol{B}\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau \\ &= \int_0^t \mathrm{e}^{\tau\boldsymbol{A}}\boldsymbol{B}\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau\end{aligned}$$

Again, we would like to find the subset of possible $\boldsymbol{x}(t)$ values for an arbitrary choice of $\boldsymbol{u}(t) \in [0, t] \to \mathbb{R}^m$. That is $\boldsymbol{u}$ is an arbitrary function between times 0 and t. Let's expand the matrix exponential into its series form

$$\begin{aligned}\boldsymbol{x}(t) &= \int_0^t \left(\boldsymbol{I} + \tau\boldsymbol{A} + \frac{1}{2}\tau^2\boldsymbol{A}^2 + \cdots\right)\boldsymbol{B}\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau \\ &= \int_0^t \left(\boldsymbol{B} + \tau\boldsymbol{AB} + \frac{1}{2}\tau^2\boldsymbol{A}^2\boldsymbol{B} + \cdots\right)\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau\end{aligned}$$

$$\begin{aligned}\boldsymbol{x}(t) &= \int_0^t \left(\boldsymbol{B} + \tau\boldsymbol{AB} + \frac{1}{2}\tau^2\boldsymbol{A}^2\boldsymbol{B} + \cdots\right)\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau \\ &= \boldsymbol{B}\int_0^t \boldsymbol{u}(t-\tau)\,\mathrm{d}\tau + \boldsymbol{AB}\int_0^t \tau\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau \\ &\quad + \boldsymbol{A}^2\boldsymbol{B}\int_0^t \frac{\tau^2}{2}\boldsymbol{u}(t-\tau)\,\mathrm{d}\tau + \cdots\end{aligned}$$

We don't need to worry too much about the details of this integral. Each of the integrals will evaluate to a vector of constants, so the important point is that $\boldsymbol{x}(t)$ will be a linear combination of the columns of $\boldsymbol{A}$, $\boldsymbol{AB}$, $\boldsymbol{A}^2\boldsymbol{B}$ etc. Again the reachable values of $\boldsymbol{x}(t)$ will be given by the range of the controllability matrix $\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^{n-1}\boldsymbol{B}\end{bmatrix}$.

One test for the rank of a matrix is to find the largest block inside the matrix that has non-zero determinant. The size of that submatrix is the rank of the matrix. A SISO system will always have a square controllability matrix, so one way to check its controllability is to find the determinant of the controllability matrix. Any square matrix that is not full rank will have a determinant equal to zero. That is, for controllability of a system with square $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$, check that

$$\boxed{\left|\boldsymbol{\mathcal{M}}_{\mathrm{c}}\right| \neq 0}$$

In the case of a non-square controllability matrix you could check the determinant of $\boldsymbol{\mathcal{M}}_{\mathrm{c}}\boldsymbol{\mathcal{M}}_{\mathrm{c}}^{\mathsf{T}}$, since the rank of a matrix $\boldsymbol{A}$ is the same as $\boldsymbol{A}\boldsymbol{A}^{\mathsf{T}}$. An even easier test is to use Matlab's `rank(ctrb(A, B))`.

7.3.4 Controllable and Equilibrium Subspaces

The subspace of equilibrium points can be smaller than the reachable (or controllable) subspace. That is, it might be possible to reach parts of the state space, but not stay there. However, all parts of the equilibrium subspace must be contained within the controllable subspace.

$$\mathcal{E}_{\boldsymbol{x}} \subseteq \mathcal{R}(\boldsymbol{\mathcal{M}}_{\mathrm{c}})$$

7.3.5 Controllability in practice

The controllability test tells us that every state variable can be influenced by the inputs to *some* extent, even if the effect is tiny. It does not guarantee that it is practical to control a system in that way. For example, if the coupling between the inputs and a state variable is weak then it may require an unrealistically large control effort to have the system show any discernable response.

You should regard a controllability test as a necessary, but not sufficient test of whether you can practically control a system. When you design a real system make sure that you check the results in simulation to see whether the control signals are sensible. Singular value decomposition (SVD) can be used to get an insight into how controllable a system is.

7.3.6 Stabilisability

A system is said to be *stabilisable* iff all its uncontrollable modes are stable. In a stabilisable system, any modes that we cannot control will naturally decay to zero anyway. That means that eventually the system state will be driven to zero, so we can stabilise the system at an operating point. This may or may not imply that the system is practical, depending on whether the natural behaviour of the uncontrolled modes is acceptable. All is not lost if a system is uncontrollable. Examine the system modes carefully to see whether the system will perform adequately anyway. Stabilisability is a weaker test than controllability; all controllable systems are stabilisable by definition, but the converse is not true.

7.4 Compensator Design

Recall that our system is described by $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$, $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$. We seek to design an appropriate feedback matrix $\boldsymbol{K}$ so that we can set $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$. We will now substitute for $\boldsymbol{u}$ in the $\dot{\boldsymbol{x}}$ equation and investigate the behaviour of the new system.

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}\\ \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}\\ \dot{\boldsymbol{x}} &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{x}\end{aligned}$$

Notice that this new system looks like a conventional state space system with the $\boldsymbol{A}$ matrix replaced by $\boldsymbol{A} - \boldsymbol{BK}$. That is, the evolution of the state is now determined by $\boldsymbol{A} - \boldsymbol{BK}$ rather than $\boldsymbol{A}$.

Recall that the open loop system poles were equal to the eigenvalues of $\boldsymbol{A}$. Similarly we can find the poles of the closed loop system as the eigenvalues of $(\boldsymbol{A} - \boldsymbol{BK})$. That is, the closed loop system poles will be the roots of the closed loop characteristic equation;

$$\boxed{\det\left(s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BK})\right) = 0}$$

As before, substituting the appropriate matrices will result in a polynomial in s. However, the entries of $\boldsymbol{K}$ will now appear in the various coefficients of the powers of s. If we know what we would like the characteristic polynomial to be, we can equate coefficients of the powers of s to determine $\boldsymbol{K}$.

7.4.1 Compensator design example

Let's look at an example drawn from Franklin, Powell and Emami-Naeini "Feedback Control of Dynamic Systems" 3rd ed, p 495.
Consider an undamped oscillator with frequency ω_0. This has a state space description of

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_0^2 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} u\ , \quad y = x_1 \quad .$$

We wish to apply feedback to the oscillator so that its closed loop poles move to $-2\omega_0$. That is, we wish to double the natural frequency of the oscillator and increase ζ to 1. First let's check the pole locations of the open loop system. We solve the characteristic equation of this system to find the pole locations.

$$\begin{aligned}\left|s\boldsymbol{I} - \boldsymbol{A}\right| = \begin{vmatrix}s & -1\\ \omega_0^2 & s\end{vmatrix} &= 0\\ \implies s^2 + \omega_0^2 &= 0\\ s &= \pm \mathrm{j}\omega_0\end{aligned}$$

As we would expect.
Figure 7.9 shows the open loop response of the oscillator. We can see that the two state variables are two sinusoidal signals $\pi/2$ out of phase. The output signal is equal to x_1.

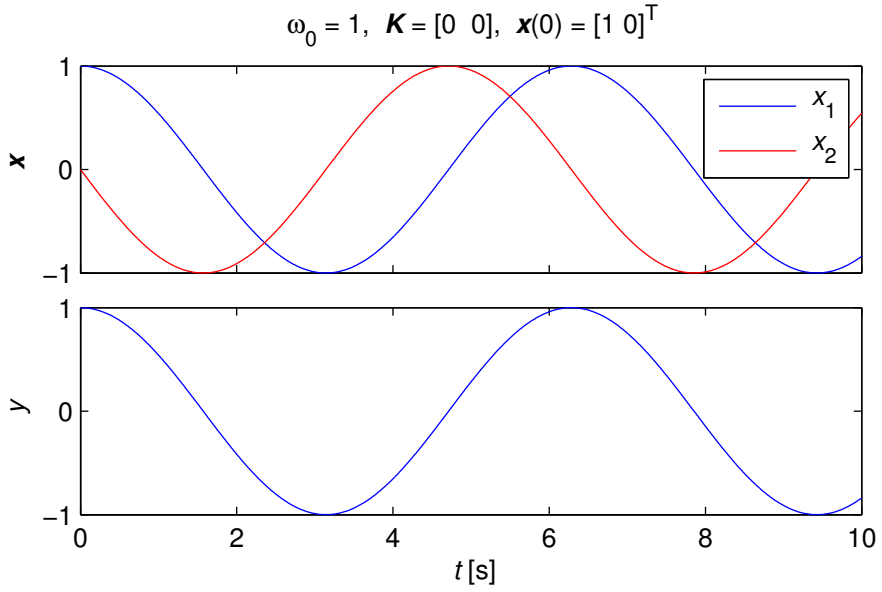


Figure 7.9: State and Output trajectories for an oscillator.

We would like both poles to move to $s = -2\omega_0 + \mathrm{j}0$. That is, we would like the characteristic polynomial of the closed loop system to be $(s + 2\omega_0)(s + 2\omega_0) = s^2 + 4\omega_0 s + 4\omega_0^2$. Let us compare the

desired characteristic equation with that of the closed loop system:

$$\begin{aligned}
|s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})| &= \det\left(\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} k_1 & k_2 \end{bmatrix}\right)\right) \\
&= \det\left(\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ k_1 & k_2 \end{bmatrix}\right) \\
&= \det\begin{bmatrix} s & -1 \\ \omega_0^2 + k_1 & s + k_2 \end{bmatrix} \\
&= s(s + k_2) + \omega_0^2 + k_1 \\
&= s^2 + k_2 s + \omega_0^2 + k_1
\end{aligned}$$

To find k_1 and k_2 we can equate the coefficients of the various powers of s in these two equations.

$$\begin{cases} s^1: & k_2 = 4\omega_0 \\ s^0: & \omega_0^2 + k_1 = 4\omega_0^2 \end{cases}$$

$$\begin{aligned}
\text{So,} \quad k_1 &= 3\omega_0^2 \\
k_2 &= 4\omega_0 \\
\implies \boldsymbol{K} &= \begin{bmatrix} 3\omega_0^2 & 4\omega_0 \end{bmatrix}.
\end{aligned}$$

Now $u = -\boldsymbol{K}\boldsymbol{x}$, so our compensator is realized by building a circuit (or writing a piece of software) to implement $u = 3\omega_0^2 x_1 + 4\omega_0 x_2$. Remember that ω_0 would be a constant in a real problem, so u is simply a linear function of $\boldsymbol{x}$.

The operation of our regulated system is shown in figure 7.10, which shows that the response of the state variables is now consistent with a pole pair at $s = -2\omega_0$. We can see that we have successfully achieved regulation of the output signal.

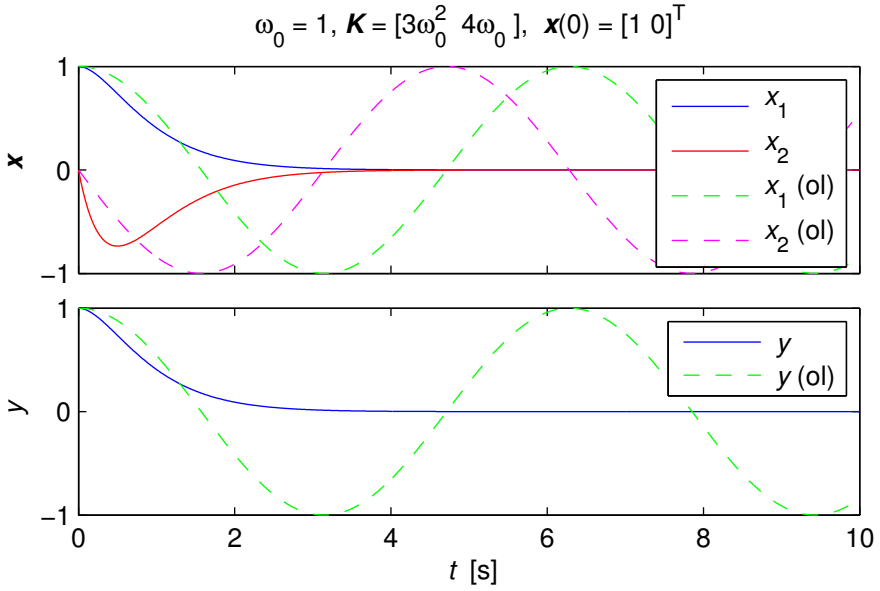


Figure 7.10: Performance of the damped oscillator system. The operation of the original system is shown in dashed lines for comparison.

7.4.2 Compensator Discussion

State space compensators such as this provide feedback from only the state variables and their derivatives. For example, the oscillator damping exercise above implemented a feedback rule of

$$u = 3\omega_0^2 x_1 + 4\omega_0 x_2 \equiv 3\omega_0^2 x_1 + 4\omega_0 \dot{x}_1$$

Such controllers are therefore simply a bank of PD compensators (or the degenerate possibility of P or D controllers). A PD controller does not increase the system type, so this simple state space controller may not be adequate for removing steady state errors (or errors in the presence of a ramp etc.). We will talk later about extending our regulator to add integral action.

7.4.3 Compensator design in control canonical form

It turns out that designing a compensator is trivial if the system is expressed in control canonical form.

In control canonical form we have matrices $\boldsymbol{A}_c = \begin{bmatrix} -d_{n-1} & -d_{n-2} & \cdots & -d_1 & -d_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$ and $\boldsymbol{B}_c = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

In this case the closed loop state matrix is

$$\boldsymbol{A}_c - \boldsymbol{B}_c \boldsymbol{K}_c = \begin{bmatrix} -d_{n-1} - k_1 & \cdots & -d_1 - k_{n-1} & -d_0 - k_n \\ 1 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}$$

The characteristic equation of this matrix will be

$$\det\left(s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\right) = 0$$
$$s^n + (d_{n-1} + k_1)s^{n-1} + \cdots + (d_0 + k_n) = 0$$

It is now easy to compare this to our desired characteristic equation,

$$\alpha_c = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n$$

Equating the two we get a system of n simultaneous equations to find the various k_i:

$$\begin{aligned} \alpha_1 &= d_{n-1} + k_1 \implies k_1 = \alpha_1 - d_{n-1} \\ \alpha_2 &= d_{n-2} + k_2 \implies k_2 = \alpha_2 - d_{n-2} \\ &\vdots \qquad\qquad\qquad\quad \vdots \\ \alpha_n &= d_0 + k_n \implies k_n = \alpha_n - d_0 \end{aligned}$$

This gives us an equation for $\boldsymbol{K}_c$ in the control canonical form.

In general, when we transform a system we must transform $\boldsymbol{K}$ as well. $\boldsymbol{K}$ provides feedback from the state vector, so the feedback will cease working if we change the definition of the state vector without changing $\boldsymbol{K}$. As before, consider a change of state vector from $\boldsymbol{x}$ to $\boldsymbol{z}$ according to $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{z}$. We wish to find $\bar{\boldsymbol{K}}$, the realisation of $\boldsymbol{K}$ appropriate to the new state vector. The input signal $\boldsymbol{u}$ produced in the two realisations must be the same. Thus

$$\begin{aligned} \bar{\boldsymbol{K}}\boldsymbol{z} &= \boldsymbol{K}\boldsymbol{x} \\ &= \boldsymbol{K}\boldsymbol{P}\boldsymbol{z} \\ \implies \bar{\boldsymbol{K}} &= \boldsymbol{K}\boldsymbol{P} \end{aligned}$$

In the current situation we have designed $\boldsymbol{K}$ in the control canonical form. If we were given the system in that form then using $\boldsymbol{K}$ is fine, but if we began by transforming the state space transformation then we need to move $\boldsymbol{K}$ back to the initial representation with the formula above.

We now have a second technique for finding $\boldsymbol{K}$ which is useful when the order of the system is greater than 3. For small systems it is normally easier to simply equate the coefficients directly.

1. Put the system into control canonical form using a transformation matrix $\boldsymbol{P}^{-1}$.
2. Determine the required $\boldsymbol{K}$ values by inspection.
3. Convert $\boldsymbol{K}$ back to the initial realisation using $\boldsymbol{P}$.

7.4.4 Ackermann's formula

Direct comparison of the closed loop poles and the desired pole locations allows us to solve for $\boldsymbol{K}$ directly, however that method becomes messy as the order of the system increases. Often we would like to avoid converting the system to control canonical form also. *If we have a SISO system*, an alternative is to use Ackermann's formula. We will skip derivation of Ackermann's formula and simply state the result. Ackermann's formula provides us with a general process that can be used to choose $\boldsymbol{K}$. It basically automates shifting the system into control canonical form, finding $\boldsymbol{K}_c$ and then shifting back to the original realization.

Ackermann's formula states that

$$\boxed{\boldsymbol{K} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \boldsymbol{\mathcal{M}}_{\mathrm{c}}^{-1} \alpha_{\mathrm{c}}(\boldsymbol{A})}$$

where $\boldsymbol{\mathcal{M}}_{\mathrm{c}}$ is the controllability matrix and $\alpha_{\mathrm{c}}(\boldsymbol{A})$ is a function of the system matrix and is defined as

$$\boxed{\alpha_{\mathrm{c}}(\boldsymbol{A}) = \boldsymbol{A}^n + \alpha_1 \boldsymbol{A}^{n-1} + \alpha_2 \boldsymbol{A}^{n-2} + \cdots + \alpha_n \boldsymbol{I}}$$

where α_i are the coefficients of the desired characteristic equation. Notice that is the controllability matrix is "small" then $\boldsymbol{\mathcal{M}}_{\mathrm{c}}^{-1}$ will be large. As a consequence, a system that is only weakly controllable will require large gain.

As an example, we will redo the oscillator example that we encountered above using Ackermann's formula. If you look back you will see that our desired polynomial is $s^2 + 4\omega s + 4\omega_0^2$. Thus we have $\alpha_1 = 4\omega_0$ and $\alpha_2 = 4\omega_0^2$. Let's find α_{c} using the equation above:

$$\begin{aligned}
\alpha_{\mathrm{c}}(\boldsymbol{A}) &= \boldsymbol{A}^2 + 4\omega_0 \boldsymbol{A} + 4\omega_0^2 \boldsymbol{I} \\
&= \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + 4\omega_0 \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + 4\omega_0^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} -\omega_0^2 & 0 \\ 0 & -\omega_0^2 \end{bmatrix} + 4\omega_0 \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + 4\omega_0^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 3\omega_0^2 & 4\omega_0 \\ -4\omega_0^3 & 3\omega_0^2 \end{bmatrix}
\end{aligned}$$

Now $\boldsymbol{\mathcal{M}}_{\mathrm{c}} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \implies \boldsymbol{\mathcal{M}}_{\mathrm{c}}^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ Substituting into Ackermann's formula, we get

$$\begin{aligned}
\boldsymbol{K} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \boldsymbol{\mathcal{M}}_{\mathrm{c}}^{-1} \alpha_{\mathrm{c}} \\
&= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3\omega_0^2 & 4\omega_0 \\ -4\omega_0^3 & 3\omega_0^2 \end{bmatrix} \\
\boldsymbol{K} &= \begin{bmatrix} 3\omega_0^2 & 4\omega_0 \end{bmatrix}
\end{aligned}$$

As before.

7.5 Choice of Pole Locations

Until now we have assumed that we knew where we wanted the closed loop poles to be. In reality we will rarely be told where the system poles should be. There are a number of strategies that we could adopt to locate the poles.

1. Select them manually to satisfy some damping or settling time requirement.
2. Use a prototype set of responses, such as ITAE, Butterworth or Bessel.
3. Use dead-beat control (discrete time only)
4. Use LQR techniques.

7.5.1 Manual selection of pole locations

The typical method of selecting poles is to arrange them such that there is a dominant pair of complex poles. These poles can be located to satisfy the system performance requirements. All other poles in the system should be placed considerably further into the left half of the s-plane so that they have negligible effect on the overall response (eg, their transient response will be over before that of the dominant poles). In practice this means that the non-dominant poles should be a factor of at least 3-5 further to the left of the s-plane. However, it is not that simple, as moving poles requires control effort. Moving poles a long way can require a larger servo for example. Therefore, try not to move poles further than necessary or poles that are close to zeros (these are hard to move).

7.5.2 Butterworth Configuration

A disadvantage of moving all but a dominant pair of poles towards the left of the s-plane (or z-plane) is that it takes a large control effort (large $\boldsymbol{K}$ and therefore $\boldsymbol{u}$). A system will tend to require less control effort if the closed loop poles are located at about the same distance from the origin, though this does make the response more sluggish. If we took this approach to the extreme we could place the closed loop poles in a Butterworth configuration, which in continuous time spaces the poles equally along a circle centered at the origin.

The following equations are the denominators of the transfer functions that generate the Butterworth configurations when $\omega_0 = 1$ (the numerator is one in each case).

$$
\begin{aligned}
&1 \quad s+1 \\
&2 \quad s^2 + \sqrt{2}s + 1 \\
&3 \quad (s+1)(s^2+s+1) \\
&4 \quad (s^2+0.7654s+1)(s^2+1.8478s+1) \\
&5 \quad (s+1)(s^2+0.6180s+1)(s^2+1.6180s+1) \\
&6 \quad (s^2+0.5176s+1)(s^2+1.4142s+1)(s^2+1.9319s+1)
\end{aligned}
$$

Matlab has good support for Butterworth filter design, so don't enter the pole locations manually. For continuous time use

```
[z, p, k] = butter(<order>, <cutoff freq>, 's')
```

For discrete time express the cutoff frequency as a fraction of the Nyquist frequency and use

```
[z, p, k] = butter(<order>, <cutoff freq>)
```

In each case the resulting poles can be found in `p`. Alternatively use `[n, d] = butter(...)` to get the characteristic polynomial in `d`.

7.5.3 ITAE responses for pole locations

An alternate way to proceed is to use the ITAE prototypes. These are pole locations calculated to minimise the **i**ntegral of the **t**ime multiplied by the **a**bsolute value of the **e**rror. That is, we use pole locations precalculated to minimize

$$J\big(u(t)\big) = \int_0^\infty t|e|\,\mathrm{d}t$$

with $e = y - y_{\text{desired}}$. We have two sets of these pole locations to choose from. The "normal" ITAE locations simply minimise J. The Bessel family of transfer functions also minimize the integral, but with the extra constraint that they exhibit no overshoot to a step input. The Bessel prototypes are slower than the ITAE responses, so use them only when necessary.

The following equations are the denominators of the transfer functions that generate the ITAE responses when $\omega_0 = 1$. For other ω_0, substitute s/ω_0 for s throughout.

$$\begin{aligned}
&1 \quad s + 1 \\
&2 \quad s + 0.7071 \pm 0.7071\mathrm{j} \\
&3 \quad (s + 0.7081)(s + 0.5210 \pm 1.068\mathrm{j}) \\
&4 \quad (s + 0.4240 \pm 1.2630\mathrm{j})(s + 0.6260 \pm 0.4141\mathrm{j}) \\
&5 \quad (s + 0.8955)(s + 0.3764 \pm 1.2920\mathrm{j})(s + 0.5758 \pm 0.5339\mathrm{j}) \\
&6 \quad (s + 0.3099 \pm 1.2634\mathrm{j})(s + 0.5808 \pm 0.7828\mathrm{j}) \\
&\quad \times (s + 0.7346 \pm 0.2873\mathrm{j})
\end{aligned}$$

$s + \alpha \pm \beta\mathrm{j}$ indicates that there is a pole pair at $-\alpha + \beta\mathrm{j}$ and $-\alpha - \beta\mathrm{j}$. Sadly, there appears to be no in-built Matlab function to produce the ITAE poles.

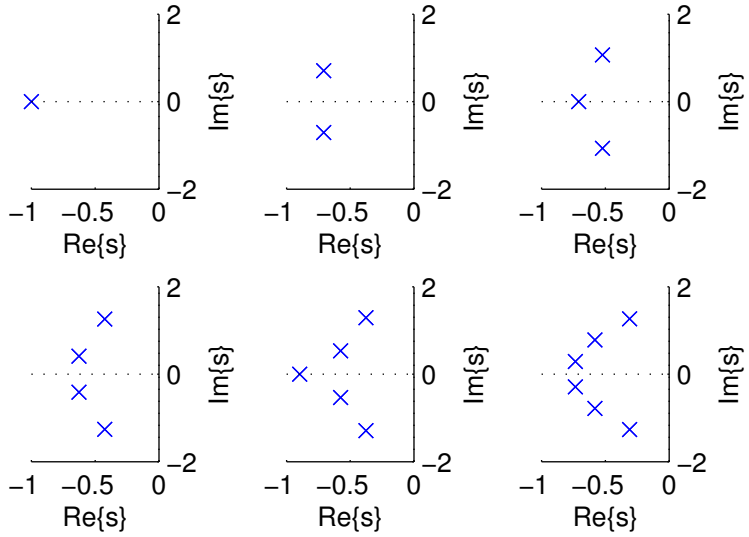


Figure 7.11: Pole locations for first to sixth order ITAE responses.

7.5.4 Bessel transfer functions

The following equations are the denominators of the transfer functions that generate the Bessel responses when $\omega_0 = 1$. For other ω_0, substitute s/ω_0 throughout.

$$
\begin{aligned}
1 \quad & s + 1 \\
2 \quad & s + 0.8660 \pm 0.5\text{j} \\
3 \quad & (s + 0.9420)(s + 0.7455 \pm 0.7112\text{j}) \\
4 \quad & (s + 0.6573 \pm 0.8302\text{j})(s + 0.9047 \pm 0.2711\text{j}) \\
5 \quad & (s + 0.9264)(s + 0.5906 \pm 0.9072\text{j})(s + 0.8516 \pm 0.4427\text{j}) \\
6 \quad & (s + 0.5385 \pm 0.9617\text{j})(s + 0.7998 \pm 0.5622\text{j}) \\
& \times (s + 0.9093 \pm 0.1856\text{j})
\end{aligned}
$$

$s + \alpha \pm \beta\text{j}$ indicates that there is a pole pair at $-\alpha + \beta\text{j}$ and $-\alpha - \beta\text{j}$. Use Matlab to get these: `[z,p,k] = besself(<order>,<cutoff frequency>)` will put the poles into `p`.

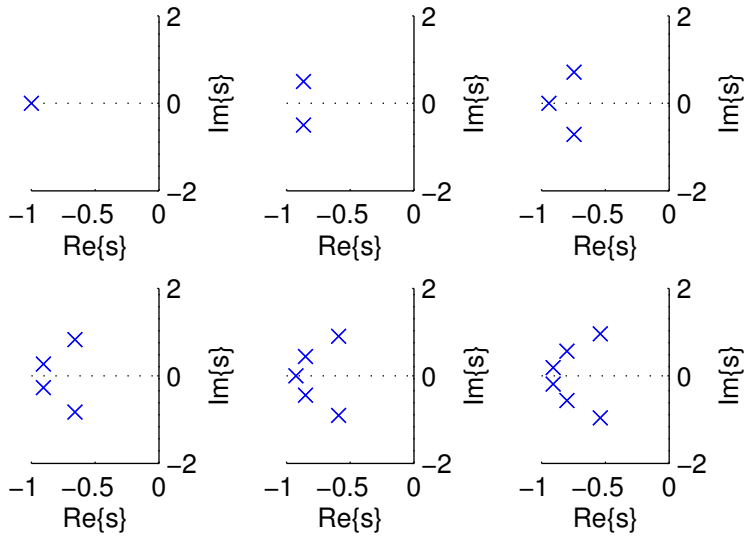


Figure 7.12: Pole-zero map for Bessel arrangements of first through sixth order.

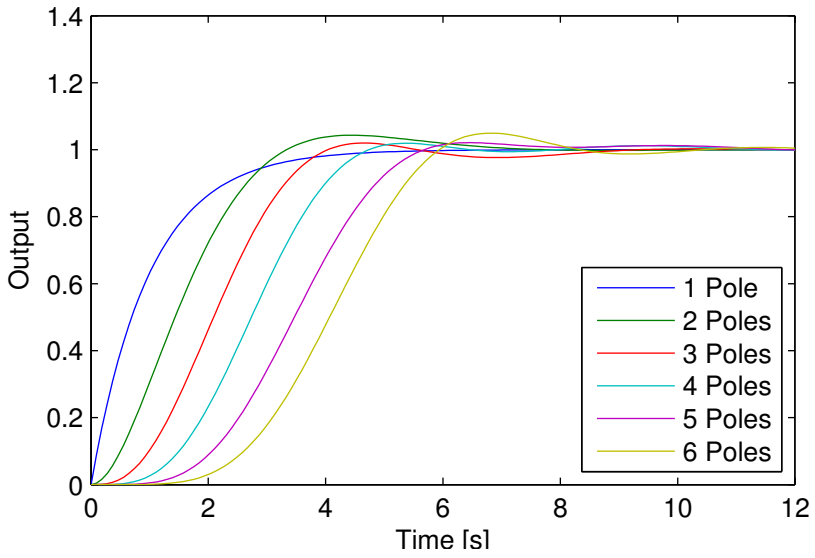


Figure 7.13: Step responses of different orders of ITAE pole arrangements.

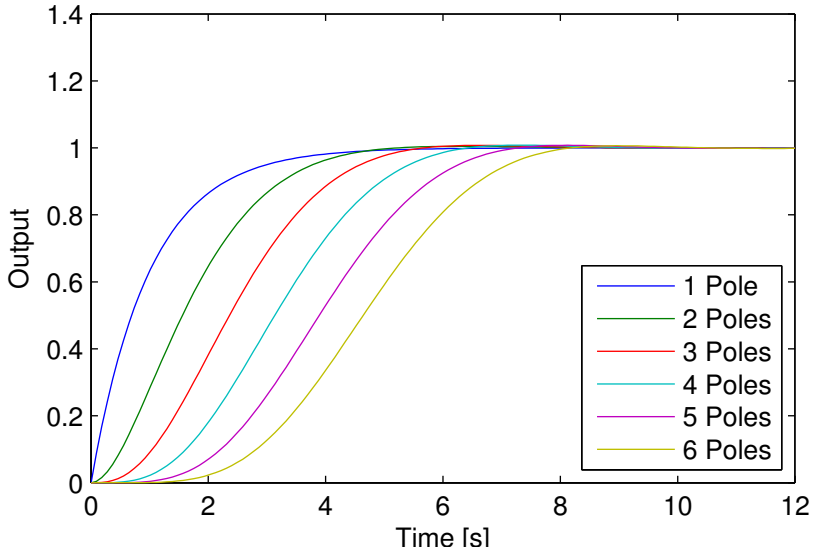


Figure 7.14: Step responses for different order Bessel pole configurations.

Figures 7.13 and 7.14 show the step responses of systems with ITAE and Bessel pole arrangements respectively.

In each case the order of the curves increases from left to right. Notice that the responses become more sluggish as we increase the poles in the same region.

7.5.5 Dead-beat Control

For a discrete time problem, a so called *dead-beat* controller can be used to drive the state to zero in the minimum time possible. That is, we can drive the state from some initial x_0 to zero in the smallest t for which $x_0 \in \mathcal{R}_t$. To build a dead-beat controller we simply place *all* of the closed loop poles at $z = 0$. Dead-beat controllers have good speed and zero steady sate error, but produce very large control signals. There is no analogue of dead-beat control for CT systems, as they can be driven to any point instantly.

7.5.6 Linear Quadratic Regulators

Notice that the magnitude of the required control effort was ignored in the ITAE approach. The ITAE prototypes can therefore lead to a controller design that produces very large control signals. A more complete approach is to include the magnitude of the control effort into our performance metric. Thus, instead of minimising $J\big(u(t)\big) = \int_0^\infty t|e|\,\mathrm{d}t$, we will add a term that penalises large values of u.
That is, we seek to minimize a function like

$$J\big(u(t)\big) = \int_0^\infty y^2(t) + \rho u^2(t)\,\mathrm{d}t$$

for a designer selected weighting ρ which indicates how "worried" the designer is about deviations in y and u. We will return to this idea when we look at *optimal control*.

7.6 Compensator Design Example

Consider a system described in state space by

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix} \quad .$$

Design a compensator such that the closed loop system has an overshoot of 6% or better and a settling time of 3 seconds. Recall from previous studies of second order systems that $\zeta = \dfrac{-\ln(\%OS/100)}{\sqrt{\pi^2 + \ln^2(\%OS/100)}}$ and $t_\mathrm{s} = \dfrac{4}{\zeta\omega_\mathrm{n}}$.

So, to achieve the specification we must have $\zeta = 0.67$ and $\omega_\mathrm{n} = 2$ rad/s, which corresponds to a dominant pole pair at $s = -1.33 \pm \mathrm{j}1.49$.

Our system is third order, so we must choose the location of a third closed loop pole. Let's place it a factor of ten further into the left half of the s-plane than the dominant poles (ie, at $s = -13.3$). Thus our desired characteristic polynomial is

$$(s + 1.33 + \mathrm{j}1.49)(s + 1.33 - \mathrm{j}1.49)(s + 13.3) = s^3 + 16s^2 + 39.55s + 53.26$$

We need to equate this to the characteristic equation of the closed loop system, $\det(s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BK}))$.

The expression for the closed loop characteristic polynomial is

$$\begin{aligned}
&\det(s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BK}) \\
&= \det\left(\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} \right) \\
&= \det\left(\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ k_1 & k_2 & k_3 \end{bmatrix} \right) \\
&= \begin{vmatrix} s & -1 & 0 \\ 0 & s & -1 \\ 18 + k_1 & 15 + k_2 & s + 2 + k_3 \end{vmatrix} \\
&= s^3 + (k_3 + 2)s^2 + (k_2 + 15)s + k_1 + 18
\end{aligned}$$

Equating the various coefficients of this characteristic equation with our desired characteristic equation yields

$$\begin{cases} s^2: & k_3 + 2 & = 16 \\ s^1: & k_2 + 15 & = 39.55 \\ s^0: & k_1 + 18 & = 53.26 \end{cases}$$

Solving for the various k values we obtain,

$$\boldsymbol{K} = \begin{bmatrix} 35.26 & 24.55 & 14 \end{bmatrix}$$

Let's compare the compensated closed loop and open loop responses to an impulse. The two responses are shown in in figure 7.15, which was produced with the matlab code

```
impulse(ss(A,B,C,D))
hold on
impulse(ss(A-B*K,B,C,D))
hold off
```

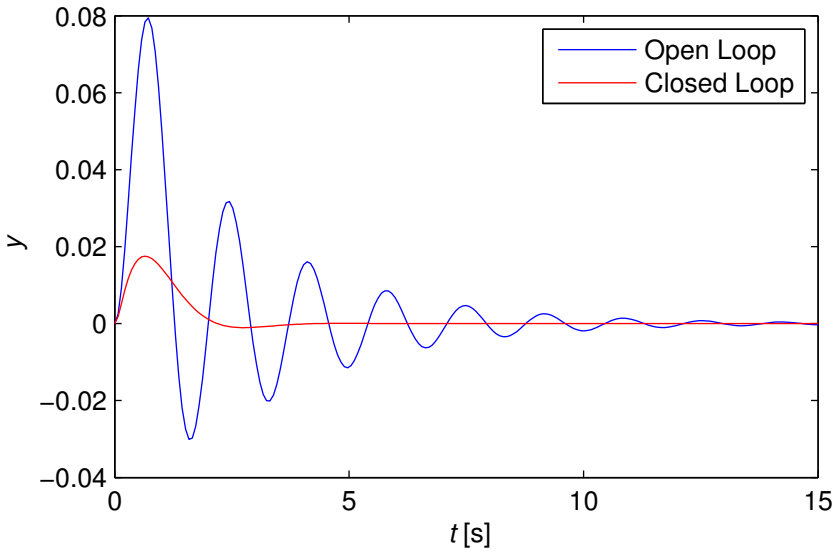


Figure 7.15: Comparison of the impulse responses of the open and closed loop (compensated) systems.

7.7 Matlab for Compensator Design

Matlab has two functions that can be used to find the required K vector to build a state-feedback controller; `acker` and `place`.

```
K = acker(A, B, P)
K = place(A, B, P)
```

Here `P` is a vector containing the list of desired pole locations. Note that there is a limit on the number of repeated roots allowed when you use `place` (see Matlab's help). However, `place` has better numerical behaviour than `acker`, so use `place` whenever you can. Note: `acker` is only suitable for SISO systems.

So, to repeat our previous example using matlab, we write:

```
K= place(A, B, [-1.33+1.49j, -1.33-1.49j, -13.3] )
>> K = 35.0537 24.3670 13.9600
```

This results in the feedback gain matrix, $\boldsymbol{K} = \begin{bmatrix}35 & 24.4 & 14\end{bmatrix}$, which is in good agreement with our manual design.

Consider the temperature control apparatus shown in figure **??**.

The state variables $x_{1..3}$ represent the temperatures in regions one to three. We aim to design a temperature controller for region three of the apparatus using the heater/cooler. That is, we want to be able to regulate x_3 to be any desired value using only u_3. We will use $t_s = 0.1$ s. Transients in temperature should decay in 20 seconds (or less).

The temperature at several locations within the apparatus can be approximated by the discrete time state space model

$$\boldsymbol{x}(t+1) = \begin{bmatrix} 1-\alpha t_s - \beta t_s & \beta t_s & 0 & 1 \\ \beta t_s & 1-2\beta t_s & \beta t_s & 0 \\ 0 & \beta t_s & 1-\beta t_s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} \alpha t_s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \beta t_s & 0 \\ 0 & 0 & \gamma \end{bmatrix} \boldsymbol{u}$$

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

$$\text{where, } \begin{cases} \alpha = 0.01 & \text{is the rate of heat flow between metal and air,} \\ \beta = 0.1 & \text{is the rate of heat flow between metal volumes,} \\ \gamma = 0.01 & \text{is coupling for the heater/cooler actuator.} \end{cases}$$

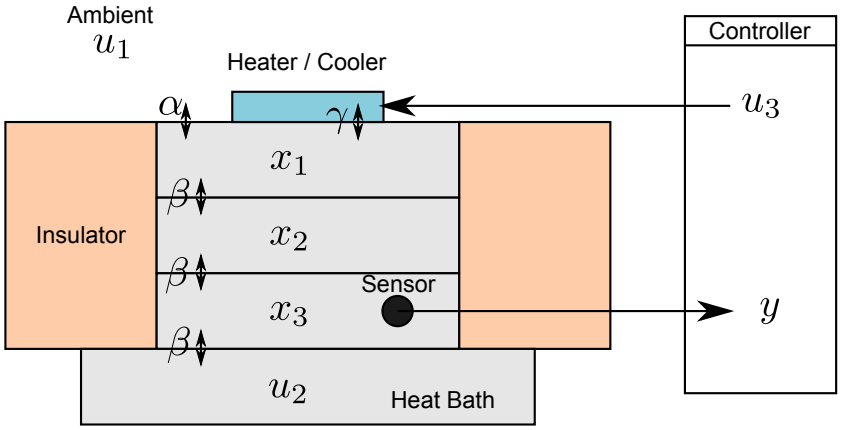


Figure 7.16: A schematic of the temperature control apparatus

Note that u_1 represents ambient temperature and u_2 is the temperature of the heat-bath. These represent real inputs to our systems, but we cannot use them to apply feedback. When designing your controller, we can only vary the heater/cooler power (u_3), so we will need to use a modified $\boldsymbol{B}$ during controller design.

$$\boldsymbol{x}(t+1) = \begin{bmatrix} 0.989 & 0.01 & 0 & 1 \\ 0.01 & 0.98 & 0.01 & 0 \\ 0 & 0.01 & 0.98 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.01 \end{bmatrix} u_3 + \begin{bmatrix} 0.001 & 0 \\ 0 & 0 \\ 0 & 0.01 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

For convenience, let's define $\boldsymbol{B}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.01 \end{bmatrix}$ and $\boldsymbol{B}_2 = \begin{bmatrix} 0.001 & 0 \\ 0 & 0 \\ 0 & 0.01 \\ 0 & 0 \end{bmatrix}$.

We will simply ignore the existence of $\boldsymbol{B}_2$ during design, but use it again when building the closed loop system.

We know that we must achieve a closed loop settling time of 20 s. Recall from previous studies that the settling time is approximately four time constants for a first order system, so a continuous-time, dominant pole at $s = \frac{4}{20} = -0.2$. Would achieve the required settling. We would need the other three poles to be at least a factor of three faster, say at $s = -0.6, -0.6 \pm \mathrm{j}0.6$. We can find the z-plane equivalents using the transformation $z = \mathrm{e}^{st_\mathrm{s}}$ (or any other appropriate discretization method). In this case we obtain a desired dominant pole location of $\lambda_\mathrm{d} = \mathrm{e}^{-0.2\times0.1} = 0.98$ and other pole locations of $\lambda = \{0.94, 0.94 \pm \mathrm{j}0.06\}$. We should check the controllability of the system using $\boldsymbol{A}$ and $\boldsymbol{B}_1$. In this case the controllability matrix is rank four, so we can proceed with our design.

```
K = place(A, B_1, [0.98 0.94 0.94+j0.06 0.94-j0.06])
S_cl = ss(A - B_1 * K, B, C, D, ts)
```

Notice that we have coupled all three inputs to the system, so we will be able to simulate the effects of environmental changes. The dynamics of the closed loop system are only modified by applying feedback to u_3, so only $\boldsymbol{B}_1\boldsymbol{K}$ alters the open loop $\boldsymbol{A}$ matrix. The Matlab command `eig(A-B_1*K)` will quickly confirm that the `place` command has correctly placed the closed loop poles in the desired location. The closed loop response of the system is shown in figure **??**. In this case the system is driven by impulses in each of the inputs successively. The impulses have amplitudes of 10, 1 and 1, and each lasts for one sampling interval.

We could repeat the design using a deadbeat controller. In this case we wish to place all four closed loop poles at $z = 0$. We will not be able to use Matlab's `place`, because it can't deal with that many colocated poles. However, as this is a SISO problem we can use `acker`.

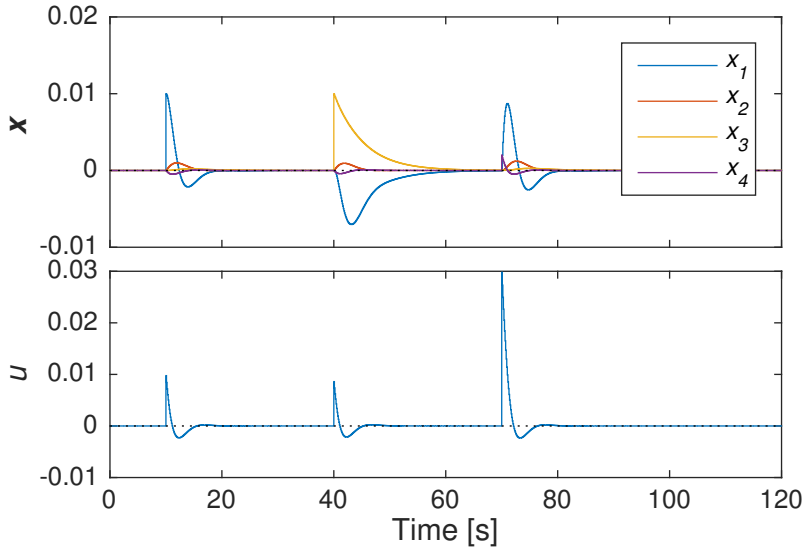


Figure 7.17: Impulse response of the closed loop system when driven by all three inputs successively.

```
K_db = acker(A, B_1, [0 0 0 0]);
S_db = ss(A - B_1 * K_db, B, C, D, ts);
```

Repeating the impulse response experiment from above yields the response shown in figure 7.18. Notice that each of the transients is complete within four sampling intervals. Though the amplitudes of the inputs in the two simulations were not identical, you can nevertheless get a sense that the control in the dead beat example is quite violent by comparison with the previous design.

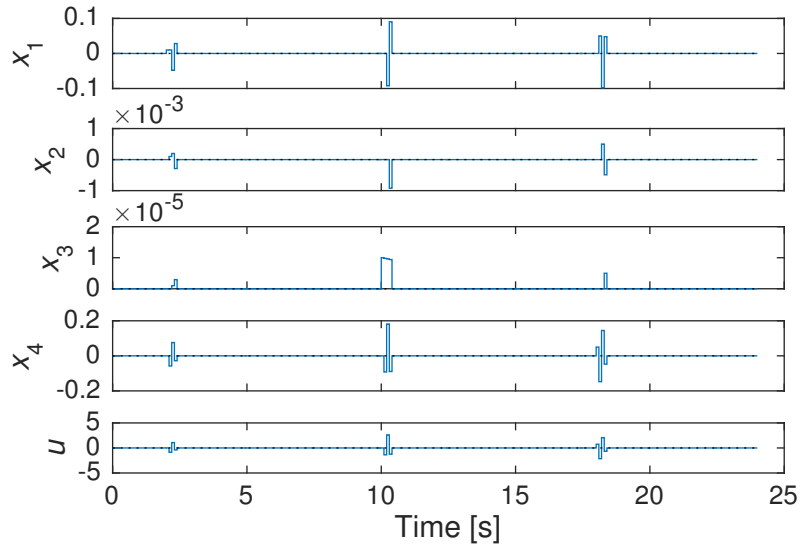


Figure 7.18: Impulse response of the deadbeat system when driven by all three inputs successively. The

7.8 Pole placement in MIMO systems

It is worth noting that `place` can find $\boldsymbol{K}$ for systems with multiple inputs and outputs. In these systems there can be more than one possible $\boldsymbol{K}$ matrix that will produce the desired pole placement. Matlab chooses a $\boldsymbol{K}$ which has low sensitivity to inaccurate implementation accuracy. So, if you use a controller generated by `place` you can be confident that it will work reasonably well in a real application.

However, there are circumstances where you might want to choose some other criteria on which to select from the possible $\boldsymbol{K}$ solutions. Common extra constraints that might be imposed on the structure of $\boldsymbol{K}$ include not applying feedback from a particular state variable (so that it need not be measured or estimated), not applying feedback to a particular actuator, minimising the number of non-zero elements of $\boldsymbol{K}$, or minimising the magnitude of the elements of $\boldsymbol{K}$. We will not cover these options in this course, in part because we will find that the LQR technique that we will look at later will allow us to tackle many of the same issues more elegantly. Nevertheless, you should be aware that there are techniques that allow those sorts of design flexibility.

As an example, let's reexamine our previous compensator design, but now add a second actuator that acts on x_2, so

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix} \quad .$$

We will design a controller that can drive the two inputs via some appropriate feedback matrix $\boldsymbol{K} = \begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \end{bmatrix}$.

We will again use the manual method of calculating the eigenvalues of the closed loop system and equating coefficients with the desired characteristic equation.

$$\begin{aligned} s^3 + 16s^2 + 39.55s + 53.26 &= \left| s\boldsymbol{I} - \boldsymbol{A} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \end{bmatrix} \right| \\ &= s^3 \\ &\quad + (k_3 + k_5 + 2)s^2 \\ &\quad\quad + (k_2 + k_4 + 2k_5 - 15k_6 - k_2k_6 + k_3k_5 + 15)s \\ &\quad\quad\quad + (k_1 + 2k_4 - 18k_6 - k_1k_6 + k_3k_4 + 18) \end{aligned}$$

We can now equate coefficients

$$\begin{cases} s^3 : 1 = 1 \\ s^2 : k_3 + k_5 + 2 = 16 \\ s^1 : k_2 + k_4 + 2k_5 - 15k_6 - k_2k_6 + k_3k_5 + 15 = 39.55 \\ s^0 : k_1 + 2k_4 - 18k_6 - k_1k_6 + k_3k_4 + 18 = 53.26 \end{cases}$$

$$\implies \begin{cases} s^3 : 1 = 1 \\ s^2 : k_3 + k_5 = 14 \\ s^1 : k_2 + k_4 + 2k_5 - 15k_6 - k_2k_6 + k_3k_5 = 24.55 \\ s^0 : k_1 + 2k_4 - 18k_6 - k_1k_6 + k_3k_4 + 18 = 35.26 \end{cases}$$

As we now have three equations in six unknowns there will be many possible solutions to this set of equations. We could make many choices for values of k_i depending on what we want to achieve.

For our example, let's imagine that we do not want to measure x_2, so we will not provide any feedback from that state variable. Therefore, let's set $k_2 = k_5 = 0$ as a starting point for $\boldsymbol{K}$.

$$\rightsquigarrow \begin{cases} k_3 = 14 \\ k_4 - 15k_6 = 24.55 \implies k_4 = 24.55 + 15k_6 \\ k_1 + 2k_4 - 18k_6 - k_1k_6 + k_3k_4 = 35.26 \end{cases}$$

Substituting the first two of these equations into the last, after a few lines of algebra we get

$$k_1 + (222 - k_1)k_6 = -357.54$$

Again we have a family of viable solutions. Let's arbitrarily choose $k_1 = 0$. We can now calculate the values for the remaining elements of $\boldsymbol{K}$.

$$\begin{aligned} & k_6 = -1.6105 \\ \implies & k_4 = 0.39189 \\ \text{and, } & \boldsymbol{K} = \begin{bmatrix} 0 & 0 & 14 \\ 0.39189 & 0 & -1.6105 \end{bmatrix} \end{aligned}$$

A check of the closed loop eigenvalues using this feedback matrix reveals that we have again achieved the required close loop pole positions, but as required we do not use feedback from x_2.

8 Servo Problems in State Space

We have previously discussed how to choose a feedback matrix $\boldsymbol{K}$ to stabilise a system in the presence of noise or system disturbance. That problem is known as a regulator problem, where the task is to regulate the system behaviour so that it remains constant. In this section we will consider the other main class of control problem, the servo or tracking problem. The requirement here is that the system output be made to follow an external command signal (known as the *reference* signal, $\boldsymbol{r}$). We will see that this task can be accomplished without changing any of our previous work. We will instead add an extra element to our overall compensator to produce the required tracking behaviour.

8.1 Introducing a Reference Input

There are a variety of ways we can introduce the reference into our system. The most obvious is to set the system input to $u = -\boldsymbol{K}\boldsymbol{x} + r$. This works badly, as it almost certainly results in a finite steady

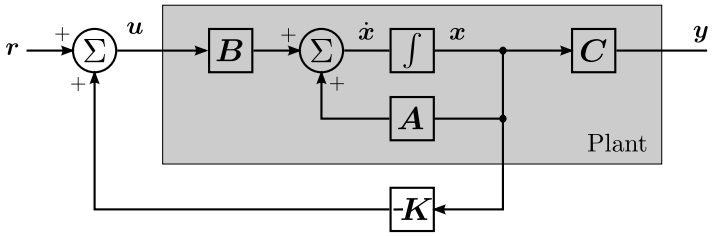


Figure 8.1: Naïve addition of a reference signal $\boldsymbol{r}$ to a state feedback controller.

state output error.

8.1.1 Reference Input – example

Let's plot the step response of our previous system $\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, $\boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix}$, $\boldsymbol{K} = \begin{bmatrix} 35.26 & 24.55 & 14 \end{bmatrix}$.

```
step(ss(A,B,C,D))
hold on
step(ss(A-BK,B,C,D))
hold off
```

The open loop and closed loop responses of this system are shown in figure 8.2 It is apparent from this figure adding feedback has altered the steady state output value.

We scale the input by including an extra block between the input and the real plant. This block $\bar{\boldsymbol{N}}$ simply provides the desired dc gain. To find $\bar{\boldsymbol{N}}$ in the SISO case we set $s = 0$ in the closed loop transfer function $\boldsymbol{C}\left(s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\right)^{-1}\boldsymbol{B}$ to find the dc gain of the system. We then set $\bar{\boldsymbol{N}}$ to add (or remove) any additional required gain. For this example we get $\bar{\boldsymbol{N}} = 2.96$.

```
step(ss(A,B,C,D))
hold on
step(ss(A-B*K,B*Nbar,C,D))
hold off
```

8.1.2 Introducing the reference input the right way

A better way to introduce the reference input is to calculate the required steady state $\boldsymbol{x}$ vector that will result in an output that matches the reference. Once we know our steady state $\boldsymbol{x}$ vector ($\boldsymbol{x}_{\mathrm{o}}$) we will

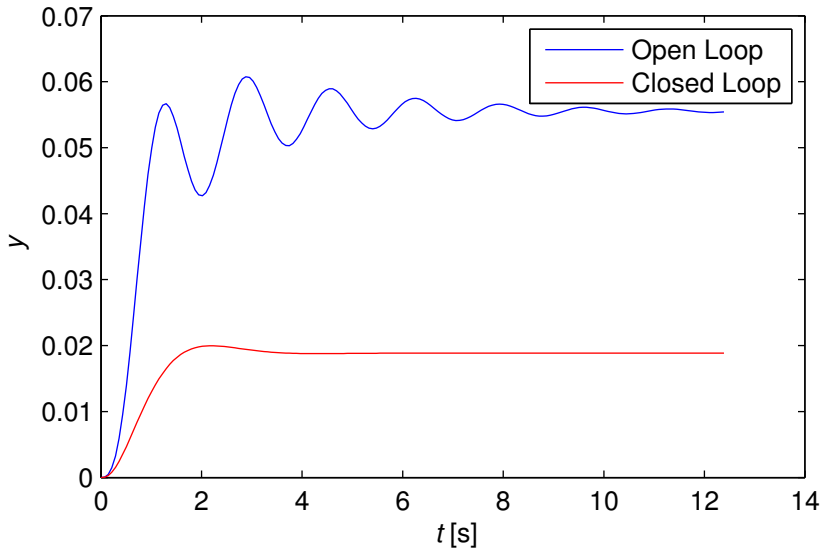


Figure 8.2: A compensated system with incorrect dc gain.

also know the required steady state input signal u_o. (Notice that we are considering a SISO system first for simplicity, though this approach can readily be extended.) The new input signal using this scheme will be

$$u = u_o - \boldsymbol{K}(\boldsymbol{x} - \boldsymbol{x}_o)$$

Now in the steady state, we should have $\dot{\boldsymbol{x}} = 0$, so our normal system equations can be simplified to

$$\begin{aligned} \mathbf{0} &= \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{B}u_o \\ y_o &= \boldsymbol{C}\boldsymbol{x}_o + Du_o \end{aligned}$$

We wish to arrange a system so that $y_o = r_o$. We have previously examined whether we can achieve a desired operating point, so for now we will simply assume that we can find some linear combination of the state vector and inputs that will result in this condition. That is, we presume that if we choose some appropriate $\boldsymbol{N}_x$ and N_u with $\boldsymbol{x}_o = \boldsymbol{N}_x r_o$ and $u_o = N_u r_o$ then we will achieve $y_o = r_o$. We can now substitute these expressions into our steady state system equations.

$$\begin{aligned} \mathbf{0} &= (\boldsymbol{A}\boldsymbol{N}_x + \boldsymbol{B}N_u)\, r_o \\ y_o &= (\boldsymbol{C}\boldsymbol{N}_x + \boldsymbol{D}N_u)\, r_o \\ \begin{bmatrix} \mathbf{0} \\ y_o \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ N_u \end{bmatrix} r_o \end{aligned}$$

However, at steady state we want $y_o = r_o$, so we can write

$$\begin{aligned} \begin{bmatrix} \mathbf{0} \\ r_o \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ N_u \end{bmatrix} r_o \\ \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ N_u \end{bmatrix} \\ \implies \begin{bmatrix} \boldsymbol{N}_x \\ N_u \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \end{aligned}$$

We can calculate $\boldsymbol{N}_x$ and N_u using this equation and use them to produce the new input for our system;

$$u_o = N_u r_o - \boldsymbol{K}(\boldsymbol{x}_o - \boldsymbol{N}_x r_o).$$

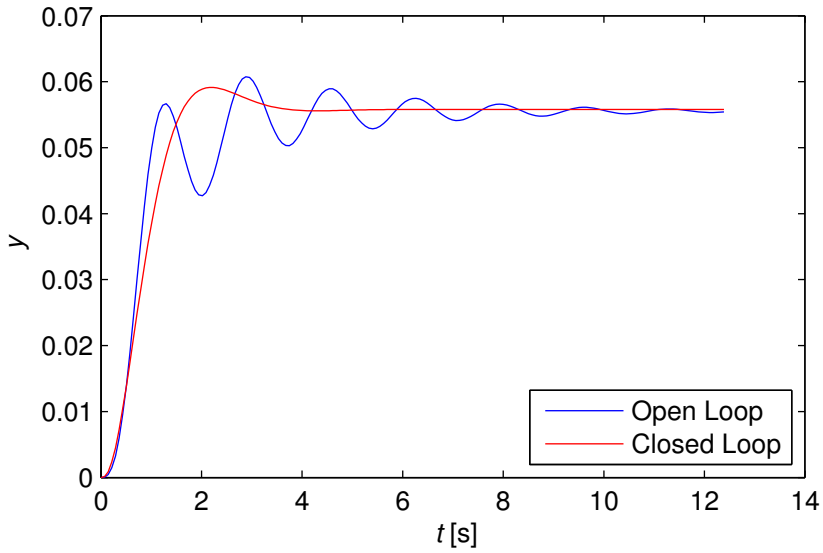


Figure 8.3: A compensated system where a prefilter has been used to correct the dc gain.

But $\boldsymbol{x}_o$ is not privileged for an LTI system, so this equation holds generally.

$$\begin{aligned} u &= N_u r - \boldsymbol{K}(\boldsymbol{x} - \boldsymbol{N}_x r) \\ &= -\boldsymbol{K}\boldsymbol{x} + (N_u + \boldsymbol{K}\boldsymbol{N}_x)r \end{aligned}$$

We can draw the total system in the following equivalent ways. For convenience we denote $N_u + \boldsymbol{K}\boldsymbol{N}_x$

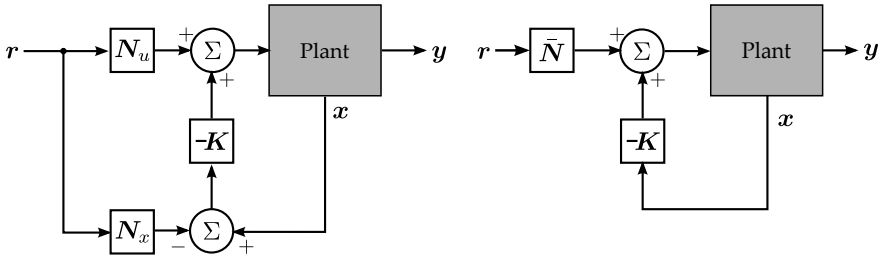


Figure 8.4: Two equivalent configurations for the prefilter. The first is preferred for implementation, though the second simpler form is fine for design and analysis.

by the symbol $\bar{\boldsymbol{N}}$. While in theory these two representations are equivalent, in practice the left system proves more robust. It is fine to use the version on the right when doing design work. Use the form on the left to implement a real system.

We can generalise the above treatment to the MIMO case and also allow for non-unity gain. The derivation is almost identical, but one need to take care of the size of the zero matrices in the equations. Consider the general situation where we want a system to have a particular dc gain described by $\boldsymbol{y}_\mathrm{o} = \boldsymbol{M}\boldsymbol{r}_\mathrm{o}$, where $\boldsymbol{M} \in \mathbb{R}^{p\times p}$ for p the number of outputs (and reference inputs). We will extend the previous treatment to develop a procedure for finding $\bar{\boldsymbol{N}}$ in this case.

$\boldsymbol{y}_o = \boldsymbol{M}\boldsymbol{r}_o$ and we presume there exists $\boldsymbol{N}_x$ and $\boldsymbol{N}_u$ such that $\boldsymbol{x}_o = N_x\boldsymbol{r}_o$ and $\boldsymbol{u}_o = \boldsymbol{N}_u\boldsymbol{r}_o$. Note that $\boldsymbol{N}_u \in \mathbb{R}^{p\times p}$.

$$
\begin{aligned}
\begin{cases} \mathbf{0}_{n\times 1} &= (\boldsymbol{A}\boldsymbol{N}_x + \boldsymbol{B}\boldsymbol{N}_u)\boldsymbol{r}_\mathrm{o} \\ \boldsymbol{y}_0 &= (\boldsymbol{C}\boldsymbol{N}_x + \boldsymbol{D}\boldsymbol{N}_u)\boldsymbol{r}_\mathrm{o} \end{cases} \\
\begin{bmatrix} \mathbf{0}_{n\times 1} \\ \boldsymbol{y}_\mathrm{o} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} \boldsymbol{r}_\mathrm{o} \\
\begin{bmatrix} \mathbf{0}_{n\times 1} \\ \boldsymbol{M}\boldsymbol{r}_\mathrm{o} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} \boldsymbol{r}_\mathrm{o} \\
\begin{bmatrix} \mathbf{0}_{n\times p} \\ \boldsymbol{M} \end{bmatrix} \boldsymbol{r}_\mathrm{o} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} \boldsymbol{r}_\mathrm{o} \\
\begin{bmatrix} \mathbf{0}_{n\times p} \\ \boldsymbol{M} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} \\
\begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0}_{n\times p} \\ \boldsymbol{M} \end{bmatrix}
\end{aligned}
$$

The same equation for finding u_o in the SISO case then applies.

$$
\begin{aligned}
\boldsymbol{u}_\mathrm{o} &= -\boldsymbol{K}\boldsymbol{x} + (\boldsymbol{N}_u + \boldsymbol{K}\boldsymbol{N}_x)\boldsymbol{r} \\
&= -\boldsymbol{K}\boldsymbol{x} + \bar{\boldsymbol{N}}\boldsymbol{r}
\end{aligned}
$$

$\bar{\boldsymbol{N}}$ maps the input vector $\boldsymbol{r}$ to be aligned with $\boldsymbol{y}$ and then scales it as necessary. This method is useful because it allows us to choose the dc effect that each input variable has on the various outputs.

In practice the design of a servo controller can be broken into two parts:

1. Design the feedback gain $\boldsymbol{K}$ using any desired method. The aim of this part of the design is to achieve the desired dynamic performance for the system.

2. Choose $\bar{\boldsymbol{N}}$ so that the dc gain of the system is correct. In the SISO case you could use a calculation of the dc gain, however, it is simpler (and necessary for a MIMO system) to use $\begin{bmatrix} \boldsymbol{N}_x \\ \boldsymbol{N}_u \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I}_m \end{bmatrix} \implies \bar{\boldsymbol{N}} = \boldsymbol{N}_u + \boldsymbol{K}\boldsymbol{N}_x$

Note the slight change to the MIMO form of the $\bar{\boldsymbol{N}}$ equation, as we have here assumed a desired response of $\boldsymbol{y} = \boldsymbol{r}$. The size of the identity matrix used must match the number of inputs (and outputs) to the system.

8.2 Integral Control

The system we have described above will ideally produce zero steady state error when presented with a fixed input. However, that method requires very careful tuning to work perfectly;

- Our model of the plant must be correct ($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,$\boldsymbol{D}$ must be accurate),
- The various gains that we apply must be implemented quite accurately, so we need to worry about things like numerical precision.

As an alternative to introducing $\bar{\boldsymbol{N}}$, we can build a more robust system by introducing integral control. Recall that we used integral control to drive steady state errors to zero in classical control systems. (We could also consider extend our treatment to add additional integrator stages to allow perfect tracking of input ramps, parabaloids or higher order polynomials).

We know that the action of our compensator is to drive all of the state variables to zero. It seems reasonable then to add a new state variable that will represent the integral of the difference between the output and our reference signal (ie, the integral of the output error). The compensator will then drive the integral term to zero by ensuring that the output is the same as the reference. For simplicity we

will assume that we wish to introduce integral control to a single output variable, though the treatment extends naturally to multiple outputs. We will denote the new state variable as x_i, which is defined by $x_\mathrm{i}(t) = \int_0^t \big(r(\tau) - y(\tau)\big)\,\mathrm{d}\tau$. We need to find an equation for x_i in our standard DE form, so we write

$$\begin{aligned}\dot{x}_\mathrm{i} &= r - y \\ &= r - \boldsymbol{C}\boldsymbol{x}\end{aligned}$$

We thus have an augmented system described by the equations

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \dot{x}_\mathrm{i} &= -\boldsymbol{C}\boldsymbol{x} + r \\ y &= \boldsymbol{C}\boldsymbol{x}\end{aligned}$$

We can write this more succinctly as

$$\begin{aligned}\begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{x}_\mathrm{i}\end{bmatrix} &= \begin{bmatrix}\boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix} + \begin{bmatrix}\boldsymbol{B} \\ 0\end{bmatrix} u + \begin{bmatrix}\boldsymbol{0} \\ 1\end{bmatrix} r \\ y &= \begin{bmatrix}\boldsymbol{C} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix}\end{aligned}$$

We now also need a larger feedback gain matrix,

$$u = -\boldsymbol{K}\boldsymbol{x} - k_\mathrm{i}x_\mathrm{i} = -\begin{bmatrix}\boldsymbol{K} & k_\mathrm{i}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix}$$

We can now solve for $\boldsymbol{K}$ and k_i simultaneously by any of the methods that we have previously employed. The matrices used for this are the augmented $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices, $\left[\begin{smallmatrix}\boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & 0\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}\boldsymbol{B} \\ 0\end{smallmatrix}\right]$. Remember that the system now has an extra pole to place. Substituting our equation for u into the system equation we get

$$\begin{aligned}\begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{x}_\mathrm{i}\end{bmatrix} &= \left(\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & 0\end{bmatrix} - \begin{bmatrix}\boldsymbol{B} \\ 0\end{bmatrix}\begin{bmatrix}\boldsymbol{K} & k_\mathrm{i}\end{bmatrix}\right)\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ 1\end{bmatrix} r \\ \begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{x}_\mathrm{i}\end{bmatrix} &= \left(\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & 0\end{bmatrix} - \begin{bmatrix}\boldsymbol{B}\boldsymbol{K} & \boldsymbol{B}k_\mathrm{i} \\ 0 & 0\end{bmatrix}\right)\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ 1\end{bmatrix} r \\ \begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{x}_\mathrm{i}\end{bmatrix} &= \begin{bmatrix}\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} & -\boldsymbol{B}k_\mathrm{i} \\ -\boldsymbol{C} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ 1\end{bmatrix} r \\ y &= \begin{bmatrix}\boldsymbol{C} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ x_\mathrm{i}\end{bmatrix}\end{aligned}$$

8.2.1 Integral control – example

Consider the example that we have previously seen. For comparison we begin by calculating a nominal value of $\bar{\boldsymbol{N}}$ for our example.

```
N = inv([A B; C D])*[zeros(length(A),1); 1];
Nbar = N(end) + K*N(1:end-1);
```

This yields $\bar{\boldsymbol{N}} = 53.05$. If we use this as a prefilter we achieve unity gain as expected. However, if we have the wrong model, then we find that the steady state gain is incorrect. Let's see the effect of using our nominal $\bar{\boldsymbol{N}}$ with $\boldsymbol{A} = \begin{bmatrix}0 & 1 & 0 \\ 0 & 0 & 1 \\ -19 & -14 & -2.2\end{bmatrix}$ instead of $\boldsymbol{A}_\mathrm{nominal} = \begin{bmatrix}0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2\end{bmatrix}$.

The resulting curve can be seen in figure 8.6. Notice that output does not reach the correct final value of $y = 1$.

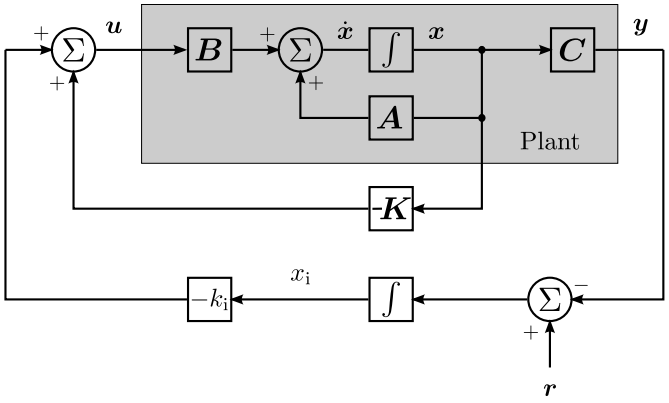


Figure 8.5: Block diagram of a state feedback system including integral control.

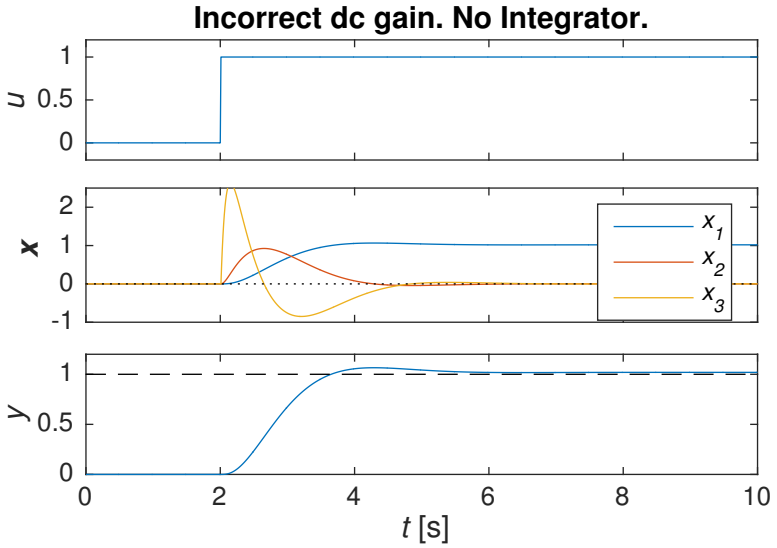


Figure 8.6: Step response of the system when the plant has been perturbed away from the nominal model (that used to design $\bar{\boldsymbol{N}}$).

We would like to add an integrator to remove the error. Begin by adding the extra state.

```
A_i = [A zeros(length(A),1) ; -C 0];
B_i = [B ; 0];
```

We next add an extra pole at $s = -10$ and calculate a new feedback gain matrix, $\begin{bmatrix}\boldsymbol{K} & k_{\mathrm{I}}\end{bmatrix}$.

```
alpha_c= [-1.33+1.49j, -1.33-1.49j, -13.3, -10];
K_i= place(A_i, B_i, alpha_c);
```

Finally, form the new closed loop system.

```
S_i = ss(A_i - B_i * K_i, [zeros(size(B)); 1], [C 0], D);
```

The dc gain is now correct as shown in figure 8.7 .

8.3 Integral Control in Discrete Time

We can also perform integral control for a discrete time system. Consider the case where again we augment a SISO system with a new state $\boldsymbol{x}_{\mathrm{i}}$ which is intended to drive the system output to match its

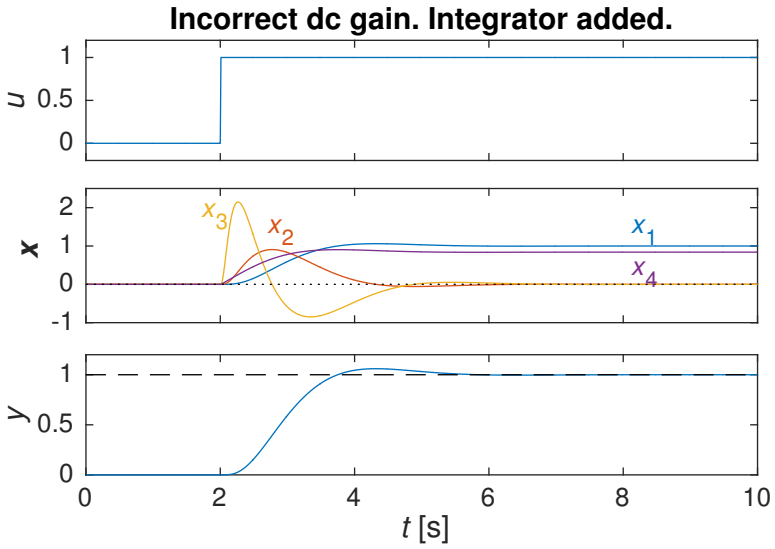


Figure 8.7: Step response of the system when integral control is included. The system was perturbed away from the nominal model.

input. The evolution of the integrator state is governed by

$$\begin{aligned} x_{\mathrm{i}}(t+1) &= x_{\mathrm{i}}(t) + r - y \\ &= x_{\mathrm{i}}(t) + r - C\boldsymbol{x} \end{aligned}$$

So, we can form an augmented system of

$$\begin{bmatrix} \boldsymbol{x}(t+1) \\ x_{\mathrm{i}}(t+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ x_{\mathrm{i}}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} r(t)$$

$$y = \begin{bmatrix} \boldsymbol{C} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_{\mathrm{i}} \end{bmatrix}$$

9 State Observers

When we developed our compensator design techniques we assumed that we had full information to the state vector. That is, we assumed that $\boldsymbol{x}$ was available so that we could use it to generate feedback.

In practice there are number of reasons that we might not have measurements of the entire state vector.

- It may be impossible to measure some state variables.
- A fully instrumented system may be very expensive.
- A measurement might be too noisy.

In this section we consider how to overcome this problem by using an estimate of the state vector in place of the real data when we build our control system. The subsystem that produces the estimate is known as an *observer*, or a *state estimator*.

Observer design requires that we have a good model of the system. In other words, we must know a reasonably accurate set of system matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$. However, we will see that we don't need to have a perfect model as we will use feedback to make the observer somewhat insensitive to modelling errors. For the purposes of this treatment we will assume $\boldsymbol{D} = 0$. Should you encounter a situation where it is not, the extension can be found in numerous texts. $\boldsymbol{D} \neq 0$ doesn't change things much, but it cumbersome to carry around in the mathematical treatment.

We have a number of questions to consider.

- Is it always possible to estimate the entire state of a system?
- How might we best estimate the state?
- How can we design an observer?

9.1 Observability

The basic premise of an observer is that we use observations of the system output $\boldsymbol{y}$ to infer the state $\boldsymbol{x}$. We should first check whether it is possible to build an observer for our system before we attempt it. That is we need to determine whether our system structure allows enough information about $\boldsymbol{x}$ to reach $\boldsymbol{y}$. At first glance we might suspect that we will only be able to determine the state $\boldsymbol{x}(t)$ from $\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t)$ if we could solve $\boldsymbol{x} = \boldsymbol{C}^{-1}\boldsymbol{y}$. This would require as many outputs as there are states, as $\boldsymbol{C}$ would need to be square and full rank. That is, we (correctly) determine that $\boldsymbol{y}(t_1)$ is observable from $\boldsymbol{x}(t_1) \in \mathbb{R}^n$ if $\boldsymbol{C} \in \{\mathbb{R}^{n\times n} : \mathcal{R}\boldsymbol{C} = \mathbb{R}^n\}$ and $\boldsymbol{C}$ is full rank. This is not very profound. It simply tells us that we can determine all of the state variables if we measure them.

However, we are not required to infer $\boldsymbol{x}$ from a single measurement of $\boldsymbol{y}$. We can watch the trajectory of $\boldsymbol{y}$ and then use *multiple* $\boldsymbol{y}$ values to untangle the behaviour of $\boldsymbol{x}$. Consider the trajectory of $\boldsymbol{y}$ produced by a non-zero initial state $\boldsymbol{x}(0)$ in an autonomous, discrete time system.

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

Each successive sample *could* add more information about $\boldsymbol{x}(0)$ until we reach the $(n-1)$-th state. In other words, the effect of $\boldsymbol{x}(0)$ is "smeared" over $n-1$ time steps as a consequence of the convolution property of LTI systems.

That is, we can write the part of the output trajectory that contains information about $\boldsymbol{x}(0) \in \mathbb{R}^n$ as

$$\begin{bmatrix} \boldsymbol{y}(0) \\ \boldsymbol{y}(1) \\ \boldsymbol{y}(2) \\ \vdots \\ \boldsymbol{y}(n-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{n-1} \end{bmatrix} \boldsymbol{x}(0)$$
$$= \boldsymbol{\mathcal{M}}_{\mathrm{o}} \boldsymbol{x}(0)$$

where $\boldsymbol{\mathcal{M}}_{\mathrm{o}}$ is the *observability matrix*.

The observability matrix is dual to the controllability matrix we encountered earlier. The observability matrix $\boldsymbol{\mathcal{M}}_{\mathrm{o}}$ must be full rank for our system to be *observable*. In the case of a single output system, $\boldsymbol{\mathcal{M}}_{\mathrm{o}}$ will be square, which is the same as requiring $\det \boldsymbol{\mathcal{M}}_{\mathrm{o}} \neq 0$. You should always test for observability before designing an observer. Matlab's `obsv` command may prove useful. Sometimes you can see that a system is unobservable. For example, if a system is in modal form, then a zero column in the $\boldsymbol{C}$ matrix corresponds with a state variable that is not observable.

9.1.1 Unobservable systems

A system is unobservable with one or more of its state variables are not coupled to the output variables. As they do not affect the output we cannot infer their behaviour by looking solely at the output.

This structure is illustrated in figure 9.1, in which we have divided the state vector into two parts $\boldsymbol{x} := \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 \end{bmatrix}^{\mathsf{T}}$. The various system matrices have also been partitioned to make clear the portions that act on $\boldsymbol{x}_1$ and those that act on $\boldsymbol{x}_2$. So, $\boldsymbol{A} := \begin{bmatrix} \boldsymbol{A}_{11} & 0 \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}$, $\boldsymbol{B} := \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix}$ and $\boldsymbol{C} := \begin{bmatrix} \boldsymbol{C}_1 & 0 \end{bmatrix}$. Notice that $\boldsymbol{x}_2$ cannot affect $\boldsymbol{y}$, so the plant is unobservable.

As was the case for controllability, a mathematical test for observability does not mean that all state variables can be inferred in reality. If the coupling of a state variable to the system output is very weak it can prove unrealistic to try to extract its value. As was the case for controllability, we could use SVD to see how close to unobservable a system might be and determine which state variables we might be trouble inferring practically.

If a system is unobservable you can sometimes redesign the system itself. For example, it is sometimes possible to add extra sensors to gather more information about the state.

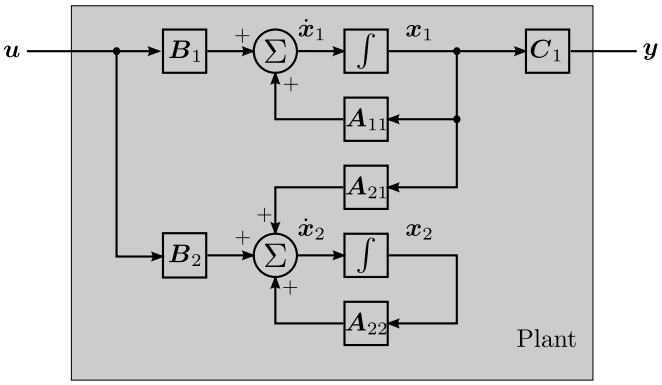


Figure 9.1: Graphical representation of an unobservable system. In this system the subset of state variables represented by $\boldsymbol{x}_2$ are not connected to the output, either directly or indirectly (though $\boldsymbol{x}_1$). As a result looking at $\boldsymbol{y}$ does not allow inference of the behaviour of $\boldsymbol{x}_2$.

An unobservable system may still be fine to work with if the unobservable modes are sufficiently well behaved. If all of the unobservable modes of a system decay to zero asymptotically (are stable), then a system is said to be *detectable*. If those modes are sufficiently fast then we could just assume the corresponding state variables to be zero and proceed with our design.

9.2 Observer Design

We know that the evolution of our plant is described by the state equation $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$. We now seek to construct a model of the system $\dot{\hat{\boldsymbol{x}}} = \widehat{\boldsymbol{A}}\hat{\boldsymbol{x}} + \widehat{\boldsymbol{B}}\boldsymbol{u}$, where we have used $\hat{\boldsymbol{x}}$ to denote our *estimate* of the state vector. When necessary we will use $\widehat{\boldsymbol{A}}$, $\widehat{\boldsymbol{B}}$, $\widehat{\boldsymbol{C}}$ to clarify that we are referring to the observer's model of our system rather than the true system. However, for simplicity we will assume a perfect model ($\widehat{\boldsymbol{A}} = \boldsymbol{A}$ and so forth) for much of the time. We also define $\tilde{\boldsymbol{x}}$ as the error in our estimate, that is the error is the difference between the real state vector and our estimate of it; $\tilde{\boldsymbol{x}} := \boldsymbol{x} - \hat{\boldsymbol{x}}$. Our task in designing an observer is to make $\tilde{\boldsymbol{x}}$ as small as possible. We also extend this notation to $\hat{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}}$ in the natural way.

9.2.1 Open loop observer

One possible way to build an observer is to run a modelled version of the system in parallel with the real system. If the system is stable (and we have a sensible model) then this observer will converge to the real state vector. There are however, a couple of problems.

- The model and the observer are likely to have different initial conditions. There can be large discrepancies between the estimate and reality while the transient decays.
- We have no way of tailoring the response of the observer to suit our application.

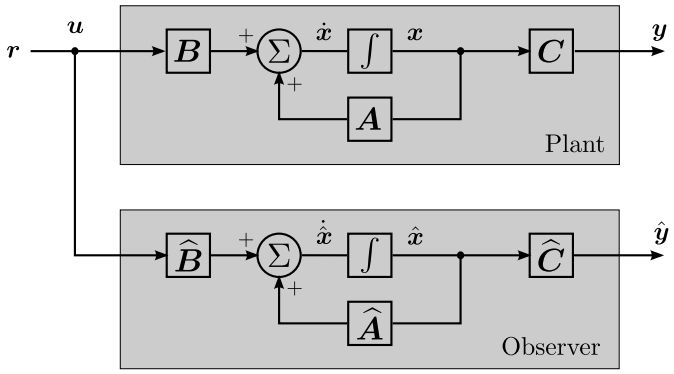


Figure 9.2: Plant with an open loop observer.

9.2.2 Closed loop observer

The open loop observer provides an estimate of the output $\hat{\boldsymbol{y}}$. We can calculate the difference between this estimate and the real output and feed this information back to the observer. That is, we will measure $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ and feed it back to the input of the observer via a gain $\boldsymbol{L}$.

In the open loop case our observer dynamics are governed by $\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u}$. In the closed loop case we must add the effects of feedback, so we replace $\boldsymbol{B}\boldsymbol{u}$ with $\boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\tilde{\boldsymbol{y}}$. Thus we have

$$\begin{aligned}
\dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\tilde{\boldsymbol{y}} \\
&= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\boldsymbol{y} - \boldsymbol{L}\hat{\boldsymbol{y}} \\
&= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\boldsymbol{C}\boldsymbol{x} - \boldsymbol{L}\boldsymbol{C}\hat{\boldsymbol{x}} \\
&= \boldsymbol{L}\boldsymbol{C}\boldsymbol{x} + (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u}
\end{aligned}$$

We are often interested in the behaviour of $\tilde{\boldsymbol{x}}$, so let's subtract the equation for $\hat{\boldsymbol{x}}$ from that for $\boldsymbol{x}$.

$$\begin{aligned}
\dot{\tilde{\boldsymbol{x}}} = \dot{\boldsymbol{x}} - \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} - (\boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\boldsymbol{C}\boldsymbol{x} - \boldsymbol{L}\boldsymbol{C}\hat{\boldsymbol{x}}) \\
&= \boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{L}\boldsymbol{C}\tilde{\boldsymbol{x}} \\
\dot{\tilde{\boldsymbol{x}}} &= (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})\,\tilde{\boldsymbol{x}}
\end{aligned}$$

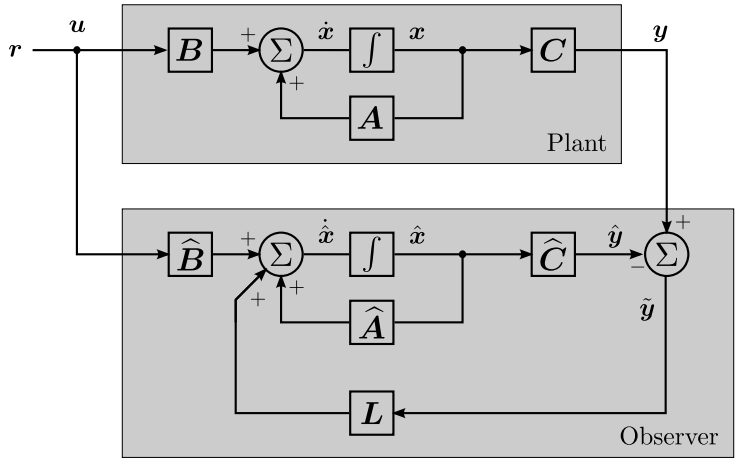


Figure 9.3: Plant with a closed loop observer.

As the behaviour of our state error is described by $\dot{\tilde{\boldsymbol{x}}} = (\boldsymbol{A} - \boldsymbol{LC})\tilde{\boldsymbol{x}}$, we can find the modes of the estimate error signal by solving the characteristic equation,

$$\det\left(s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{LC})\right) = \det\left(s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{LC}\right).$$

This equation is very similar to that which we encountered while designing compensators. This time we can control the location of the observer poles by choosing $\boldsymbol{L}$ appropriately. This allows us to set how quickly the estimator will converge to the true state. We find the appropriate value for $\boldsymbol{L}$ using the same techniques we used to find $\boldsymbol{K}$ earlier. The only difference is that we will find that *observer* canonical form is more convenient for designing the observer in SISO systems (basically because $\boldsymbol{B}$ is a column vector and $\boldsymbol{C}$ is a row vector).

We can use any of the methods described in the section on compensator design to determine $\boldsymbol{L}$; we can solve directly, convert to observer canonical form and solve by inspection, or alternatively we can use Ackermann's formula.

In this case we express Ackermann's equation as

$$\boldsymbol{L} = \alpha_{\mathrm{e}}(\boldsymbol{A})\boldsymbol{\mathcal{M}}_{\mathrm{o}}^{-1}\begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}$$

where α_{e} is the desired closed loop polynomial. The roots of α_{e} are the poles that govern the modes of the estimator error.

The compensator and observer design problems are obviously very similar, but not identical. In one we seek to position the poles of $\boldsymbol{A}-\boldsymbol{BK}$, in the other $\boldsymbol{A}-\boldsymbol{LC}$. In particular notice the position of $\boldsymbol{B}/\boldsymbol{C}$ is different in the two.

We can make the equations look more similar if we recognise that the poles of poles of $\boldsymbol{A} - \boldsymbol{LC}$ will be the same as those of $(\boldsymbol{A} - \boldsymbol{LC})^{\mathsf{T}}$. Now $(\boldsymbol{A} - \boldsymbol{LC})^{\mathsf{T}} = \boldsymbol{A}^{\mathsf{T}} - \boldsymbol{C}^{\mathsf{T}}\boldsymbol{L}^{\mathsf{T}}$.

We now have the observer equation in exactly the same form as the compensator equation (so long as we transpose the matrices before and after). In matlab for example, we could find $\boldsymbol{L}$ with

```
L = place(A.', C.', alpha_o).'
```

We considered a number of approaches to placement of the compensator poles. We have a similar set of options now, though there are a couple of extra considerations.

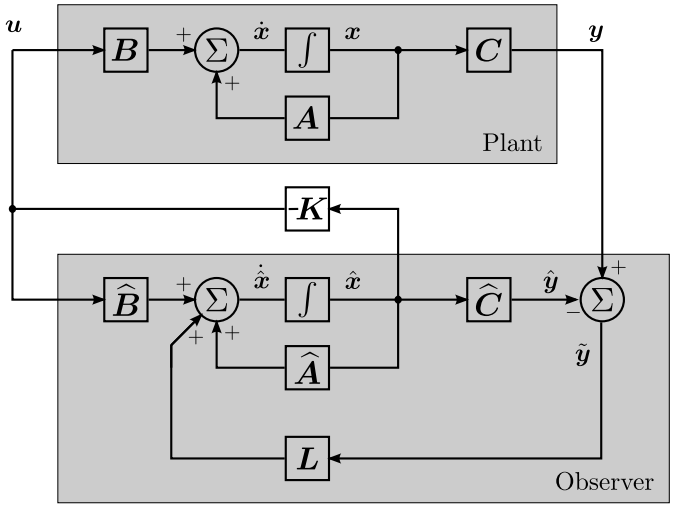


Figure 9.4: A regulator where the feedback is derived from the state estimate produced by a closed loop observer.

- We don't need to worry about control effort as we don't need to control a real plant, just the model in the observer. Large signals in an observer are just large numbers in a computer.
- The observer poles must be faster than the compensator poles, so that the estimate becomes meaningful before the compensator acts on it.

The requirement that the observer be faster than the compensator translates to a requirement to place the poles further to the left in the s-plane (or closer to the origin in the z-plane). However, you should be wary of placing them too far to the left (or centre), as this increases the bandwidth of the system and makes it more susceptible to noise. A factor of 3-5 faster is generally sufficient.

9.3 Using observers for regulator problems

We now have an observer that generates an estimate of the state vector $\hat{\boldsymbol{x}}$. We can now modify our compensator design to use the estimate of the state vector rather than the real vector.

Recall that when we developed the compensator, we arrived at a state transition equation that incorporated the effect of the feedback,

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} = \boldsymbol{Ax} - \boldsymbol{BKx}$$

We now modify this to use $\hat{\boldsymbol{x}}$ rather than $\boldsymbol{x}$ for the feedback source.

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{Ax} - \boldsymbol{BK}\hat{\boldsymbol{x}} \\ &= \boldsymbol{Ax} - \boldsymbol{BK}(\boldsymbol{x} - \tilde{\boldsymbol{x}}) \\ \dot{\boldsymbol{x}} &= (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} + \boldsymbol{BK}\tilde{\boldsymbol{x}}\end{aligned}$$

Thus the dynamics of the system depend on $\boldsymbol{A} - \boldsymbol{BK}$ as before, but now there is an extra term arising from the estimate error $\tilde{\boldsymbol{x}}$. However, we know how the error behaves, so we will be able to incorporate this into our model.

We can build a new state space representation of the entire system. It will contain twice as many states as the original system as we now must model the behaviour of both $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. Recall that $\dot{\tilde{\boldsymbol{x}}} = (\boldsymbol{A} - \boldsymbol{LC})\tilde{\boldsymbol{x}}$. Combining this with the previous equation yields our new state equation:

$$\begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{\tilde{\boldsymbol{x}}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A} - \boldsymbol{BK} & \boldsymbol{BK} \\ 0 & \boldsymbol{A} - \boldsymbol{LC}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ \tilde{\boldsymbol{x}}\end{bmatrix}$$

In this equation remember that each of the terms in the vectors/matrix is another vector or matrix. We are simply concatenating the two different entries into larger entities. We can now determine the dynamics of the new super-system by solving the characteristic equation:

$$\det \begin{bmatrix} s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BK} & -\boldsymbol{BK} \\ 0 & s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{LC} \end{bmatrix} = 0$$

This matrix is in block triangular form which makes it easy to simplify the determinant calculation.

$$\begin{aligned} \det \begin{bmatrix} s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BK} & -\boldsymbol{BK} \\ 0 & s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{LC} \end{bmatrix} &= 0 \\ \implies \det[s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BK}].\det[s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{LC}] &= 0 \\ \alpha_{\mathrm{c}}(s)\alpha_{\mathrm{e}}(s) &= 0 \end{aligned}$$

Thus the total system poles are a combination of the poles of the compensated system (α_{c}) and the observer (α_{e}). If we place our observer poles sufficiently far to the left (or centre) the compensator poles should be dominant and the switch to using $\hat{\boldsymbol{x}}$ should have a negligible effect on the final result. This is known as the separation principle; we can design our compensator and observer separately and combine them without concern.

9.4 Implementation of an Observer

The formulation above describes the evolution of $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, which is useful for control system analysis and design. However, for implementation we need to find the evolution of $\hat{\boldsymbol{x}}$ so that we can use the estimate.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{Ax} - \boldsymbol{BK}\hat{\boldsymbol{x}} \\ \text{and, } \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{LCx} + (\boldsymbol{A} - \boldsymbol{BK} - \boldsymbol{LC})\hat{\boldsymbol{x}} \end{aligned}$$

If we are careful we find see tjat the matrices in the bracketed term refer to the estimates of the system matrices, so if we want to generalise to the case where our model differs from the real system then we have

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{LCx} + (\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{B}}\boldsymbol{K} - \boldsymbol{L}\widehat{\boldsymbol{C}})\hat{\boldsymbol{x}}$$

So, for simulation work we form an augmented state matrix that is goverened by

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\hat{\boldsymbol{x}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{BK} \\ \boldsymbol{LC} & \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{B}}\boldsymbol{K} - \boldsymbol{L}\widehat{\boldsymbol{C}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix}$$

Of course, for implementation we do not know $\boldsymbol{x}$, nor do we need to calculate it. We would simply use

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{Ly} + (\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{B}}\boldsymbol{K} - \boldsymbol{L}\widehat{\boldsymbol{C}})\hat{\boldsymbol{x}} \\ \boldsymbol{u} &= -\boldsymbol{K}\hat{\boldsymbol{x}} \end{aligned}$$

Figure 9.5 shows a schematic for such a system.

9.5 Observers in servo systems

We need to consider how the addition of the reference signal will change the observer. We can follow two approaches here:

1. Design the system so that changes in the reference do not excite the observer error. That is, we drive the observer in the same way as the plant.
2. Allow reference changes to excite the observer in such a way that we can tailor the transient response.

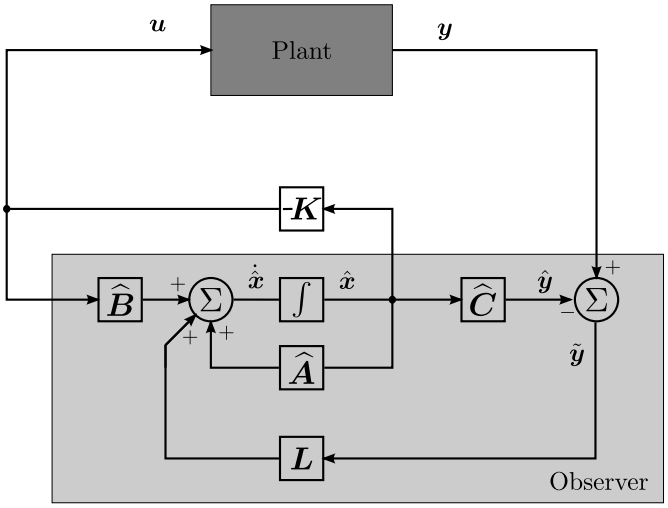


Figure 9.5: Implementation of an observer.

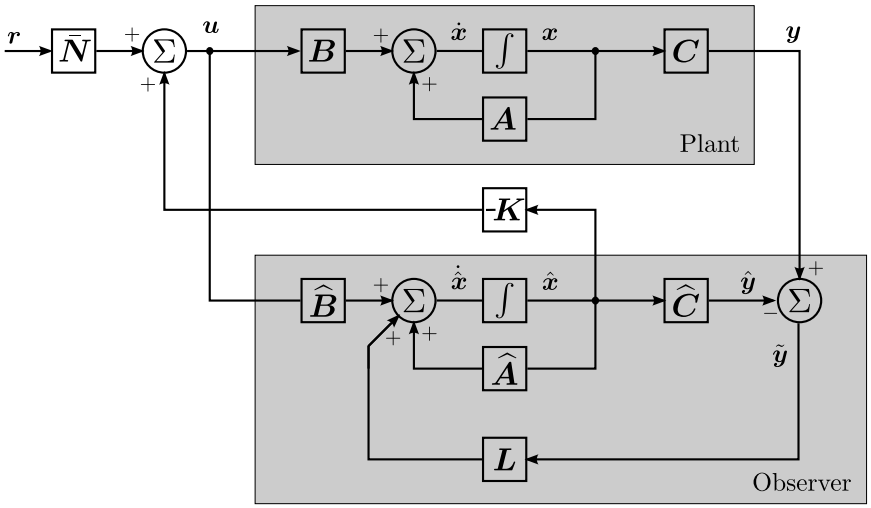


Figure 9.6: Block diagram for a system employing estimated state feedback and using a prefilter $\bar{\boldsymbol{N}}$ to adjust the dc gain for the reference signal $\boldsymbol{r}$.

The second approach is very powerful, as it actually allows us to place the zeros of the system in an arbitrary location. This give us almost complete control over the system's transient response. However, we will use the simpler approach as it is usually sufficient.

This is a trivial change to the system diagram. We have just added the prefilter matrix $\bar{\boldsymbol{N}}$ to adjust the gain of the system.

9.6 Observer Design Example

Consider the system described in the regulator lecture. We now wish to design an appropriate state observer and use its state estimate to stabilise the system. Recall that the regulator poles were at $s \in \{-1.33 \pm \mathrm{j}1.49, -13.3\}$. We need to place three observer poles at least three times further into the s-plane than the regulator poles. Let's place the poles just a factor of three further to the left. That is, we will put the observer poles at $s \in \{-4 \pm \mathrm{j}4.5, -39\}$

First let's check the system's observability. `obsv(A,C)` yields the identity matrix (which is full rank). Therefore the system is observable. Let's find $\boldsymbol{L}$ to place the observer poles:

```
L= place(A', C', [-39,-4+4.5j,-4-4.5j])'
```

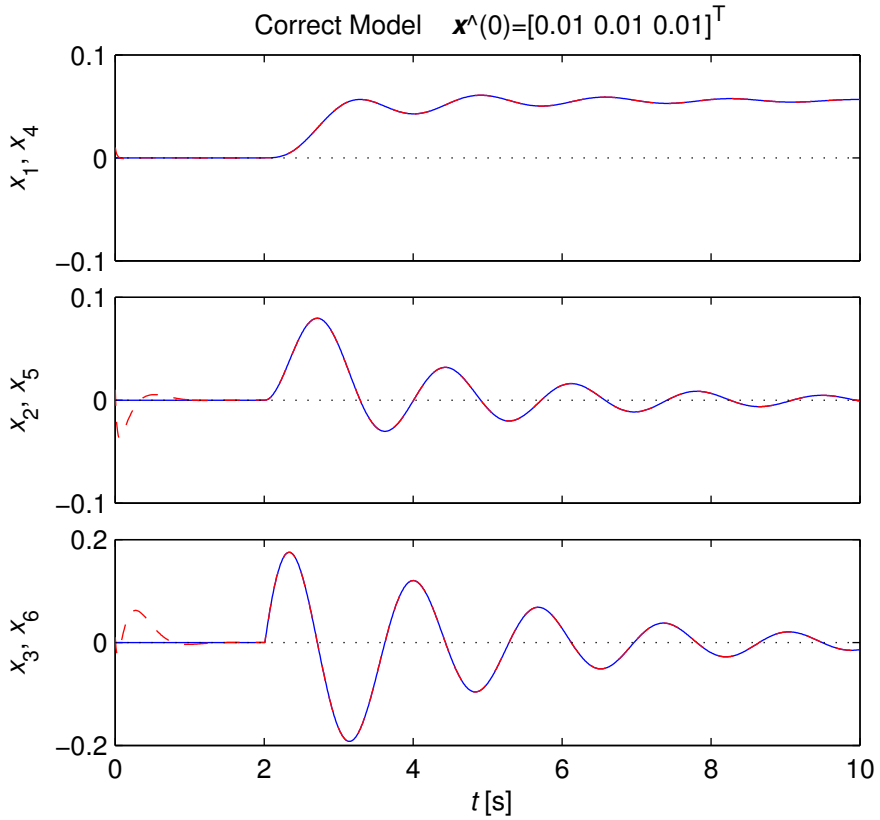


Figure 9.7: Comparison of a plant and an observer than is given an incorrect initial guess at the state. The plant responses are the solid blue curves, while the red dashed responses are those of the estimate. In this case, $\boldsymbol{x} = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^{\mathsf{T}}$, but $\hat{\boldsymbol{x}} = \begin{bmatrix}0.01 & 0.01 & 0.01\end{bmatrix}^{\mathsf{T}}$

.

which yields $\boldsymbol{L} = \begin{bmatrix}45.9\\248\\231\end{bmatrix}$.

Let's run a simulation of the system open loop and see how the observer performs in predicting the system state. We will give the observer an incorrect initial guess at the state to see the observer transient. The result can be seen in figure 9.7.

9.6.1 Incorrect Model

Consider the case where we have in incorrect model of the system. We will again use the system that we have previously discussed, which had $\boldsymbol{A} = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\-18 & -15 & -2\end{bmatrix}$. However, we will use $\widehat{\boldsymbol{A}} = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\-19 & -14 & -2.2\end{bmatrix}$ for the model inside the observer. This has the effects of shifting the open loop eigenvalues from $\{-1.29, -0.36 \pm$ j3.73$\}$ to $\{-1.47, -0.37 \pm$ j3.58$\}$, which makes the exponential mode faster and decreases the frequency of the oscilliatory mode slightly while increasing its damping. In the following figures we will see the step response of an open loop observer (9.8) , a closed loop observer with relatively slow response (9.9) and a closed loop observer with fast observer poles (9.10) . In all cases there is no controller used ($\boldsymbol{K} = 0$), so we are observing the natural open loop behaviour of the plant. On the following graphs, $x_4 = \hat{x}_1$, $x_5 = \hat{x}_2$ and $x_6 = \hat{x}_3$.

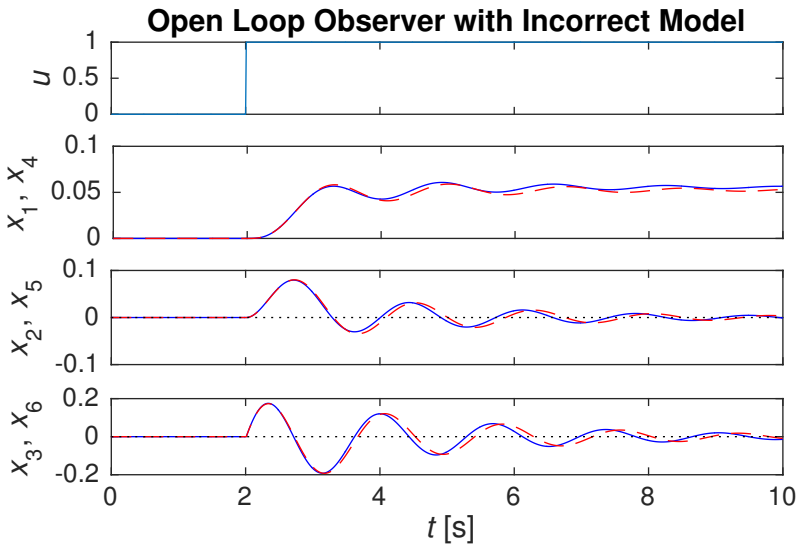


Figure 9.8: Operation of an observer that has an incorrect model of the plant.

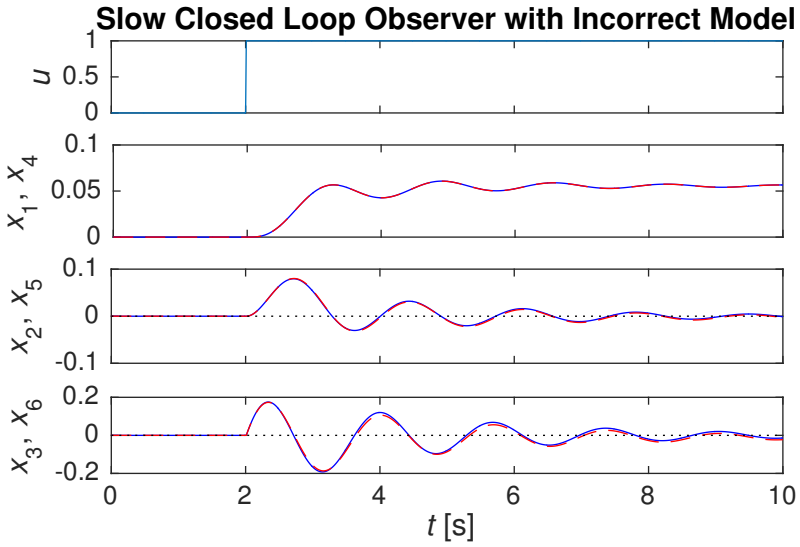


Figure 9.9: Operation of an observer having an incorrect model and slow observer poles.

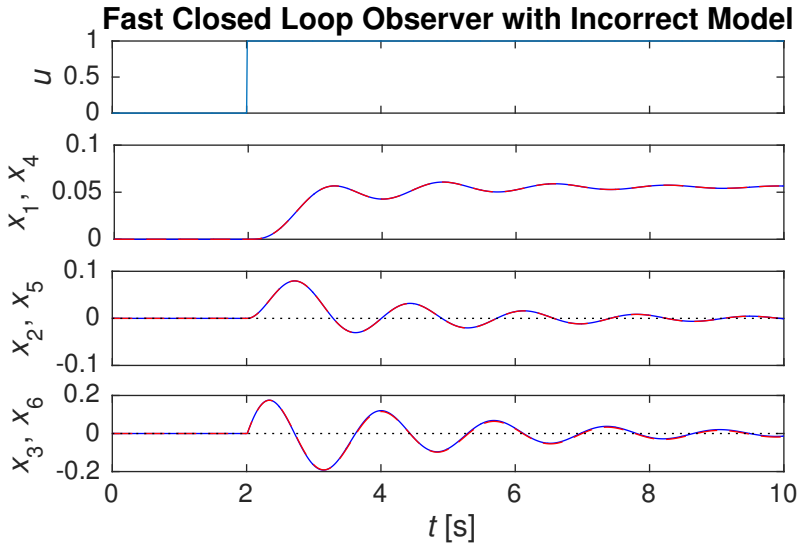


Figure 9.10: Operation of an observer having an incorrect model and slow observer poles.

Disturbance Rejection with an Incorrect Model

We can also examine the response of the three observers when the plant is excited with an impulse, but the observer is not. This is a model of a simple disturbance to the plant that we do not know about, so therefore cannot feed into the observer directly. The respones of the open loop observer, slow closed loop observer and fsat closed loop observer are shown in figures 9.11, 9.12 and 9.13 respectively. We see that the open loop observer does not respond to the disturbance at all. The closed loop observers do respond, with the observer having fast poles acting to reduce the state error more effectively. However, we should not get too excited about this approach, as fast observers are more susceptible to high frequency noise in the measurements.

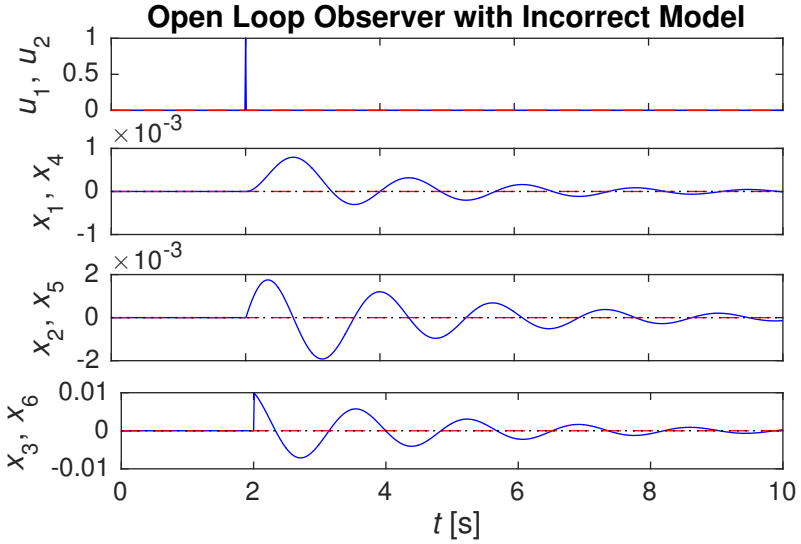


Figure 9.11: Operation of an observer that has an incorrect model of the plant.

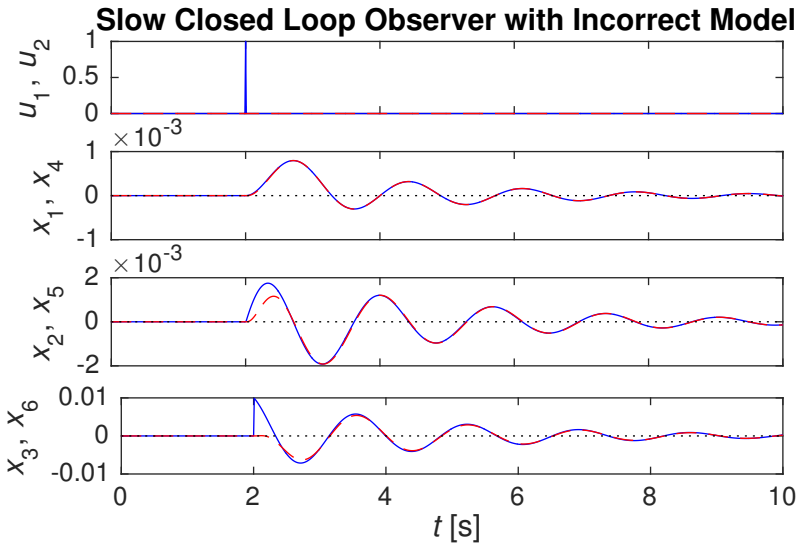


Figure 9.12: Operation of an observer having an incorrect model and slow observer poles.

One problem of making the observer poles too fast is that the effects of measurement noise is more significant. Sensor noise is modelled as a signal $\boldsymbol{v}$, such that $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{v}$. This noise is impressed upon $\tilde{\boldsymbol{y}}$ and then fed back to the estimator state via $\boldsymbol{L}$, which degrades the quality of the estimate. In this

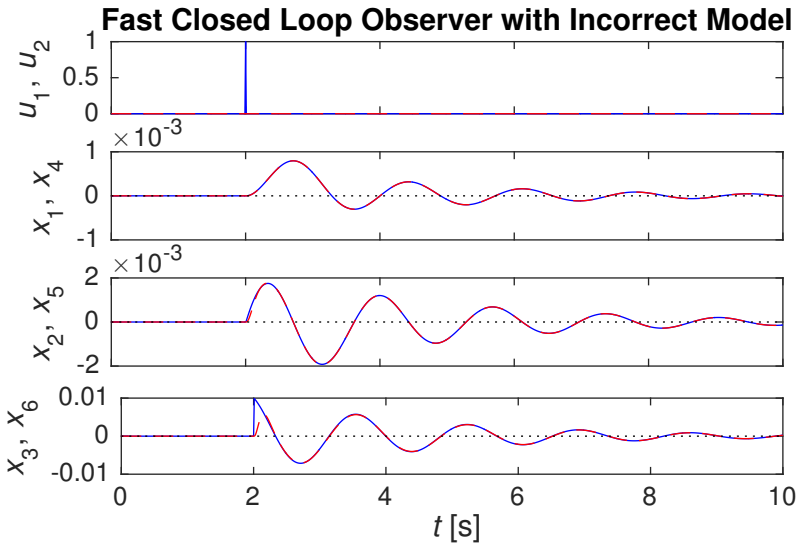


Figure 9.13: Operation of an observer having an incorrect model and slow observer poles.

case we have

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{B}\boldsymbol{K}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{L}\boldsymbol{C}\hat{\boldsymbol{x}} + \boldsymbol{L}\boldsymbol{C}\boldsymbol{x} + \boldsymbol{L}v$$

$$\implies \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\hat{\boldsymbol{x}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}\boldsymbol{K} \\ \boldsymbol{L}\boldsymbol{C} & \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} - \boldsymbol{L}\boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} & 0 \\ \boldsymbol{B} & \boldsymbol{L} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

In the examples to follow we have contaminated the measurement with white noise having variance 100×10^{-6}. That is,

$$y = \boldsymbol{C}\boldsymbol{x} + v, \text{ where } v \sim \mathcal{N}\left(0, 100 \times 10^{-6}\right)$$

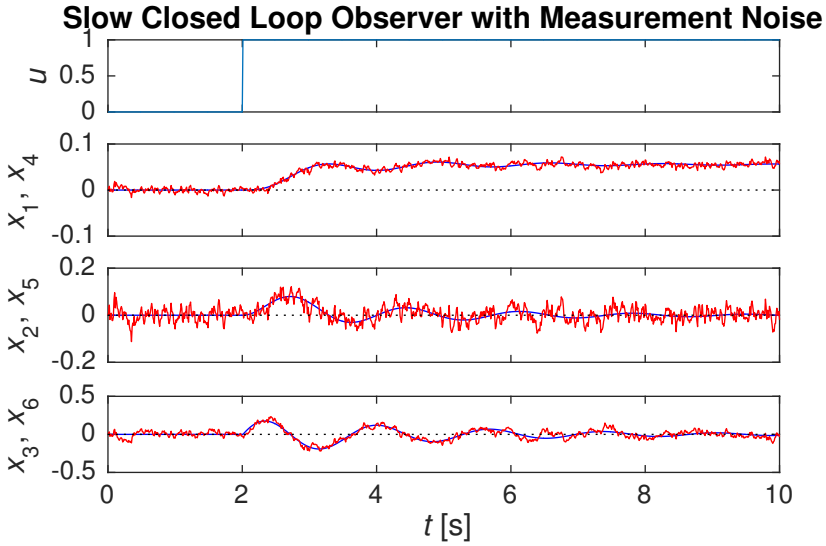


Figure 9.14: Operation of an observer having finite measurement noise and slow observer poles.

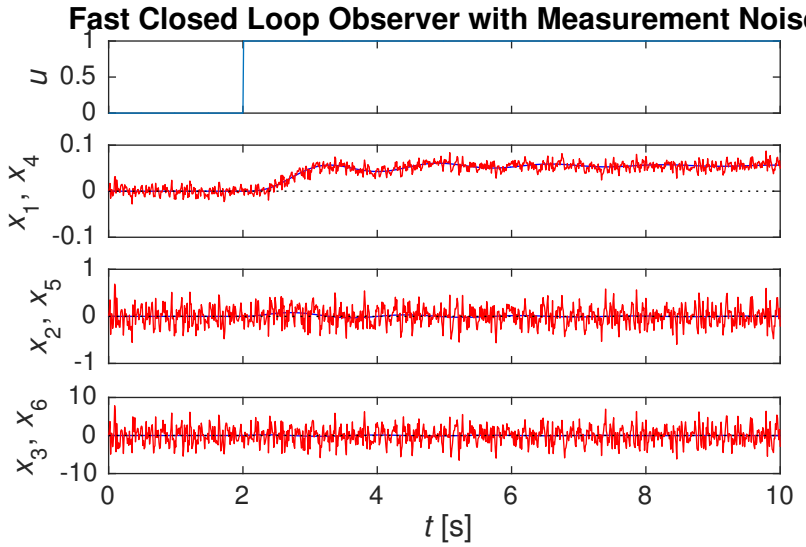


Figure 9.15: Operation of an observer having finite measurement noise and slow observer poles.