Fault-Tolerant and Secure Control Systems

Shreyas Sundaram

Department of Electrical and Computer Engineering
University of Waterloo

Acknowledgments

These notes are for the graduate course on *Fault-Tolerant and Secure Control Systems* offered at the University of Waterloo. They are works in progress, and will be continually updated and corrected.

The LaTeX template for *The Not So Short Introduction to LaTeX 2ε* by T. Oetiker et al. was used to typeset portions of these notes.

Shreyas Sundaram
University of Waterloo

Contents

Chapter 1

Introduction

The world around us is replete with complex engineered systems: automobiles [11], air traffic control systems [54], building energy management systems [88], industrial process systems [79], and the electrical power grid [58] are just a few examples. Many of these systems are life- and mission-critical, where disruptions (either by intent or by accident) could have dire consequences. There have been various incidents in recent years that exemplify this fact [27, 50, 81].

- In 1999, a gas pipeline belonging to Olympic Pipeline Co exploded, killing three people. One of the many factors that led to this explosion was that the supervisory control system became unresponsive and failed to take action to relieve the buildup of pressure in the pipeline. The reason for this failure was later determined to be caused by an update of the control system software just before the incident took place, without adequate testing in an offline environment [7].
- In 2000, a disgruntled employee hacked into the control system responsible for managing the sewage system in Queensland, Australia, causing it to release $264,000$ tons of sewage into parks and rivers [80].
- In 2003, an overloaded transmission line in Ohio tripped, starting a chain of cascading failures. Around the same time, the monitoring systems at power control rooms failed, leading operators to miss the alarms. Within hours, a large portion of the Northeastern United States and Canada were affected by a major power failure, affecting more than 50 million people.

In addition to the above high-profile incidents, faults can have a detrimental impact on low-profile day-to-day activities. For example, modern *high-performance* buildings are typically designed to operate efficiently and minimize energy usage via the use of many sensors and advanced control strategies. However, the intended gains will not be realized if the sensors or control systems fail, leading to energy waste and uncomfortable conditions [43, 45].

To avoid physical and economic damage caused by such incidents, it is imperative to design systems to be resilient to unexpected events. Much work has been done over the past few decades on developing mechanisms for obtaining fault-tolerance in different types of systems. These range from conceptually simple modular redundancy schemes to more advanced model-based fault-diagnosis techniques. The need for a rigorous theory of *security* in control systems has only recently been recognized as a fertile and important area of research. The report [81] highlights several key differences in security requirements for control systems and traditional information technology systems. One particularly important differentiating factor is that control systems involve the regulation of physical processes. This fact imposes hard real-time constraints on the system, since a delay in processing could lead to instabilities or cascading failures in the control loop, potentially causing severe physical and economic damage. Information technology systems, on the other hand, focus on the protection and delivery of data as the key priority; while delays or down-time are highly undesirable, they can frequently be tolerated without causing major or permanent damage. The paper [9] echoes the call for the development of a rigorous theory of security in control systems, detailing several open challenges for research. Furthermore, the North American Electric Reliability Corporation (NERC) and the US Department of Energy have mandated that electric utilities comply with strict cyber-security standards, and are turning down *smart power grid* projects that do not incorporate such measures in their design [52, 33]. However, ensuring security and reliability in these cases poses a unique challenge; the complex interplay of control, communications and computation in such systems necessitates a holistic approach to design.

At a philosophical level, one can view attacks as a type of 'worst-case' fault, where the location and magnitude of the fault are chosen so as to cause the most damage to the system. In this course, we will be adopting this perspective, and developing a methodology to deal with faults and attacks using rigorous mathematical tools. We will be focusing on a particular class of systems that involve *dynamics*, which we will describe next.

1.1 Dynamical Systems and Control Theory

For the purposes of this course, a *system* is an abstract object that accepts *inputs* and produces *outputs* in response. Systems are often composed of smaller components that are interconnected together, leading to behavior that is more than just the sum of its parts. In the control literature, systems are also commonly referred to as *plants* or *processes*.

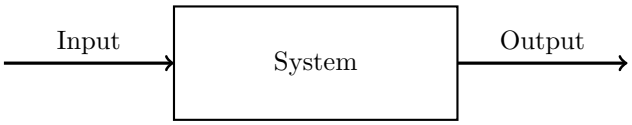

Figure 1.1: An abstract representation of a system.

The term *dynamical system* loosely refers to any system that has a state and some dynamics (i.e., a rule specifying how the state evolves in time). This description applies to a very large class of systems, from automobiles and aviation to industrial manufacturing plants and the electrical power grid. The presence of dynamics implies that the behavior of the system cannot be entirely arbitrary; the temporal behavior of the system's state and outputs can be predicted to some extent by an appropriate *model* of the system.

In physical systems, the inputs are applied via *actuators*, and the outputs are measurements of the system state provided by *sensors*.

Example 1.1. Consider a simple model of a car in motion, and let the speed of the car at any time t be given by $v(t)$. One of the inputs to the system is the acceleration $a(t)$, applied by the throttle (the actuator). Suppose that we are interested in the speed of the car at every second (i.e., at $t = 0, 1, 2, \ldots$), and that the acceleration of the car only changes every second, and is held constant in between changes. We can then write

$$v(t+1) = v(t) + a(t), \tag{1.1}$$

since the speed of the car one second from now is just the current speed plus the acceleration during that second. The quantity $v(t)$ is the state of the system, and equation (1.1) specifies the dynamics. There is a speedometer on the car, which is a sensor that measures the speed once per second. The value provided by the sensor is denoted by $s(t) = v(t)$, and this is taken to be the output of the system.

What is Control Theory? The field of *control systems* deals with applying or choosing the inputs to a given system to make it behave in a certain way (i.e., make the state or output of the system follow a certain trajectory). A key way to achieve this is via the use of *feedback*, where the input depends on the output in some way. This is also called *closed loop control*. Typically, one uses a computational element in the feedback loop to process the sensor measurements and convert it to an appropriate actuator signal; this is called a *controller*. Feedback control is everywhere, from engineered systems (such as automobiles and aircraft) to economic systems, ecological systems (predator/prey populations, global climate) and biological systems (e.g., physiology in animals and plants).

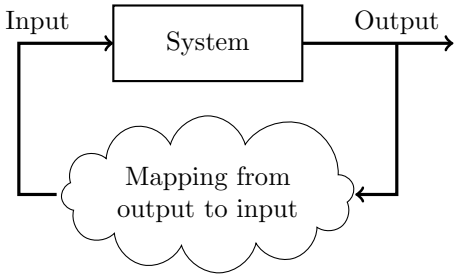

Figure 1.2: Feedback Control.

Example 1.2 (Cruise Control)**.** Consider again the simple model of a car from Example 1.1. A *cruise control system* for the car would work as follows.

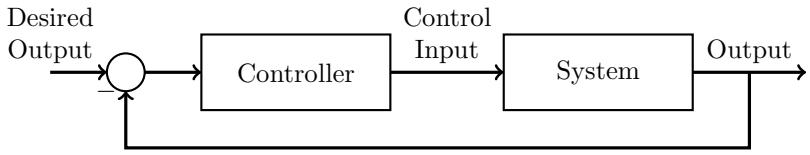

Figure 1.3: Block Diagram of a feedback control system.

- The speedometer in the car measures the current speed and produces $s(t) = v(t)$.
- The controller in the car uses these measurements to produce control signals: if the current measurement $s(t)$ is less than the desired cruising speed, the controller sends a signal to the throttle to accelerate, and if $s(t)$ is greater than the desired speed, the throttle is asked to allow the car to slow down.
- The throttle performs the action specified by the controller.

Example 1.3 (Inverted Pendulum)**.** Suppose we try to balance a stick vertically in the palm of our hand. The sensor, controller and actuator in this example are our eyes, our brain, and our hand, respectively. This is an example of a feedback control system. Now what happens if we try to balance the stick with our eyes closed? The stick inevitably falls. This illustrates another type of control, known as *feedforward* control, where the input to the system does not depend on the output. As this example illustrates, feedforward control is not *robust* to disturbances – if the stick is not perfectly balanced to start, or if our hand moves very slightly, the stick will fall.

Note that systems typically have many inputs, some of which are not under the control of the designer. One common example of this is the presence of *disturbances* in the system. For example, the motion of the car might be affected by wind gusts, or might encounter slippery road conditions. Control theory uses mathematical tools and analysis to determine the proper inputs to apply (based on the outputs of the system) in order to obtain correct, safe and efficient behavior, despite the presence of disturbances.

1.2 Overview of Fault-Tolerance and Security

Components of systems can fail due to a variety of factors, ranging from wear-and-tear, adverse conditions, accidents, or targeted attacks. With regard to the discussion in the previous section, faults or attacks can manifest themselves in the plant (changing the system dynamics), sensors (providing incorrect state measurements), or actuators (applying incorrect inputs). This can cause the functioning of the system to be severely impacted, unless care has been taken to design fault-tolerance into the system. The goal of fault-tolerant control is to allow the system to gracefully degrade or continue functioning under failures, and prevent faults from propagating to other parts of the system. Fault-tolerance can broadly be broken down into two objectives [6]:

1. **Fault detection and identification (FDI)**: determine whether a fault has occurred, and isolate the component that has failed.
2. **Fault accommodation**: take steps to correct for the fault, or reconfigure your system to avoid the faulty component.

Figure 1.4 shows a block diagram from [6] that illustrates the structure of a fault-tolerant control system.

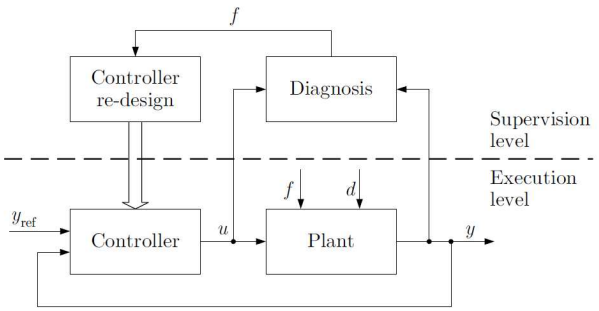

Figure 1.4: A fault-tolerant control system [6].

1.2.1 Fault Detection

To perform fault-detection, one compares the actual behavior of the plant to *expected* behavior, and raises an alarm if the two deviate. The expected behavior is usually determined from a *model* of the plant; when there are disturbances or the model is uncertain, the expected and actual outputs will not coincide exactly. In such cases, the control system designer must determine 'how different' the two signals are allowed to be before raising an alarm. There is a trade-off here: if the threshold is set too low, there will be many false alarms, and if the threshold is too high, some legitimate faults might be missed. The relative weighting of these two factors will depend on the application at hand. It is important to note that the above scheme is a *real-time* fault monitoring system, in that it runs concurrently with the system. This is in contrast to *scheduled* fault-diagnosis, where the system is checked at regular intervals according to some maintenance schedule to ensure that it is operating correctly. Real-time monitoring is important for safety-critical systems, so that a fault does not cause the system to become unstable, or progress to a state where it cannot be repaired.

There are a variety of methods to perform fault-detection. The most straightforward method is to use *physical redundancy*: the fault-prone components are replicated, and a comparison mechanism (e.g., majority voter) is used to determine which components are operating correctly. This scheme has the benefit of being simple to understand and implement, but could potentially be costly due to the replication of components.

An alternative (or perhaps complementary) method to diagnose faults is to analyze the behavior of the component over *time*, using a *model* of how the component is supposed to behave. This is known as *analytical redundancy* or *temporal redundancy*. The following example (motivated by [12]) illustrates this concept.

Example 1.4. Consider again the model of the car in Example 1.1, with internal state $v(t)$ (its speed) and dynamics given by (1.1). Now suppose that the sensor can potentially fail, so that it stops providing correct measurements of the speed. We would like to develop a mechanism to diagnose this failure.

Physical Redundancy: The most obvious way to detect failures would be to install *two* sensors, each of which provides an independent measurement of the speed, i.e., $s_1(t) = v(t)$ and $s_2(t) = v(t)$. If one of the sensors stops working, we would detect this by noting that the two measurements do not match. If we put in a third sensor, we could tell which sensor is malfunctioning simply by majority voting on the three sensors.

Analytical Redundancy: Instead of installing two sensors in order to detect a failure, suppose that we knew the model of the system (given by equation (1.1)), along with the acceleration input $a(t)$. Then, if the sensor is working (i.e., $s(t) = v(t)$), it must be the case that

$$s(t+1) = s(t) + a(t).$$

However, if the sensor is malfunctioning, this equation will (generally) not hold. Thus we would detect a failure *without* duplicating the sensor, but instead by using the *temporal* redundancy in the sensor measurements.

The above concept can be generalized to handle dynamical systems that are characterized by differential or difference equations (as is the case with many physical systems). In this course, we will show how to use linear system theory and linear algebra to treat such systems. By leveraging both temporal and physical redundancies, one may be able to operate the system and diagnose faults more efficiently.

Note that when one is concerned about attacks (i.e., worst-case faults), a properly designed fault-detection mechanism can be used to discover the source of attacks and raise an alarm so that appropriate defensive actions can be taken.

1.2.2 Fault Accommodation

While detecting and identifying faults quickly is very important, designing the system to function correctly despite faults is also a prime goal. The standard technique to achieve this is again via modular redundancy: if the fault-prone components are replicated and a voter compares their outputs, the correct components will prevail as long as less than half of the components are faulty. This is the approach taken in high-risk avionics such as the space-shuttle (which uses five redundant computers) [78], and the Boeing 777 (which uses three redundant computers, each of which consists of three internally redundant computing elements) [97]. Furthermore, at least one of the redundant elements in both of these applications is designed by a different team and using different technology than the other elements, so that a common coding problem or design defect will not affect all

of the elements in the same way. While this is a simple and effective method to achieve fault-tolerance, it clearly comes at a large cost due to the replication of technology and design effort by different teams.

In the second half of this course, we will study design techniques for control systems that are tolerant to different types of faults or attacks. Depending on the fault models that one is interested in, we will show how reliable control can be achieved without necessarily resorting to a large number of redundant components or controllers.

1.3 What is Covered in This Course?

The objective of this course is to provide an overview of different techniques for constructing reliable control systems. Due to the very broad applicability of this topic, this course will not attempt to discuss every possible application. Instead, we will deal with the underlying mathematical theory, analysis, and design of fault- and attack-tolerant systems. Topics will include model-based techniques for fault diagnosis, graph-based analysis techniques for linear systems, and the application of traditional fault-tolerance techniques to synthesizing reliable control mechanisms. The course will also cover recent research on the topics of tolerating packet dropouts in networked control systems, exchanging information in multi-agent systems despite the presence of malicious agents, and analyzing the vulnerability of large-scale complex systems (such as the power grid and the Internet) to attacks and failures.

The trajectory of the course will be as follows.

- **Mathematical background:** Before we can analyze a given system to determine its fault-tolerance properties, we will need some mathematical tools. Our primary tools will be linear algebra, linear system theory, and graph theory. We will review these concepts at the start of the course.
- **Fault-diagnosis:** Once we have the tools from linear system theory, we can start examining mathematical models of systems, and come up with systematic ways to analyze their outputs in order to detect faults.
- **Design:** We will then study ways to design control systems that are able to maintain proper behavior despite faults in the system. Our investigations will range from classical modular redundancy techniques (such as those used in aircraft) to more recent work on designing self-correcting controllers. We will also discuss topics in fault and attack-tolerant control of multi-agent systems, and time-permitting, end with a brief overview of work on studying the robustness of large-scale complex networks.

Chapter 2

Linear System Theory

In this course, we will be dealing primarily with *linear* systems, a special class of systems for which a great deal is known. During the first half of the twentieth century, linear systems were analyzed using *frequency domain* (e.g., Laplace and z-transform) based approaches in an effort to deal with issues such as noise and bandwidth issues in communication systems. While they provided a great deal of intuition and were sufficient to establish some fundamental results, frequency domain approaches presented various drawbacks when control scientists started studying more complicated systems (containing multiple inputs and outputs, nonlinearities, noise, and so forth). Starting in the 1950's (around the time of the space race), control engineers and scientists started turning to *state-space* models of control systems in order to address some of these issues. These time-domain approaches are able to effectively represent concepts such as the internal state of the system, and also present a method to introduce optimality conditions into the controller design procedure. We will be using the state-space (or "modern") approach to control almost exclusively in this course, and the purpose of this chapter is to review some of the essential concepts in this area.

2.1 Discrete-Time Signals

Given a field $\mathbb{F}$, a *signal* is a mapping from a set of numbers to $\mathbb{F}$; in other words, signals are simply functions of the set of numbers that they operate on. More specifically:

- A *discrete-time* signal f is a mapping from the set of integers $\mathbb{Z}$ to $\mathbb{F}$, and is denoted by $f[k]$ for $k \in \mathbb{Z}$. Each instant k is also known as a *time-step*.
- A *continuous-time* signal f is a mapping from the set of real numbers $\mathbb{R}$ to $\mathbb{F}$ and is denoted by $f(t)$ for $t \in \mathbb{R}$.

One can obtain discrete-time signals by *sampling* continuous-time signals. Specifically, suppose that we are interested in the value of the signal $f(t)$ at times $t = 0, T, 2T, \ldots,$

for some positive constant T. These values form a sequence $f(kT)$ for $k \in \mathbb{N}$, and if we simply drop the constant T from the notation (for convenience), we obtain the discrete-time sequence $f[k]$ for $k \in \mathbb{N}$. Since much of modern control deals with sampled versions of signals (due to the reliance on digital processing of such signals), we will be primarily working with discrete-time signals in this course, and focusing on the case where $\mathbb{F} = \mathbb{C}$ (the field of complex numbers).

2.2 Linear Time-Invariant Systems

Consider the system from Figure 1.1, with an input signal u and an output signal y. The system is either discrete-time or continuous-time, depending on the types of the signals. In our discussion, we will focus on the discrete-time case, but the results and definitions transfer in a straightforward manner to continuous-time systems.

In order to analyze and control the system, we will be interested in how the outputs respond to the inputs. We will be particularly interested in systems that satisfy the following property.

> **Definition 2.1** (Principle of Superposition)**.** Suppose that the output of the system is $y_1[k]$ in response to input $u_1[k]$ and $y_2[k]$ in response to input $u_2[k]$. The Principle of Superposition holds if the output of the system in response to the input $\alpha u_1[k] + \beta u_2[k]$ is $\alpha y_1[k] + \beta y_2[k]$, where α and β are arbitrary real numbers. Note that this must hold for *any* inputs $u_1[k]$ and $u_2[k]$.

The system is said to be **linear** if the Principle of Superposition holds. The system is said to be **time-invariant** if the output of the system is $y[k - \kappa]$ when the input is $u[k - \kappa]$ (i.e., a time-shifted version of the input produces an equivalent time-shift in the output). These concepts are illustrated in Fig. 2.1.

2.3 Mathematical Models

Mathematical models for many systems can be derived either from first principles (e.g., using Newton's Laws of motion for mechanical systems and Kirchoff's voltage and current laws for electrical systems). As noted earlier, many physical systems are inherently continuous-time, and thus their models would involve systems of differential equations. However, by *sampling* the system, one can essentially use discrete-time models to analyze such systems. In this course, we will predominantly be working with systems that can be represented by a certain mathematical form, and not focus too much on the explicit system itself. In other words, we will be interested in studying general properties and

Figure 2.1: (a) The Principle of Superposition. (b) The Time-Invariance Property.

analysis methods that can be applied to a *variety* of systems. To introduce the general mathematical system model that we will be considering, however, it will first be useful to consider a few examples.

2.3.1 A Simple Model for a Car

Consider again the model of the car from Chapter 1, with speed $v(t)$ at any time t, and acceleration input $a(t)$. Along the lines of Example 1.1, suppose that we *sample* the velocity of the car every T seconds, and that the acceleration is held constant between sampling times. We can then write

$$v[k+1] = v[k] + Ta[k], \tag{2.1}$$

where $v[k]$ is shorthand for $v(kT), k \in \mathbb{N}$, and $a[k]$ is the acceleration that is applied at time $t = kT$. Now, suppose that we wish to also consider the amount of fuel in the car at any given time-step k: let $g[k]$ denote this amount. We can consider a very simple model for fuel consumption, where the fuel decreases linearly with the distance traveled. Let $d[k]$ denote the distance traveled by the car between sampling times kT and $(k+1)T$, and recall from basic physics that under constant acceleration $a[k]$ and initial velocity $v[k]$, this distance is given by

$$d[k] = Tv[k] + \frac{T^2}{2}a[k].$$

Thus, if we let δ denote some coefficient that indicates how the fuel decreases with distance, we can write

$$g[k+1] = g[k] - \delta d[k] = g[k] - \delta T v[k] - \delta\frac{T^2}{2}a[k]. \tag{2.2}$$

Equations (2.1) and (2.2) together form a two-state model of a car. We can put them together concisely using matrix-vector notation as follows:

$$\underbrace{\begin{bmatrix} v[k+1] \\ g[k+1] \end{bmatrix}}_{\mathbf{x}[k+1]} = \begin{bmatrix} 1 & 0 \\ -\delta T & 1 \end{bmatrix} \underbrace{\begin{bmatrix} v[k] \\ g[k] \end{bmatrix}}_{\mathbf{x}[k]} + \begin{bmatrix} T \\ -\delta \frac{T^2}{2} \end{bmatrix} a[k] \ . \tag{2.3}$$

The state of this system is given by the vector $\mathbf{x}[k] = \begin{bmatrix} v[k] & g[k] \end{bmatrix}'$, and the input is the acceleration $a[k]$, as before. If we consider the speedometer that provides a measurement of the speed at every sampling instant, the output of the system would be

$$s[k] = v[k] = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}[k].$$

We could also have a fuel sensor that measures $g[k]$ at each time-step; in this case, we would have two outputs, given by the vector

$$\mathbf{y}[k] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}[k],$$

where the first component corresponds to the sensor measurement of the speed, and the second component corresponds to the fuel sensor.

2.3.2 Longitudinal Dynamics of an F-8 Aircraft

Consider a model for the (sampled) linearized longitudinal dynamics of an F-8 aircraft [87]:

$$\underbrace{\begin{bmatrix} V[k+1] \\ \gamma[k+1] \\ \alpha[k+1] \\ q[k+1] \end{bmatrix}}_{\mathbf{x}[k+1]} = \begin{bmatrix} 0.9987 & -3.2178 & -4.4793 & -0.2220 \\ 0 & 1 & 0.1126 & 0.0057 \\ 0 & 0 & 0.8454 & 0.0897 \\ 0.0001 & -0.0001 & -0.8080 & 0.8942 \end{bmatrix} \underbrace{\begin{bmatrix} V[k] \\ \gamma[k] \\ \alpha[k] \\ q[k] \end{bmatrix}}_{\mathbf{x}[k]} + \begin{bmatrix} -0.0329 \\ 0.0131 \\ -0.0137 \\ -0.0092 \end{bmatrix} \mathbf{u}[k], \tag{2.4}$$

where $V[k]$ is the velocity of the aircraft, $\gamma[k]$ is the flight-path angle, $\alpha[k]$ is the angle-of-attack, and $q[k]$ is the pitch rate. The input to the aircraft is taken to be the deflection of the elevator flaps.

The output of the system will depend on the sensors that are installed on the aircraft. For example, if there is a sensor to measure the velocity and the pitch rate, the output would be given by

$$\mathbf{y}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}[k].$$

2.3.3 Linear Feedback Shift Register

Consider, a *linear feedback shift register (LFSR)*, which is a digital circuit used to implement functions such as random number generators and "noise" sequences in computers.

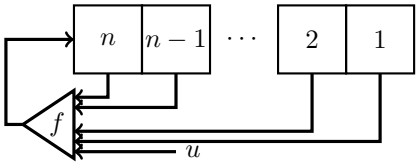

Figure 2.2: A Linear Feedback Shift Register

The LFSR consists of n registers that are chained together. At each clock-cycle k, each register takes the value of the next register in the chain. The last register's value is computed as a function of the values of other registers further up the chain, and perhaps with an additional input $u[k]$. If we denote the value stored in register i at time-step (or clock-cycle) k by $x_i[k]$, we obtain the model

$$\begin{aligned} x_1[k+1] &= x_2[k] \\ x_2[k+1] &= x_3[k] \\ &\vdots \\ x_{n-1}[k+1] &= x_n[k] \\ x_n[k+1] &= \alpha_0 x_1[k] + \alpha_1 x_2[k] + \cdots + \alpha_{n-1} x_n[k] + \beta u[k], \end{aligned} \tag{2.5}$$

for some scalars $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}, \beta$. These scalars and input are usually chosen so that the sequence of values generated by register 1 exhibits certain behavior (e.g., simulates the generation of a "random" sequence of values). Let $y[k] = x_1[k]$ denote the output of the system. One can write the above equations more compactly by defining the *state vector*

$$\mathbf{x}[k] \triangleq \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_n[k] \end{bmatrix}' ,$$

from which we obtain

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{n-1} \end{bmatrix}}_{\mathbf{A}} \mathbf{x}[k] + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix}}_{\mathbf{B}} u[k]$$

$$y[k] = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}[k].$$

2.4 State-Space Models

In the last section, we saw some examples of physical systems that can be modeled via a set of discrete-time equations, which were then put into a matrix-vector form. Such forms known as *state-space models* of linear systems, and can generally have multiple inputs and outputs to the system, with general system matrices.[1]

The state-space model for a discrete-time linear system is given by

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] \ . \end{aligned} \tag{2.6}$$

The state-space model of a continuous-time linear system is given by

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \ . \end{aligned} \tag{2.7}$$

- The vector $\mathbf{x}$ is called the **state vector** of the system. We will denote the number of states in the system by n, so that $\mathbf{x} \in \mathbb{R}^n$. The quantity n is often called the **order** or **dimension** of the system.
- In general, we might have multiple inputs $u_1, u_2, \ldots, u_m$ to the system. In this case, we can define an **input vector** $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}'$.
- In general, we might have multiple outputs $y_1, y_2, \ldots, y_p$. In this case, we can define the **output vector** $\mathbf{y} = \begin{bmatrix} y_1 & y_2 \cdots & y_p \end{bmatrix}'$. Note that each of these outputs represents a sensor measurement of some of the states of the system.
- The **system matrix A** is an $n \times n$ matrix representing how the states of the system affect each other.
- The **input matrix B** is an $n \times m$ matrix representing how the inputs to the system affect the states.
- The **output matrix C** is a $p \times n$ matrix representing the portions of the states that are measured by the outputs.
- The **feedthrough matrix D** is a $p \times m$ matrix representing how the inputs affect the outputs directly (i.e., without going through the states first).

[1] Although we will be focusing on linear systems, many practical systems are *nonlinear*. Since state-space models are time-domain representations of systems, they can readily capture nonlinear dynamics. When the states of the system stay close to some nominal operating point, nonlinear systems can often be *linearized*, bringing them into the form (2.6) or (2.7). We will not discuss nonlinear systems in too much further detail in this course.

2.4.1 Transfer Functions of Linear State-Space Models

While the state-space models (2.6) and (2.7) are a time-domain representation of systems, one can also convert them to the frequency domain by taking the z-transform (or Laplace transform in continuous-time). Specifically, if we take the z-transform of (2.6), we obtain:

$$\begin{aligned} z\mathbf{X}(z) - z\mathbf{x}(0) &= \mathbf{A}\mathbf{X}(z) + \mathbf{B}\mathbf{U}(z) \\ \mathbf{Y}(z) &= \mathbf{C}\mathbf{X}(z) + \mathbf{D}\mathbf{U}(z) \ . \end{aligned}$$

Note that this includes the initial conditions of all the states. The first equation can be rearranged to solve for $\mathbf{X}(z)$ as follows:

$$(z\mathbf{I} - \mathbf{A})\mathbf{X}(z) = z\mathbf{x}(0) + \mathbf{B}\mathbf{U}(z) \Leftrightarrow \mathbf{X}(z) = (z\mathbf{I} - \mathbf{A})^{-1} z\mathbf{x}(0) + (z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(z) \ .$$

Substituting this into the equation for $\mathbf{Y}(z)$, we obtain

$$\mathbf{Y}(z) = \mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1} z\mathbf{x}(0) + \left(\mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\right)\mathbf{U}(z) \ .$$

> The transfer function of the state-space model $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k]$, $\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k]$ (when $\mathbf{x}(0) = 0$) is
>
> $$\mathbf{H}(z) = \mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \ . \qquad (2.8)$$

Note that $\mathbf{H}(z)$ is a $p \times m$ matrix, and thus it is a generalization of the transfer function for standard single-input single-output systems. In fact, it is a matrix where entry i, j is a transfer function describing how the j–th input affects the i–th output.

Example 2.1. Calculate the transfer function for the state space model

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}}_{\mathbf{A}} \mathbf{x}[k] + \underbrace{\begin{bmatrix} 0 \\ 4 \end{bmatrix}}_{\mathbf{B}} u[k], \quad \mathbf{y}[k] = \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}[k] + \underbrace{\begin{bmatrix} 3 \\ 0 \end{bmatrix}}_{\mathbf{D}} u[k] \ .$$

$$\begin{aligned} \mathbf{H}(z) &= \mathbf{C}\left(z\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{D} \\ &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} z & -1 \\ 2 & z+3 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \end{bmatrix} \\ &= \frac{1}{z^2 + 3z + 2} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} z+3 & 1 \\ -2 & z \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{3z^2 + 10z + 9}{z^2 + 3z + 2} \\ \frac{1}{z+2} \end{bmatrix}. \end{aligned}$$

2.4.2 State-Space Realizations and Similarity Transformations

Suppose we have a linear system with transfer function $\mathbf{H}(z)$ (which can be a matrix, in general). We have seen that the transfer function is related to the matrices in the state space model via (2.8). Recall that the transfer function describes how the input to the system affects the output (when the initial state of the system is zero). In some sense, this might seem to indicate that the exact representation of the internal states of the system might be irrelevant, as long as the input-output behavior is preserved. In this section, we will see that there are multiple *state-space realizations* for a given system that correspond to the same transfer function.

Consider any particular state-space model of the form (2.6). Now, let us choose an arbitrary invertible $n \times n$ matrix $\mathbf{T}$, and define a new state vector

$$\bar{\mathbf{x}}[k] = \mathbf{T}\mathbf{x}[k] \ .$$

In other words, the states in the vector $\bar{\mathbf{x}}[k]$ are linear combinations of the states in the vector $\mathbf{x}[k]$. Since $\mathbf{T}$ is a constant matrix, we have

$$\begin{aligned}\bar{\mathbf{x}}[k+1] = \mathbf{T}\mathbf{x}[k+1] = \mathbf{T}\mathbf{A}\mathbf{x}[k] + \mathbf{T}\mathbf{B}\mathbf{u}[k] &= \underbrace{\mathbf{T}\mathbf{A}\mathbf{T}^{-1}}_{\bar{\mathbf{A}}}\bar{\mathbf{x}}[k] + \underbrace{\mathbf{T}\mathbf{B}}_{\bar{\mathbf{B}}}\mathbf{u}[k] \\ \mathbf{y} = \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] &= \underbrace{\mathbf{C}\mathbf{T}^{-1}}_{\bar{\mathbf{C}}}\bar{\mathbf{x}}[k] + \mathbf{D}\mathbf{u}[k] \ .\end{aligned}$$

Thus, after this transformation, we obtain the new state-space model

$$\begin{aligned}\bar{\mathbf{x}}[k+1] &= \bar{\mathbf{A}}\bar{\mathbf{x}}[k] + \bar{\mathbf{B}}\mathbf{u}[k] \\ \mathbf{y} &= \bar{\mathbf{C}}\bar{\mathbf{x}}[k] + \mathbf{D}\mathbf{u}[k] \ .\end{aligned}$$

Note that the inputs and outputs were *not* affected by this transformation; only the internal state vector and matrices changed. The transfer function corresponding to this model is given by

$$\begin{aligned}\bar{\mathbf{H}}(z) = \bar{\mathbf{C}}(z\mathbf{I} - \bar{\mathbf{A}})^{-1}\bar{\mathbf{B}} + \mathbf{D} &= \mathbf{C}\mathbf{T}^{-1}(z\mathbf{I} - \mathbf{T}\mathbf{A}\mathbf{T}^{-1})^{-1}\mathbf{T}\mathbf{B} + \mathbf{D} \\ &= \mathbf{C}\mathbf{T}^{-1}(z\mathbf{T}\mathbf{T}^{-\mathbf{1}} - \mathbf{T}\mathbf{A}\mathbf{T}^{-1})^{-1}\mathbf{T}\mathbf{B} + \mathbf{D} \\ &= \mathbf{C}\mathbf{T}^{-1}\mathbf{T}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{T}^{-1}\mathbf{T}\mathbf{B} + \mathbf{D} \\ &= \mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \\ &= \mathbf{H}(z) \ .\end{aligned}$$

Thus the transfer function for the realization with state-vector $\bar{\mathbf{x}}$ is the same as the transfer function for the realization with state-vector $\mathbf{x}$. For this reason, the transformation $\bar{\mathbf{x}} = \mathbf{T}\mathbf{x}$ is called a **similarity transformation**. Since $\mathbf{T}$ can be *any* invertible matrix, and since there are an infinite number of invertible $n \times n$ matrices to choose from, we see that there are an **infinite number of realizations** for any given transfer function $\mathbf{H}(z)$.

Similarity transformations are a very useful tool to analyze the behavior of linear systems, as we will see in later sections.

2.5 Stability of Linear Systems

Consider the system

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] \ , \tag{2.9}$$

without any inputs. This is known as an *autonomous* system. The following definition plays a central role in control theory.

Definition 2.2 (Stability)**.** The linear system $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$ is said to be *stable* if

$$\lim_{k\to\infty} \mathbf{x}[k] = \mathbf{0}$$

starting from any initial state $\mathbf{x}[0]$.

To obtain conditions for stability, it is first instructive to consider the scalar system $x[k+1] = \alpha x[k]$, where $\alpha \in \mathbb{R}$. Since $x[1] = \alpha x[0]$, $x[2] = \alpha x[1] = \alpha^2 x[0]$, and so forth, we have $x[k] = \alpha^k x[0]$. Now, in order for $x[k]$ to go to zero regardless of the value of $x[0]$, we must have $\alpha^k \to 0$ as $k \to \infty$, and this happens if and only if $|\alpha| < 1$. This is the necessary and sufficient condition for stability of a scalar linear system.

One can extend this to general state-space models of the form (2.9). To give the main idea of the proof, suppose that $\mathbf{A}$ is diagonalizable and write

$$\mathbf{A} = \mathbf{T}\Lambda\mathbf{T}^{-1} = \mathbf{T}\begin{bmatrix} \lambda_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \lambda_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \lambda_n \end{bmatrix}\mathbf{T}^{-1},$$

where each λ_i is an eigenvalue of $\mathbf{A}$. It is easy to verify that

$$\mathbf{x}[k] = \mathbf{A}^k\mathbf{x}[0] = \mathbf{T}\Lambda^k\mathbf{T}^{-1}\mathbf{x}[0] = \mathbf{T}\begin{bmatrix} \lambda_1^k & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \lambda_2^k & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \lambda_n^k \end{bmatrix}\mathbf{T}^{-1}\mathbf{x}[0].$$

This expression goes to zero for any $\mathbf{x}[0]$ if and only if $|\lambda_i| < 1$ for all $i \in \{1, 2, \ldots, n\}$. The proof for the most general case (where $\mathbf{A}$ is not diagonalizable) can be obtained by considering the Jordan form of $\mathbf{A}$ (see Appendix A.4.5); we will omit the mathematical details because they do not add to much to the discussion or understanding here. Thus, we obtain the following fundamental result.

Theorem 2.1. *The linear system* $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$ *is stable if and only if all eigenvalues of* $\mathbf{A}$ *have magnitude smaller than* 1 *(i.e., they are contained within the open unit circle in the complex plane).*

Frequently, the system under consideration is not stable (i.e., the $\mathbf{A}$ matrix contains eigenvalues of magnitude larger than 1), and the objective is to choose the inputs to the system so that $\mathbf{x}[k] \to 0$ as $k \to \infty$. We will study conditions under which this is possible in the next few sections.

2.6 Properties of Linear Systems

We now turn our attention to analyzing the state-space model (2.6) for the purpose of controlling the system. There are several properties of such systems that we will be studying.

2.6.1 Controllability

Definition 2.3 (Controllability)**.** The system (2.6) is said to be *controllable* if, for any initial state $\mathbf{x}[0]$ and any desired state $\mathbf{x}^*$, there is a nonnegative integer L and a sequence of inputs $\mathbf{u}[0], \mathbf{u}[1], \ldots, \mathbf{u}[L]$ such that $\mathbf{x}[L+1] = \mathbf{x}^*$.

To derive conditions on the system matrices $\mathbf{A}$ and $\mathbf{B}$ under which the system is controllable, suppose we start at some state $\mathbf{x}[0]$ at time-step 0, and note that:

$$\begin{aligned}
\mathbf{x}[1] &= \mathbf{A}\mathbf{x}[0] + \mathbf{B}\mathbf{u}[0] \\
\mathbf{x}[2] &= \mathbf{A}\mathbf{x}[1] + \mathbf{B}\mathbf{u}[1] = \mathbf{A}^2\mathbf{x}[0] + \mathbf{A}\mathbf{B}\mathbf{u}[0] + \mathbf{B}\mathbf{u}[1] \\
\mathbf{x}[3] &= \mathbf{A}\mathbf{x}[2] + \mathbf{B}\mathbf{u}[2] = \mathbf{A}^3\mathbf{x}[0] + \mathbf{A}^2\mathbf{B}\mathbf{u}[0] + \mathbf{A}\mathbf{B}\mathbf{u}[1] + \mathbf{B}\mathbf{u}[2].
\end{aligned}$$

Continuing in this way, we can write

$$\mathbf{x}[L+1] - \mathbf{A}^{L+1}\mathbf{x}[0] = \underbrace{\begin{bmatrix}\mathbf{A}^L\mathbf{B} & \mathbf{A}^{L-1}\mathbf{B} & \cdots & \mathbf{B}\end{bmatrix}}_{\mathcal{C}_L} \underbrace{\begin{bmatrix}\mathbf{u}[0] \\ \mathbf{u}[1] \\ \vdots \\ \mathbf{u}[L]\end{bmatrix}}_{\mathbf{u}[0:L]}.$$

The matrix $\mathcal{C}_L$ is called the *controllability matrix* for the pair $(\mathbf{A}, \mathbf{B})$. In order to go from $\mathbf{x}[0]$ to any value $\mathbf{x}[L+1]$, it must be the case that

$$\mathbf{x}[L+1] - \mathbf{A}^{L+1}\mathbf{x}[0] \in \mathcal{R}\left(\mathcal{C}_L\right),$$

where $\mathcal{R}(\cdot)$ denotes the range space of a matrix (see Appendix A.4.1). If we want to be able to go from any arbitrary initial state to any other arbitrary final state in $L+1$ time-steps, it must be the case that $\mathcal{R}\left(\mathcal{C}_L\right) = \mathbb{R}^n$, which is equivalent to saying that $\text{rank}(\mathcal{C}_L) = n$. If this condition is satisfied, then we can find n linearly independent columns within $\mathcal{C}_L$, and select the inputs $\mathbf{u}[0], \mathbf{u}[1], \ldots, \mathbf{u}[L]$ to combine those columns in such a way that we can obtain any $\mathbf{x}[L+1]$. However, if the rank of $\mathcal{C}_L$ is less than n, then there might be some $\mathbf{x}[L+1]$ that we cannot obtain. In this case, we can wait a few more time-steps and hope that the rank of the matrix $\mathcal{C}_L$ increases to n. How long should we wait?

To answer this question, note that the rank of $\mathcal{C}_L$ is a nondecreasing function of L, and bounded above by n. Suppose ν is the first integer for which $\text{rank}(\mathcal{C}_\nu) = \text{rank}(\mathcal{C}_{\nu-1})$. This is equivalent to saying that the extra columns in $\mathcal{C}_\nu$ (given by $\mathbf{A}^\nu\mathbf{B}$) are all linearly dependent on the columns in $\mathcal{C}_{\nu-1}$; mathematically, there exists a matrix $\mathbf{K}$ such that

$$\mathbf{A}^\nu\mathbf{B} = \begin{bmatrix}\mathbf{A}^{\nu-1}\mathbf{B} & \mathbf{A}^{\nu-2}\mathbf{B} & \cdots & \mathbf{B}\end{bmatrix}\mathbf{K} \ .$$

In turn, this implies that

$$\mathbf{A}^{\nu+1}\mathbf{B} = \mathbf{A}\mathbf{A}^\nu\mathbf{B} = \mathbf{A}\mathcal{C}_{\nu-1}\mathbf{K} = \begin{bmatrix}\mathbf{A}^\nu\mathbf{B} & \mathbf{A}^{\nu-1}\mathbf{B} & \cdots & \mathbf{A}\mathbf{B}\end{bmatrix}\mathbf{K} \ ,$$

and so the matrix $\mathbf{A}^{\nu+1}\mathbf{B}$ can be written as a linear combination of the columns in $\mathcal{C}_\nu$ (which can themselves be written as linear combinations of columns in $\mathcal{C}_{\nu-1}$). Continuing in this way, we see that

$$\text{rank}(\mathcal{C}_0) < \text{rank}(\mathcal{C}_1) < \cdots < \text{rank}(\mathcal{C}_{\nu-1}) = \text{rank}(\mathcal{C}_\nu) = \text{rank}(\mathcal{C}_{\nu+1}) = \cdots \ ,$$

i.e., the rank of $\mathcal{C}_L$ monotonically increases with L until $L = \nu - 1$, at which point it stops increasing. Since the matrix $\mathbf{B}$ contributes $\text{rank}(\mathbf{B})$ linearly independent columns to the controllability matrix, the rank of the controllability matrix can increase for at most $n - \text{rank}(\mathbf{B})$ time-steps before it reaches its maximum value, and so the integer ν is upper bounded as $\nu \leq n - \text{rank}(\mathbf{B}) + 1$. This yields the following result.

Theorem 2.2. *Consider the system* (2.6)*, where* $\mathbf{x}[k] \in \mathbb{R}^n$*. For any positive integer* L*, define the* **controllability matrix**

$$\mathcal{C}_L = \begin{bmatrix}\mathbf{A}^L\mathbf{B} & \mathbf{A}^{L-1}\mathbf{B} & \cdots & \mathbf{B}\end{bmatrix} \ . \tag{2.10}$$

The system is controllable if and only if $\text{rank}(\mathcal{C}_{n-\text{rank}(\mathbf{B})}) = n$.

In the linear systems literature, the integer ν is called the *controllability index* of the pair $(\mathbf{A}, \mathbf{B})$. For simplicity, one often uses the fact that $n - \text{rank}(\mathbf{B}) \leq n-1$, and just checks the rank of $\mathcal{C}_{n-1}$ to verify controllability.

Example 2.2. Consider the system given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The controllability matrix for this system is

$$\mathcal{C}_{n-\text{rank}(\mathbf{B})} = \mathcal{C}_1 = \begin{bmatrix} \mathbf{AB} & \mathbf{B} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}.$$

This matrix has rank equal to 2, and so the system is controllable. Specifically, since

$$\mathbf{x}[2] = \mathbf{A}^2\mathbf{x}[0] + \mathcal{C}_1 \begin{bmatrix} u[0] \\ u[1] \end{bmatrix},$$

we can go from any initial state $\mathbf{x}[0]$ to any state $\mathbf{x}[2]$ at time-step 2 simply by applying the inputs

$$\begin{bmatrix} u[0] \\ u[1] \end{bmatrix} = \mathcal{C}_1^{-1} \left(\mathbf{x}[2] - \mathbf{A}^2\mathbf{x}[0] \right).$$

Example 2.3. Consider the system given by

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The controllability matrix for this system is

$$\mathcal{C}_{n-\text{rank}(\mathbf{B})} = \mathcal{C}_1 = \begin{bmatrix} \mathbf{AB} & \mathbf{B} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

This matrix only has rank 1, and thus the system is not controllable. Specifically, starting from an initial state of zero, one can never drive the second state to a nonzero value.

2.6.2 Observability

Definition 2.4 (Observability)**.** The system is said to be *observable* if, for any initial state $\mathbf{x}[0]$, and for any *known* sequence of inputs $\mathbf{u}[0], \mathbf{u}[1], \ldots$, there is a positive integer L such that $\mathbf{x}[0]$ can be recovered from the outputs $\mathbf{y}[0], \mathbf{y}[1], \ldots, \mathbf{y}[L]$.

To relate the concept of observability to the system matrices, if we simply iterate the output equation in (2.6) for $L+1$ time-steps, we get:

$$\underbrace{\begin{bmatrix}\mathbf{y}[0]\\ \mathbf{y}[1]\\ \mathbf{y}[2]\\ \vdots\\ \mathbf{y}[L]\end{bmatrix}}_{\mathbf{y}[0:L]} = \underbrace{\begin{bmatrix}\mathbf{C}\\ \mathbf{CA}\\ \mathbf{CA}^2\\ \vdots\\ \mathbf{CA}^L\end{bmatrix}}_{\mathcal{O}_L}\mathbf{x}[0] + \underbrace{\begin{bmatrix}\mathbf{D} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{CB} & \mathbf{D} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{CAB} & \mathbf{CB} & \mathbf{D} & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathbf{CA}^{L-1}\mathbf{B} & \mathbf{CA}^{L-2}\mathbf{B} & \mathbf{CA}^{L-3}\mathbf{B} & \cdots & \mathbf{D}\end{bmatrix}}_{\mathcal{J}_L}\underbrace{\begin{bmatrix}\mathbf{u}[0]\\ \mathbf{u}[1]\\ \mathbf{u}[2]\\ \vdots\\ \mathbf{u}[L]\end{bmatrix}}_{\mathbf{u}[0:L]}. \quad (2.11)$$

The matrix $\mathcal{O}_L$ is called the *observability* matrix for the pair $(\mathbf{A}, \mathbf{C})$, and the matrix $\mathcal{J}_L$ is called the *invertibility matrix* for the tuple $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$; this terminology will become clear in next section. Rearranging the above equation, we obtain

$$\mathbf{y}[0:L] - \mathcal{J}_L\mathbf{u}[0:L] = \mathcal{O}_L\mathbf{x}[0].$$

Since the inputs to the system are assumed to be *known* in this case, the entire left hand side of the above equation is known. Thus, the objective is to uniquely recover $\mathbf{x}[0]$ from the above equation; this is possible if and only if $\text{rank}(\mathcal{O}_L) = n$. In this case, the system is said to be *observable*. As in the case of the controllability matrix, one can show that there exists an integer μ such that

$$\text{rank}(\mathcal{O}_0) < \text{rank}(\mathcal{O}_1) < \cdots < \text{rank}(\mathcal{O}_{\mu-1}) = \text{rank}(\mathcal{O}_\mu) = \text{rank}(\mathcal{O}_{\mu+1}) = \cdots \ ,$$

i.e., the rank of $\mathcal{O}_L$ monotonically increases with L until $L = \mu - 1$, at which point it stops increasing. Since the matrix $\mathbf{C}$ contributes $\text{rank}(\mathbf{C})$ linearly independent rows to the controllability matrix, the rank of the observability matrix can increase for at most $n - \text{rank}(\mathbf{C})$ time-steps before it reaches its maximum value, and so the integer μ is upper bounded as $\mu \leq n - \text{rank}(\mathbf{C}) + 1$. This yields the following result.

Theorem 2.3. *Consider the system* (2.6)*, where* $\mathbf{x}[k] \in \mathbb{R}^n$*. For any positive integer* L*, define the* **observability matrix**

$$\mathcal{O}_L = \begin{bmatrix}\mathbf{C}\\ \mathbf{CA}\\ \mathbf{CA}^2\\ \vdots\\ \mathbf{CA}^L\end{bmatrix} . \quad (2.12)$$

The system is observable if and only if $\text{rank}(\mathcal{O}_{n-\text{rank}(\mathbf{C})}) = n$.

The integer μ is called the *observability index* of the system.

Remark 2.1. It is easy to show that the pair $(\mathbf{A}, \mathbf{C})$ is observable if and only if the pair $(\mathbf{A}', \mathbf{C}')$ is controllable; simply transpose the observability matrix and rearrange the columns (which does not change the rank of the observability matrix) to resemble the controllability matrix. Thus, controllability and observability are known as *dual* properties of linear systems. Note that this does *not* mean that a given system that is controllable is also observable (since the controllability matrix involves the matrix $\mathbf{B}$, and the observability matrix involves $\mathbf{C}$).

Example 2.4. Consider the pair $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}$, with no inputs to the system. The observability matrix for this pair is

$$\mathcal{O}_{n-\text{rank}(\mathbf{C})} = \mathcal{O}_1 = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

which has rank 2, and thus the pair is observable. The initial state of the system can be recovered as follows:

$$\mathbf{y}[0:1] = \mathcal{O}_1 \mathbf{x}[0] \;\; \Rightarrow \mathcal{O}_1^{-1}\mathbf{y}[0:1] = \mathbf{x}[0].$$

2.6.3 Invertibility

In the last section, we assumed that the inputs to the system were completely known; this allowed us to subtract them out from the outputs of the system, and then recover the initial state (provided that the system was observable). However, there may be cases where some or all of the inputs to the system are completely unknown and arbitrary, the system is called a *linear system with unknown inputs* [35]. For such systems, it is often of interest to "invert" the system in order to reconstruct some or all of the unknown inputs (assuming that the initial state is known), and this problem has been studied under the moniker of *dynamic system inversion* [72, 75]. This concept will be very useful when we discuss the diagnosis of faults and attacks in linear systems, since such events can often be modeled via unknown inputs to the system.

> **Definition 2.5** (Invertibility)**.** The system (2.6) is said to have an L-delay inverse if it is possible to uniquely recover the input $\mathbf{u}[k]$ from the outputs of the system up to time-step $\mathbf{y}[k+L]$ (for some nonnegative integer L), assuming that the initial state $\mathbf{x}[0]$ is known. The system is *invertible* if it has an L-delay inverse for some finite L. The least integer L for which an L-delay inverse exists is called the inherent delay of the system.

To illustrate the idea, let us start by considering an example.

Example 2.5. Consider the system

$$\mathbf{x}[k+1] = \begin{bmatrix} 0 & 1 \\ 2 & -3 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}[k], \;\; \mathbf{y}[k] = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}[k].$$

Clearly $\mathbf{y}[k]$ provides no information about $\mathbf{u}[k]$. Similarly, $\mathbf{y}[k+1] = \mathbf{CAx}[k]+\mathbf{CBu}[k] = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}[k]$, which again does not contain any information about $\mathbf{u}[k]$. However, $\mathbf{y}[k+2] = \mathbf{CA}^2\mathbf{x}[k] + \mathbf{CABu}[k] + \mathbf{CBu}[k+1] = \begin{bmatrix} 2 & -3 \end{bmatrix} \mathbf{x}[k] + \mathbf{u}[k]$, and thus we can recover $\mathbf{u}[k]$ as $\mathbf{y}[k+2] - \begin{bmatrix} 2 & -3 \end{bmatrix} \mathbf{x}[k]$, provided that we knew $\mathbf{x}[k]$. Specifically, if we knew $\mathbf{x}[0]$, we would be able to recover $\mathbf{u}[0]$ from the above expression, and then we could determine $\mathbf{x}[1] = \mathbf{Ax}[0] + \mathbf{Bu}[0]$. We could then repeat the procedure to find $\mathbf{u}[k]$ (and $\mathbf{x}[k]$) for all $k \in \mathbb{N}$).

To come up with a systematic procedure to analyze invertibility of systems, consider again the output of the linear system (2.6) over $L+1$ time-steps for any nonnegative integer L; rearranging (2.11), we see that

$$\mathbf{y}[0:L] - \mathcal{O}_L \mathbf{x}[0] = \mathcal{J}_L \mathbf{u}[0:L], \tag{2.13}$$

where the left side is now assumed to be completely known. The matrix $\mathcal{J}_L$ will completely characterize our ability to recover the inputs to the system. First, note from (2.11) that the last Lm columns of $\mathcal{J}_L$ have the form $\begin{bmatrix} \mathbf{0} \\ \mathcal{J}_{L-1} \end{bmatrix}$. The rank of $\mathcal{J}_L$ is thus equal to the number of linearly independent columns from the last Lm columns (given by $\text{rank}(\mathcal{J}_{L-1})$), plus any additional linearly independent columns from the first m columns. Thus,

$$\text{rank}(\mathcal{J}_L) \leq m + \text{rank}\left(\begin{bmatrix} \mathbf{0} \\ \mathcal{J}_{L-1} \end{bmatrix} \right) = m + \text{rank}(\mathcal{J}_{L-1}), \tag{2.14}$$

for all nonnegative integers L, where we define $\text{rank}(\mathcal{J}_{-1}) = 0$ for convenience.

Now, note that the input $\mathbf{u}[0]$ enters equation (2.13) through the first m columns of the matrix $\mathcal{J}_L$. Thus, in order to recover $\mathbf{u}[0]$, it must be the case that:

1. The first m columns of $\mathcal{J}_L$ are linearly independent of each other (otherwise there exists some nonzero $\mathbf{u}[0]$ such that the first m columns times $\mathbf{u}[0]$ is the zero vector, which is indistinguishable from the case where $\mathbf{u}[0] = \mathbf{0}$).
2. The first m columns of $\mathcal{J}_L$ are linearly independent of *all other* columns of $\mathcal{J}_L$ (otherwise, there exists some nonzero $\mathbf{u}[0]$ and some nonzero $\mathbf{u}[1:L]$ such that $\mathcal{J}_L \mathbf{u}[0:L] = \mathbf{0}$, which is indistinguishable from case where $\mathbf{u}[0:L] = \mathbf{0}$).

If both of the above conditions are satisfied, then one can find a matrix $\mathbf{P}$ such that $\mathbf{P}\mathcal{J}_L = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix}$, which means that the input $\mathbf{u}[0]$ can be recovered as

$$\mathbf{P}\left(\mathbf{y}[0:L] - \mathcal{O}_L \mathbf{x}[0]\right) = \mathbf{P}\mathcal{J}_L \mathbf{u}[0:L] = \mathbf{u}[0] \;\; .$$

Since $\mathbf{u}[0]$ is now known, one can obtain $\mathbf{x}[1] = \mathbf{Ax}[0]+\mathbf{Bu}[0]$, and can repeat the process to obtain $\mathbf{u}[k]$ for all positive integers k.

The condition that the first m columns of $\mathcal{J}_L$ be linearly independent of all other columns and of each other is equivalent to saying that

$$\text{rank}(\mathcal{J}_L) = m + \text{rank}\left(\begin{bmatrix}\mathbf{0} \\ \mathcal{J}_{L-1}\end{bmatrix}\right) = m + \text{rank}(\mathcal{J}_{L-1}),$$

i.e., equality holds in (2.14). Thus, to check for invertibility of the linear system (2.6), we can start with $\mathcal{J}_0 = \mathbf{D}$, and increase L until we find $\text{rank}(\mathcal{J}_L) = m + \text{rank}(\mathcal{J}_{L-1})$. At what point should we stop increasing L and announce that the system is *not* invertible? To answer this question, we will use the following argument from [72]. First, suppose that the system is not invertible for $L = 0, 1, \ldots, n$. Then from (2.14), we have

$$\text{rank}(\mathcal{J}_n) \leq m - 1 + \text{rank}(\mathcal{J}_{n-1)} \leq 2(m-1) + \text{rank}(\mathcal{J}_{n-2}) \leq \cdots \leq (n+1)(m-1).$$

Note that we use $m-1$ in each of the above inequalities because we know that (2.14) holds with strict inequality (due to the fact that the system is not invertible for those delays). Based on the above inequality, the null space of $\mathcal{J}_n$ has dimension

$$(n+1)m - \text{rank}(\mathcal{J}_n) \geq (n+1)m - (n+1)(m-1) = n+1.$$

Let $\mathbf{N}$ be a matrix whose columns from a basis for the null space of $\mathcal{J}_n$, and note that $\mathbf{N}$ has at least $n+1$ columns. Thus, any input of the form $\mathbf{u}[0:n] = \mathbf{N}\mathbf{v}$ for some vector $\mathbf{v}$ would produce $\mathcal{J}_n\mathbf{u}[0:n] = \mathcal{J}_n\mathbf{N}\mathbf{v} = \mathbf{0}$. Now, also note that

$$\mathbf{x}[n+1] = \mathbf{A}^{n+1}\mathbf{x}[0] + \mathcal{C}_n\mathbf{u}[0:n] = \mathbf{A}^{n+1}\mathbf{x}[0] + \mathcal{C}_n\mathbf{N}\mathbf{v}.$$

Note that the matrix $\mathcal{C}_n\mathbf{N}$ has n rows and at least $n+1$ columns; thus it has a null space of dimension at least one. Thus, if we pick the vector $\mathbf{v}$ to be any vector in this null space, we see that $\mathbf{x}[n+1] = \mathbf{A}^{n+1}\mathbf{x}[0]$ and $\mathbf{y}[0:n] = \mathcal{O}_n\mathbf{x}[0]$. In other words, the input sequence $\mathbf{u}[0:n] = \mathbf{N}\mathbf{v}$ chosen in this way produces the same output over $n+1$ time-steps as the input sequence $\mathbf{u}[0:n] = \mathbf{0}$, and also leaves the state $\mathbf{x}[n+1]$ in the same position as the all zero input. If the input is $\mathbf{u}[k] = \mathbf{0}$ for all $k \geq n+1$, we see that we can never determine whether $\mathbf{u}[0:n] = \mathbf{N}\mathbf{v}$ or $\mathbf{u}[0:n] = \mathbf{0}$. Thus, if the system is not invertible for $L = n$, it is never invertible.

Theorem 2.4 ([72])**.** *Consider the system* (2.6)*, where* $\mathbf{x}[k] \in \mathbb{R}^n$ *and* $\mathbf{u}[k] \in \mathbb{R}^m$*. The system is invertible with delay* L *if and only if*

$$\textit{rank}(\mathcal{J}_L) = m + \textit{rank}(\mathcal{J}_{L-1}), \tag{2.15}$$

for some $L \leq n$*, where* $\textit{rank}(J_{-1})$ *is defined to be zero.*

It is worth noting that the upper bound on the inherent delay was improved in [94] to be $L = n - \text{nullity}(\mathbf{D}) + 1$; the proof is quite similar to the one above.

Example 2.6. Consider the F-8 aircraft given by equation (2.4), and suppose that the actuator on the aircraft could be faulty, whereby the actual input that is applied to the aircraft is different from the specified input. Mathematically, this can be modeled by setting the input to be $\mathbf{u}[k]+\mathbf{f}[k]$, where $\mathbf{u}[k]$ is the specified input, and $\mathbf{f}[k]$ is a additive error caused by the fault. The dynamics of the aircraft then become

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{B}\mathbf{f}[k].$$

where the $\mathbf{A}$ and $\mathbf{B}$ matrices are specified in (2.4). Suppose that there is a single sensor on the aircraft that measures the pitch rate, i.e.,

$$\mathbf{y}[k] = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}[k].$$

Assuming that the initial state of the aircraft is known, is it possible to determine the fault input $\mathbf{f}[k]$ by looking at the output of the system? This is equivalent to asking whether the input $\mathbf{f}[k]$ is invertible. To answer this, we try to find an $L \leq n$ such that (2.15) holds. For $L = 0$, we have $\mathcal{J}_0 = \mathbf{D} = \mathbf{0}$, and thus the condition is not satisfied. For $L = 1$, we have

$$\mathcal{J}_1 = \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{CB} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -0.0092 & 0 \end{bmatrix},$$

which has a rank of 1. Thus, $\text{rank}(\mathcal{J}_1) - \text{rank}(\mathcal{J}_0) = 1$, and the system is invertible with delay 1.

A drawback of the above analysis is that the initial state of the system is assumed to be known, and furthermore, the state at future time-steps is obtained via the estimate of the input. However, if there is noise in the system, this may not provide an accurate representation of future states. In the next section, we will study relax the condition on knowledge of the initial state.

2.6.4 Strong Observability

While the notions of observability and invertibility deal with the separate relationships between the initial states and the output, and between the input and the output, respectively, they do not consider the relationship between the states and input (taken together) and the output. To deal with this, the following notion of *strong observability* has been established in the literature (e.g., see [51, 35, 65, 90]).

> **Definition 2.6** (Strong Observability)**.** A linear system of the form (2.6) is said to be *strongly observable* if, for any initial state $\mathbf{x}[0]$ and any *unknown* sequence of inputs $\mathbf{u}[0], \mathbf{u}[1], \ldots$, there is a positive integer L such that $\mathbf{x}[0]$ can be recovered from the outputs $\mathbf{y}[0], \mathbf{y}[1], \ldots, \mathbf{y}[L]$.

By the linearity of the system, the above definition is equivalent to saying that $\mathbf{y}[k] = 0$ for all k implies $\mathbf{x}[0] = 0$ (regardless of the values of the unknown inputs $\mathbf{u}[k]$).

Recall that observability and invertibility of the system could be determined by examining the observability and invertibility matrices of the system (separately); in order to characterize strong observability, we must examine the relationship between the observability and invertibility matrices. Also recall that in order for the system to be invertible, the columns of the matrix multiplying $\mathbf{u}[0]$ in (2.11) needed to be linearly independent of each other and of the columns multiplying the other unknown quantities (i.e., $\mathbf{u}[1:L]$) in that equation. Using an identical argument, we see that the initial state $\mathbf{x}[0]$ can be recovered from (2.11) if and only if

$$\operatorname{rank}\left(\begin{bmatrix}\mathcal{O}_L & \mathcal{J}_L\end{bmatrix}\right) = n + \operatorname{rank}\left(\mathcal{J}_L\right)$$

for some nonnegative integer L; in other words, all columns of the observability matrix must be linearly independent of each other, and of all columns of the invertibility matrix.

Once again, one can ask if there is an upper bound on the number of time-steps that one would have to wait for before the above condition is satisfied (if it is satisfied at all). There is, in fact, such a bound, and the following derivation comes from [76].

First, a state $\mathbf{x}[0]$ is said to be *weakly unobservable* over $L+1$ time-steps if there exists an input sequence $\mathbf{u}[0:L]$ such that $\mathbf{y}[0:L] = \mathbf{0}$. Let Σ_L denote the set of all weakly unobservable states over $L+1$ time-steps (note that this set forms a subspace of $\mathbb{R}^n$). It is easy to see that

$$\Sigma_{L+1} \subseteq \Sigma_L \tag{2.16}$$

for all nonnegative integers L: if the state cannot be reconstructed after viewing the outputs over $L+2$ time-steps, it cannot be reconstructed after viewing the outputs after just $L+1$ time-steps. Next, let β denote the first nonnegative integer for which $\Sigma_\beta = \Sigma_{\beta+1}$. If $\mathbf{x}_0$ is any state in $\Sigma_{\beta+1}$, there must be an input $\mathbf{u}_0$ such that

$$\mathbf{x}_1 \triangleq \mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{u}_0 \in \Sigma_\beta;$$

this is because starting from $\mathbf{x}_0$, the input sequence starting with $\mathbf{u}_0$ causes the output to be zero for $\beta+2$ time-steps, and leads through the state $\mathbf{x}_1$. But, since $\Sigma_\beta = \Sigma_{\beta+1}$, we know that starting from $\mathbf{x}_1$ there is an input sequence that keeps the output zero for $\beta+2$ time-steps. This means that $\mathbf{x}_0 \in \Sigma_{\beta+2}$, because if we start from $\mathbf{x}_0$, we can apply $\mathbf{u}_0$ to go to $\mathbf{x}_1$, and then apply the input that keeps the output zero for $\beta+2$ time-steps. So we have shown that $\mathbf{x}_0 \in \Sigma_{\beta+1} \Rightarrow \mathbf{x}_0 \in \Sigma_{\beta+2}$, or equivalently $\Sigma_{\beta+1} \subseteq \Sigma_{\beta+2}$. From (2.16) we see that the opposite inclusion also holds, and so we have $\Sigma_{\beta+1} = \Sigma_{\beta+2}$. Continuing in this way, we see that

$$\Sigma_0 \supset \Sigma_1 \supset \Sigma_2 \supset \cdots \supset \Sigma_\beta = \Sigma_{\beta+1} = \cdots.$$

Since all of these spaces are subspaces of $\mathbb{R}^n$, we see that the dimension of the space can decrease at most n times, and so we have $\beta \leq n$. This leads to the following result.

Theorem 2.5. *Consider the system* (2.6) *with* $\mathbf{x}[k] \in \mathbb{R}^n$. *The system is strongly observable if and only if*

$$rank\left(\begin{bmatrix}\mathcal{O}_L & \mathcal{J}_L\end{bmatrix}\right) = n + rank(\mathcal{J}_L) \tag{2.17}$$

for some $L \leq n$.

Note that if the system is strongly observable, and the matrix $\begin{bmatrix}\mathbf{B}\\ \mathbf{D}\end{bmatrix}$ is full column rank, one can recover the unknown inputs as well. This is because we can obtain $\mathbf{x}[k]$ from $\mathbf{y}[k : k+L]$ for some $L \leq n$, and also $\mathbf{x}[k+1]$ from $\mathbf{y}[k+1 : k+L+1]$. Rearranging (2.6), we obtain

$$\begin{bmatrix}\mathbf{x}[k+1] - \mathbf{A}\mathbf{x}[k]\\ \mathbf{y}[k]\end{bmatrix} = \begin{bmatrix}\mathbf{B}\\ \mathbf{D}\end{bmatrix}\mathbf{u}[k],$$

and this uniquely specifies $\mathbf{u}[k]$.

Example 2.7. Consider again the F-8 from Example 2.6. To check whether one can recover the fault input $\mathbf{f}[k]$ can be recovered, regardless of the states of the system, we check whether the system is strongly observable. Specifically, for $L = n$, we have

$$\mathcal{O}_n = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.0001 & -0.0001 & -0.8080 & 0.8942 \\ 0.0001 & -0.0004 & -1.4058 & 0.7271 \\ 0.0002 & -0.0009 & -1.7765 & 0.5240 \\ 0.0002 & -0.0015 & -1.9261 & 0.3092 \end{bmatrix},$$

$$\mathcal{J}_n = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -0.0092 & 0 & 0 & 0 & 0 \\ 0.0028 & -0.0092 & 0 & 0 & 0 \\ 0.0125 & 0.0028 & -0.0092 & 0 & 0 \\ 0.0195 & 0.0125 & 0.0028 & -0.0092 & 0 \end{bmatrix}.$$

One can verify that $\text{rank}\left(\begin{bmatrix}\mathcal{O}_n & \mathcal{J}_n\end{bmatrix}\right) - \text{rank}(\mathcal{J}_n) = 1$, and thus the system is not strongly observable.

However, suppose that we also have a sensor that measures the velocity of the aircraft (in addition to the pitch rate). The $\mathbf{C}$ matrix would then become

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

and one can verify that the system is strongly observable in this case. Specifically, one can recover the fault input $\mathbf{f}[k]$ from the output of the system $\mathbf{y}[k : k+n]$ without knowing the initial state of the system.

2.6.5 System Properties and Similarity Transformations

We will now see how the properties of a given system are affected by performing a similarity transformation. Specifically, suppose we start with a particular system (2.6) and we perform a similarity transformation $\bar{\mathbf{x}} = \mathbf{T}\mathbf{x}$ to obtain a new system

$$\begin{aligned} \bar{\mathbf{x}}[k+1] &= \bar{\mathbf{A}}\bar{\mathbf{x}}[k] + \bar{\mathbf{B}}\mathbf{u}[k] \\ \mathbf{y}[k] &= \bar{\mathbf{C}}\bar{\mathbf{x}}[k] + \mathbf{D}\mathbf{u}[k] \ , \end{aligned}$$

where $\bar{\mathbf{A}} = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}$, $\bar{\mathbf{B}} = \mathbf{T}\mathbf{B}$, and $\bar{\mathbf{C}} = \mathbf{C}\mathbf{T}^{-1}$. The controllability matrix for this new realization is

$$\begin{aligned} \bar{\mathcal{C}} &= \begin{bmatrix} \bar{\mathbf{B}} & \bar{\mathbf{A}}\bar{\mathbf{B}} & \cdots & \bar{\mathbf{A}}^{n-1}\bar{\mathbf{B}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{T}\mathbf{B} & \mathbf{T}\mathbf{A}\mathbf{T}^{-1}\mathbf{T}\mathbf{B} & \cdots & (\mathbf{T}\mathbf{A}\mathbf{T})^{n-1}\mathbf{T}\mathbf{B} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{T}\mathbf{B} & \mathbf{T}\mathbf{A}\mathbf{B} & \cdots & \mathbf{T}\mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \\ &= \mathbf{T}\begin{bmatrix} \mathbf{B} & \mathbf{A}\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \\ &= \mathbf{T}\mathcal{C} \ . \end{aligned}$$

Thus, the controllability matrix for the new realization is just $\mathbf{T}$ times the controllability matrix for the original realization. Recall that if $\mathbf{M}$ is a matrix and $\mathbf{T}$ is invertible, then the rank of $\mathbf{T}\mathbf{M}$ is the same as the rank of $\mathbf{M}$ (in general, this is only true if $\mathbf{T}$ is invertible). This means that the rank of the controllability matrix for the new realization is the same as the rank of the controllability matrix for the original realization. Similarly, one can show that the rank of the observability and invertibility matrices are also unchanged. This brings us to the following result.

> Performing a similarity transformation does not change the controllability, observability, invertibility or strong observability of the system. In particular, the realization obtained from a similarity transformation is controllable/observable/invertible/strongly observable if and only if the original realization is controllable/observable/invertible/strongly observable.

Kalman Canonical Forms

Since similarity transformations do not change the properties of system, they are quite useful for analyzing system behavior. One such transformation is used to put the system into *Kalman controllability canonical form*. We will derive this transformation and form here.

First, consider the controllability matrix $\mathcal{C}_{n-1}$ for a given pair $(\mathbf{A}, \mathbf{B})$. Suppose that the system is not controllable, so that $\text{rank}(\mathcal{C}_{n-1}) = r < n$. Let $\mathbf{R}$ be an $n \times r$ matrix whose columns form a basis for the range space of $\mathcal{C}_{n-1}$ (i.e., these columns can be any set of r linearly independent columns from the controllability matrix). Define the square matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \bar{\mathbf{R}} \end{bmatrix},$$

where the $n \times (n-r)$ matrix $\bar{\mathbf{R}}$ is chosen so that $\mathbf{T}$ is invertible. Now, note that because the matrix $\mathbf{B}$ is contained in $\mathcal{C}_{n-1}$ and since all columns in the controllability matrix can be written as a linear combination of the columns in $\mathbf{R}$, we have

$$\mathbf{B} = \mathbf{R}\mathbf{B}_c = \mathbf{T}\begin{bmatrix}\mathbf{B}_c \\ \mathbf{0}\end{bmatrix}.$$

for some matrix $\mathbf{B}_c$. Similarly, recall from the derivation of the controllability index in Section 2.6.1 that $\mathbf{A}^n\mathbf{B}$ does not add any extra linearly independent columns to the controllability matrix, and so the range space of $\mathbf{A}^n\mathcal{C}_{n-1}$ is the same as the range space of $\mathcal{C}_{n-1}$. This means that $\mathbf{A}\mathbf{R} = \mathbf{R}\mathbf{A}_c$ for some matrix $\mathbf{A}_c$ (since the columns of $\mathbf{R}$ form a basis for the range space of $\mathcal{C}_{n-1}$). Using these facts, we see that

$$\begin{aligned}
\bar{\mathbf{A}} \triangleq \mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \mathbf{T}^{-1}\mathbf{A}\begin{bmatrix}\mathbf{R} & \bar{\mathbf{R}}\end{bmatrix} &= \mathbf{T}^{-1}\begin{bmatrix}\mathbf{A}\mathbf{R} & \mathbf{A}\bar{\mathbf{R}}\end{bmatrix} \\
&= \mathbf{T}^{-1}\begin{bmatrix}\mathbf{R}\mathbf{A}_c & \mathbf{A}\bar{\mathbf{R}}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{T}^{-1}\mathbf{R}\mathbf{A}_c & \mathbf{T}^{-1}\mathbf{A}\bar{\mathbf{R}}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{A}_c & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{\bar{c}}\end{bmatrix}, \\
\bar{\mathbf{B}} \triangleq \mathbf{T}^{-1}\mathbf{B} = \mathbf{T}^{-1}\mathbf{T}\begin{bmatrix}\mathbf{B}_c \\ \mathbf{0}\end{bmatrix} &= \begin{bmatrix}\mathbf{B}_c \\ \mathbf{0}\end{bmatrix},
\end{aligned}$$

where we used the fact that $\mathbf{T}^{-1}\mathbf{R} = \mathbf{T}^{-1}\mathbf{T}\left[\begin{smallmatrix}\mathbf{I}_r \\ \mathbf{0}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{I}_r \\ \mathbf{0}\end{smallmatrix}\right]$, and defined $\mathbf{A}_{12}$ and $\mathbf{A}_{\bar{c}}$ to be the top r and bottom $n-r$ rows of $\mathbf{T}^{-1}\mathbf{A}\bar{\mathbf{R}}$, respectively. It is easy to verify that the controllability matrix for the pair $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$ is given by

$$\bar{\mathcal{C}}_{n-1} = \begin{bmatrix}\mathbf{A}_c^{n-1}\mathbf{B}_c & \mathbf{A}_c^{n-2}\mathbf{B}_c & \cdots & \mathbf{A}_c\mathbf{B}_c & \mathbf{B}_c \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0}\end{bmatrix}.$$

Note that this is just the controllability matrix for the pair $(\mathbf{A}_c, \mathbf{B}_c)$ with some additional rows of zeros, and since $\text{rank}(\bar{\mathcal{C}}_{n-1}) = \text{rank}(\mathcal{C}_{n-1}) = r$, we see that this pair is controllable. Thus, the Kalman controllable canonical form for a given pair $(\mathbf{A}, \mathbf{B})$ is obtained by the pair

$$\bar{\mathbf{A}} = \begin{bmatrix}\mathbf{A}_c & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{\bar{c}}\end{bmatrix}, \ \bar{\mathbf{B}} = \begin{bmatrix}\mathbf{B}_c \\ \mathbf{0}\end{bmatrix}, \tag{2.18}$$

where the pair $(\mathbf{A}_c, \mathbf{B}_c)$ is controllable. Note that if the original pair $(\mathbf{A}, \mathbf{B})$ is controllable to begin with, we have $\mathbf{A}_c = \mathbf{A}$ and $\mathbf{B}_c = \mathbf{B}$ in the above form.

One can also perform a similarity transformation using the *observability matrix* instead of the controllability matrix, and obtain the *Kalman observability canonical form*

$$\bar{\mathbf{A}} = \begin{bmatrix}\mathbf{A}_o & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{A}_{\bar{o}}\end{bmatrix}, \ \bar{\mathbf{C}} = \begin{bmatrix}\mathbf{C}_o & \mathbf{0}\end{bmatrix}, \tag{2.19}$$

where the pair $(\mathbf{A}_o, \mathbf{C}_o)$ is observable. The details are similar to the derivation of the Kalman controllability canonical form, and are left as an exercise.

2.7 State-Feedback Control

We have seen how to model systems in state-space form, and to check for properties of the state-space realization. We will now see what this means for state-space control design.

It is again instructive to consider the simple scalar plant $x[k+1] = \alpha x[k] + \beta u[k]$, where $\alpha, \beta \in \mathbb{R}$, and $\beta \neq 0$. Recall from Section 2.5 that this system is stable (with $u[k] = 0$) if and only if $|\alpha| < 1$. On the other hand, if $|\alpha| > 1$, perhaps one can use the input $u[k]$ in order to prevent the system from going unstable. Specifically, suppose that we use *state-feedback* and apply $u[k] = -Kx[k]$ at each time-step, for some scalar K. The closed loop system is then $x[k+1] = (\alpha - \beta K)x[k]$, which is stable if and only if $|\alpha - \beta K| < 1$. Clearly, one can satisfy this condition with an appropriate choice of K as long as $\beta \neq 0$. In fact, choosing $K = \frac{\alpha}{\beta}$ would produce $x[k+1] = 0$, meaning that we get stability after just one time-step!

To generalize this, suppose that we have a plant with state-space model (2.6). For now, suppose that we have access to the entire state vector $\mathbf{x}[k]$ – this is not a realistic assumption in practice, because we only have access to the output vector $\mathbf{y}[k]$ (which measures a subset of the states), but let us just assume access to the full state for now. We would like use these states to construct a feedback input so that we can place the closed loop eigenvalues of the system at certain (stable) locations. We will focus on *linear state feedback* of the form

$$\mathbf{u}[k] = -\mathbf{K_1}x_1[k] - \mathbf{K_2}x_2[k] - \cdots - \mathbf{K_n}x_n[k] = -\underbrace{\begin{bmatrix}\mathbf{K}_1 & \mathbf{K}_2 & \cdots & \mathbf{K}_n\end{bmatrix}}_{\mathbf{K}}\mathbf{x}[k] \ .$$

Note that $\mathbf{u}[k]$ is a vector, in general. With this input, the closed loop state-space model becomes

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] = (\mathbf{A} - \mathbf{B}\mathbf{K})\,\mathbf{x}[k] \\ \mathbf{y}[k] &= (\mathbf{C} - \mathbf{D}\mathbf{K})\,\mathbf{x}[k] \ .\end{aligned}$$

The stability of this closed loop system is characterized by the eigenvalues of the matrix $\mathbf{A} - \mathbf{B}\mathbf{K}$, and so the idea is to choose the feedback matrix $\mathbf{K}$ so that those eigenvalues are inside the unit circle (i.e., have magnitude less than 1). The following result shows when this is possible.

> It is possible to arbitrarily place the closed loop eigenvalues via state feedback of the form $\mathbf{u}[k] = -\mathbf{K}\mathbf{x}[k]$ if and only if the pair $(\mathbf{A}, \mathbf{B})$ is controllable.

The proof of necessity can be obtained by appealing to the Kalman Controllable Canonical form; the proof of sufficiency is more complicated. The above result is quite important, and we will make use of it several times, but we will omit the proof of the result here. See [95] for details.

It may be possible to find $\mathbf{K}$ such that $\mathbf{A} - \mathbf{BK}$ is stable even if the pair $(\mathbf{A}, \mathbf{B})$ is not controllable; for example, consider the case where $\mathbf{A}$ is stable, and $\mathbf{B}$ is the zero matrix.

> If there is a matrix $\mathbf{K}$ such that the eigenvalues of $\mathbf{A} - \mathbf{BK}$ have magnitude less than 1, the system is said to be *stabilizable*.

Note that if a system is stabilizable but not controllable, there are some eigenvalues that cannot be placed at arbitrary locations.

Remark 2.2. If the system is controllable, the MATLAB commands `place` and `acker` can be used to find the matrix $\mathbf{K}$ such that the eigenvalues of $\mathbf{A} - \mathbf{BK}$ are at desired locations.

2.8 State Estimators and Observer Feedback

Consider again the plant (2.6). We have seen that if this realization is controllable, we can arbitrarily place the closed loop eigenvalues via state feedback of the form $\mathbf{u}[k] = -\mathbf{Kx}[k]$. However, there is one problem: it assumes that we have access to the entire state vector $\mathbf{x}[k]$. This is typically not the case in practice, since we only have access to the output $\mathbf{y}[k]$, which represents sensor measurements of only a few of the states. Measuring all of the states via sensors is usually not possible, since sensors can be expensive, and some states simply cannot be measured (for example, the state might represent the temperature inside an extremely hot reactor, where it is not possible to place a sensor without damaging it). How can we place the closed loop eigenvalues if we do not have access to the entire state?

The commonly used method to get around this problem is to construct an *estimator* for the state based on the output $\mathbf{y}[k]$. Specifically, the output measures some of the state variables, which are affected by the states that we do not measure. So by examining how the measured states change with time, we can potentially determine the values of the unmeasured states as well. We will do this by constructing a *state estimator* (also called a *state observer*). As one can imagine, the ability to construct such an estimator will be closely tied to the concept of *observability* that we discussed in Section 2.6.2. We will then use the state estimate $\hat{\mathbf{x}}[k]$ provided by the observer to control the system. This is called *observer feedback* and the feedback loop will look like this:

For now, we allow the observer to have access to the input $\mathbf{u}[k]$; later in the course, we will see how to build state estimators when some of the inputs to the system are unknown. Once the observer is constructed, the observer feedback input to the system is given by

$$\mathbf{u}[k] = -\mathbf{K}\hat{\mathbf{x}}[k] \ ,$$

where $\mathbf{K}$ is the same gain matrix that we would use if we had access to the actual system state (i.e., if we were using state feedback $\mathbf{u}[k] = -\mathbf{K}\mathbf{x}[k]$).

2.8.1 State Estimator Design

To see how we can obtain an estimate of the entire state, suppose that we construct a new system with state $\mathbf{z}[k]$ that *mimics* the behavior of the plant:

$$\hat{\mathbf{x}}[k+1] = \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] \ .$$

If we initialize this system with $\hat{\mathbf{x}}[0] = \mathbf{x}[0]$ and we apply the same input $\mathbf{u}[k]$ to this system and the plant, we would have $\hat{\mathbf{x}}[k] = \mathbf{x}[k]$ for all time. Thus, we would have a perfect estimate of the state for all time, and we could use the state feedback control $\mathbf{u}[k] = -\mathbf{K}\hat{\mathbf{x}}[k]$, where $\mathbf{K}$ is the control gain required to place the eigenvalues at desired locations. In summary, if we knew the initial state $\mathbf{x}[0]$ of the system, we could technically obtain an estimate of the state at any time. However, there are some problems with this:

- We may not know the initial state of the system (especially if we cannot measure some of the states of the system).
- The above observer does not make use of any measurements of the states, and thus it has no way of correcting itself if the estimated states start diverging from the actual states (e.g., due to noise or disturbances in the system).

In order to fix these shortcomings, we will modify the observer equation as follows:

$$\hat{\mathbf{x}}[k+1] = \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]) \ . \tag{2.20}$$

In this modified observer, the role of the *corrective term* $\mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k])$ is to utilize the measurements of the state vector in order to help the observer do a good job of tracking the state. Specifically, since $\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k]$, the term $\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]$ represents the error between the measured states and the estimates of those states. If $\hat{\mathbf{x}}[k] = \mathbf{x}[k]$ (i.e., the state observer is perfectly synchronized with the state), then the term $\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]$ will be zero. If the state estimate is different from the actual state, however, the hope is that the term $\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]$ will also be nonzero, and help to reduce the estimation error to zero. The gain matrix $\mathbf{L}$ is used to ensure that this will happen.

To see how to choose $\mathbf{L}$, let us examine the *estimation error* defined as $\mathbf{e}[k] = \mathbf{x}[k] - \hat{\mathbf{x}}[k]$. The evolution of the estimation error is given by

$$\begin{aligned}\mathbf{e}[k+1] &= \mathbf{x}[k+1] - \hat{\mathbf{x}}[k+1] \\ &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] - \mathbf{A}\hat{\mathbf{x}}[k] - \mathbf{B}\mathbf{u}[k] - \mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]) \\ &= \mathbf{A}(\mathbf{x}[k] - \hat{\mathbf{x}}[k]) - \mathbf{L}(\mathbf{C}\mathbf{x}[k] - \mathbf{C}\hat{\mathbf{x}}[k]) \\ &= \mathbf{A}\mathbf{e}[k] - \mathbf{L}\mathbf{C}\mathbf{e}[k] \\ &= (\mathbf{A} - \mathbf{L}\mathbf{C})\mathbf{e}[k] \ .\end{aligned}$$

This is simply an autonomous linear system, and if we would like the estimation error to go to zero regardless of the initial estimation error, we have to choose the matrix $\mathbf{L}$ so that the eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}$ all have magnitude less than 1.

Condition For Placing Eigenvalues of A – LC: Observability

To determine conditions on $\mathbf{A}$ and $\mathbf{C}$ which will allow us to arbitrarily place the eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}$, let us make a connection to controllability. Recall that if the pair ($\mathbf{A}$, $\mathbf{B}$) is controllable, then it is possible to place the eigenvalues of $\mathbf{A} - \mathbf{B}\mathbf{K}$ arbitrarily via a choice of matrix $\mathbf{K}$. For the observer, we are dealing with the matrix $\mathbf{A} - \mathbf{L}\mathbf{C}$; this is different from $\mathbf{A} - \mathbf{B}\mathbf{K}$ because the gain matrix $\mathbf{L}$ pre-multiplies the matrix $\mathbf{C}$, whereas the gain matrix $\mathbf{K}$ post-multiplies the matrix $\mathbf{B}$. However, note that the eigenvalues of a matrix are the same as the eigenvalues of the transpose of the matrix. This means that the eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}$ are the same as the eigenvalues of the matrix $\mathbf{A}' - \mathbf{C}'\mathbf{L}'$, and this matrix has the same form as $\mathbf{A} - \mathbf{B}\mathbf{K}$. Based on our discussion of controllability, we know that if the pair $(\mathbf{A}', \mathbf{C}')$ is controllable, then we can choose $\mathbf{L}$ to place the eigenvalues of $\mathbf{A}' - \mathbf{C}'\mathbf{L}'$ (and thus $\mathbf{A} - \mathbf{L}\mathbf{C}$) at arbitrary locations. Recall that the pair $(\mathbf{A}', \mathbf{C}')$ is controllable if and only if the pair $(\mathbf{A}, \mathbf{C})$ is observable, which brings us to the following result.

> The eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}$ can be placed arbitrarily if and only if the pair $(\mathbf{A}, \mathbf{C})$ is observable. If there exists a matrix $\mathbf{L}$ such that $\mathbf{A} - \mathbf{L}\mathbf{C}$ is stable, the pair $(\mathbf{A}, \mathbf{C})$ is said to be *detectable*.

2.8.2 The Separation Principle

In the last section, we designed the observer (2.20) and used the observer feedback input $\mathbf{u}[k] = -\mathbf{K}\hat{\mathbf{x}}[k]$ where $\mathbf{K}$ was chosen to place the eigenvalues of $\mathbf{A} - \mathbf{B}\mathbf{K}$ at desired locations. Note that we chose the gain $\mathbf{K}$ *independently* of the observer – we pretended that we were just using full state feedback, instead of an estimate of the state. To make

sure that this is valid, let us examine the equations for the entire closed loop system:

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ &= \mathbf{A}\mathbf{x}[k] - \mathbf{B}\mathbf{K}\hat{\mathbf{x}}[k] \\ \hat{\mathbf{x}}[k+1] &= \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}} - \mathbf{D}\mathbf{u}[k]) \\ &= (\mathbf{A} - \mathbf{B}\mathbf{K} - \mathbf{L}\mathbf{C})\hat{\mathbf{x}}[k] + \mathbf{L}\mathbf{C}\mathbf{x}[k] \ .\end{aligned}$$

The closed loop system therefore has $2n$ states (n states corresponding to the plant, and n states corresponding to the observer). In matrix-vector form, this is

$$\begin{bmatrix}\mathbf{x}[k+1] \\ \hat{\mathbf{x}}[k+1]\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{A} & -\mathbf{B}\mathbf{K} \\ \mathbf{L}\mathbf{C} & \mathbf{A} - \mathbf{B}\mathbf{K} - \mathbf{L}\mathbf{C}\end{bmatrix}}_{\mathbf{A}_{cl}} \begin{bmatrix}\mathbf{x}[k] \\ \hat{\mathbf{x}}[k]\end{bmatrix} \ .$$

The closed loop poles are given by the eigenvalues of the matrix $\mathbf{A}_{cl}$. It is hard to determine the eigenvalues of $\mathbf{A}_{cl}$ from its current form, but recall that the eigenvalues of a matrix are unchanged if we perform a similarity transformation on it. Let us define the similarity transformation matrix

$$\mathbf{T} = \begin{bmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{I} & -\mathbf{I}\end{bmatrix} \ ,$$

which has the property that $\mathbf{T} = \mathbf{T}^{-1}$. We then have

$$\begin{aligned}\text{eig}(\mathbf{A}_{cl}) &= \text{eig}\left(\mathbf{T}\mathbf{A}_{cl}\mathbf{T}^{-1}\right) \\ &= \text{eig}\left(\begin{bmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{I} & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{A} & -\mathbf{B}\mathbf{K} \\ \mathbf{L}\mathbf{C} & \mathbf{A} - \mathbf{B}\mathbf{K} - \mathbf{L}\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{I} & -\mathbf{I}\end{bmatrix}\right) \\ &= \text{eig}\left(\begin{bmatrix}\mathbf{A} - \mathbf{B}\mathbf{K} & \mathbf{B}\mathbf{K} \\ \mathbf{0} & \mathbf{A} - \mathbf{L}\mathbf{C}\end{bmatrix}\right) \ .\end{aligned}$$

Now, we use the fact that the eigenvalues of a matrix of the form $\begin{bmatrix}\mathbf{M}_1 & \mathbf{M}_2 \\ 0 & \mathbf{M}_3\end{bmatrix}$ (where $\mathbf{M}_1$ and $\mathbf{M}_3$ are square matrices) are just the eigenvalues of matrix $\mathbf{M}_1$ together with the eigenvalues of matrix $\mathbf{M}_3$. This means that the eigenvalues of $\mathbf{A}_{cl}$ are simply the eigenvalues of $\mathbf{A} - \mathbf{B}\mathbf{K}$ together with the eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}$. In other words, the closed loop eigenvalues are the state feedback eigenvalues along with the eigenvalues of the observer – this shows that we can, in fact, design the feedback gain $\mathbf{K}$ independently of the observer.

2.9 Matrix Pencil Characterizations of System Properties

While the properties of controllability, observability, invertibility and strong observability were characterized by directly examining the controllability, observability and invertibility matrices, there are alternative characterizations that are also possible. We will discuss one such characterization here that has the appealing feature of not depending on powers of the matrix $\mathbf{A}$ (which appears in all of the matrices described above).

2.9.1 Controllability and Observability

Theorem 2.6. *Consider the system* (2.6) *with* $\mathbf{x}[k] \in \mathbb{R}^n$, $\mathbf{u}[k] \in \mathbb{R}^m$ *and* $\mathbf{y}[k] \in \mathbb{R}^p$.

1. *The pair* $(\mathbf{A}, \mathbf{B})$ *is controllable (stabilizable) if and only if* $\text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) = n$ *for all* $z \in \mathbb{C}$ *(*$|z| \geq 1$*).*
2. *The pair* $(\mathbf{A}, \mathbf{C})$ *is observable (detectable) if and only if* $\text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n \\ \mathbf{C}\end{bmatrix}\right) = n$ *for all* $z \in \mathbb{C}$*, (*$|z| \geq 1$*).*

Proof. We will prove the controllability result; the observability result can be obtained in a similar manner.

(Sufficiency:) First, we will prove that

$$\text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) = n \ \forall \ z \in \mathbb{C} \Rightarrow \text{rank}(\mathcal{C}_{n-1}) = n.$$

Recall from Section 2.6.5 that if the rank of the controllability matrix is $r < n$ (i.e., it is not controllable), there exists a matrix $\mathbf{T}$ (constructed from r linearly independent columns from the controllability matrix) such that

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \begin{bmatrix}\mathbf{A}_c & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{\bar{c}}\end{bmatrix}, \ \mathbf{T}^{-1}\mathbf{B} = \begin{bmatrix}\mathbf{B}_c \\ \mathbf{0}\end{bmatrix},$$

where $\mathbf{A}_c$ is an $r \times r$ matrix and $\mathbf{A}_{\bar{c}}$ is an $(n-r) \times (n-r)$ matrix. Next, note that multiplying a matrix on the left and right by invertible matrices does not change the rank of the matrix. Thus,

$$\begin{aligned}\text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) &= \text{rank}\left(\mathbf{T}^{-1}\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}\begin{bmatrix}\mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-r}\end{bmatrix}\right) \\ &= \text{rank}\left(\begin{bmatrix}\mathbf{T}^{-1}\mathbf{A}\mathbf{T} - z\mathbf{I}_n & \mathbf{T}^{-1}\mathbf{B}\end{bmatrix}\right) \\ &= \text{rank}\left(\begin{bmatrix}\mathbf{A}_c - z\mathbf{I}_r & \mathbf{A}_{12} & \mathbf{B}_c \\ \mathbf{0} & \mathbf{A}_{\bar{c}} - z\mathbf{I}_{n-r} & \mathbf{0}\end{bmatrix}\right).\end{aligned}$$

This matrix will have rank n if and only if all of its rows are linearly independent (because there are only n rows in the matrix). In particular, this means that all of bottom $n-r$ rows of the matrix must be linearly independent, which is equivalent to saying that all of the rows of the matrix $\mathbf{A}_{\bar{c}} - z\mathbf{I}_{n-r}$ must be linearly independent. However, whenever $n - r > 0$, we can choose z to be an eigenvalue of $\mathbf{A}_{\bar{c}}$, and this would cause these rows to be linearly dependent (since $\det(\mathbf{A}_{\bar{c}} - z\mathbf{I}_{n-r}) = 0$ for any eigenvalue z). Thus, the only way for $\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}$ to have rank n for all z is if $r = n$ (i.e., the system is controllable).

(Necessity:) Next, we will prove that

$$\text{rank}(\mathcal{C}_{n-1}) = n \Rightarrow \text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) = n \ \forall \ z \in \mathbb{C}.$$

We will do this by proving the *contrapositive*, that is,

$$\exists z_0 \in \mathbb{C} \text{ s.t. } \operatorname{rank}\left(\begin{bmatrix}\mathbf{A} - z_0\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) < n \Rightarrow \operatorname{rank}(\mathcal{C}_{n-1}) < n.$$

To do this, note that if $\operatorname{rank}\left(\begin{bmatrix}\mathbf{A} - z_0\mathbf{I}_n & \mathbf{B}\end{bmatrix}\right) < n$, then the rows of this matrix form a linearly dependent set. Thus, there exists a row vector $\mathbf{v}'$ such that

$$\mathbf{v}'\begin{bmatrix}\mathbf{A} - z_0\mathbf{I}_n & \mathbf{B}\end{bmatrix} = \mathbf{0},$$

or equivalently

$$\mathbf{v}'\mathbf{A} = z_0\mathbf{v}', \;\; \mathbf{v}'\mathbf{B} = \mathbf{0}.$$

This means that $\mathbf{v}'$ must be a left eigenvector of $\mathbf{A}$, corresponding to eigenvalue z_0. This means that $\mathbf{v}'\mathbf{A}^k = z_0^k\mathbf{v}'$ for any $k \in \mathbb{N}$, and thus

$$\mathbf{v}'\begin{bmatrix}\mathbf{A}^{n-1}\mathbf{B} & \mathbf{A}^{n-2}\mathbf{B} & \cdots & \mathbf{AB} & \mathbf{B}\end{bmatrix} = \begin{bmatrix}z_0^{n-1}\mathbf{v}'\mathbf{B} & z_0^{n-2}\mathbf{v}'\mathbf{B} & \cdots & z_0\mathbf{v}'\mathbf{B} & \mathbf{v}'\mathbf{B}\end{bmatrix} = \mathbf{0}.$$

Thus the rows of $\mathcal{C}_{n-1}$ are also linearly dependent, which means that

$$\operatorname{rank}(\mathcal{C}_{n-1}) < n.$$

□

The tests for controllability and observability in Theorem 2.6 are called the *Popov-Belevitch-Hautus* tests. Further discussion of these tests can be found in [1, 36].

2.9.2 Invertibility

To characterize alternative conditions for invertibility and strong observability, it will be useful to introduce the following terminology.

Definition 2.7 (Matrix Pencil)**.** For the linear system (2.6), the matrix

$$\mathbf{P}(z) = \begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}$$

is called the *matrix pencil* of the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. The variable z is an element of $\mathbb{C}$.

Theorem 2.7. *Consider the system* (2.6) *with* $\mathbf{x}[k] \in \mathbb{R}^n$, $\mathbf{u}[k] \in \mathbb{R}^m$ *and* $\mathbf{y}[k] \in \mathbb{R}^p$. *The system is invertible if and only if*

$$rank\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right) = n + m$$

for at least one $z \in \mathbb{C}$.

Proof. To prove this result, we will make use of the transfer function of the system. Specifically, recall from Section 2.4.1 that the z-transform of system (2.6) yields the following input-output representation:

$$\mathbf{Y}(z) = \mathbf{C}(z\mathbf{I}_n - \mathbf{A})^{-1}\mathbf{x}[0] + \underbrace{\left(\mathbf{C}(z\mathbf{I}_n - \mathbf{A})^{-1})\mathbf{B} + \mathbf{D}\right)}_{\mathbf{H}(z)}\mathbf{U}(z).$$

The transfer function matrix $\mathbf{H}(z)$ captures how the input to the system is reflected in the output, and thus the system is invertible if and only if the transfer function has rank m (over the field of all rational functions of z). We will now relate the rank of the transfer function matrix to the rank of $\mathbf{P}(z)$, and to do this, we use the following trick from [91]:

$$\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_n & \mathbf{0} \\ \mathbf{C}(\mathbf{A} - z\mathbf{I}_n)^{-1} & \mathbf{I}_p\end{bmatrix}\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{C}(z\mathbf{I}_n - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\end{bmatrix}\begin{bmatrix}\mathbf{I}_n & (\mathbf{A} - z\mathbf{I}_n)^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I}_m\end{bmatrix}.$$

Since the left-most and right-most matrices on the right-hand-side of the above equation are invertible, we have

$$\begin{aligned}\operatorname{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right) &= \operatorname{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{H}(z)\end{bmatrix}\right) \\ &= \operatorname{rank}\left(\mathbf{A} - z\mathbf{I}_n\right) + \operatorname{rank}\left(\mathbf{H}(z)\right).\end{aligned}$$

If the system is invertible, then $\operatorname{rank}\left(\mathbf{H}(z)\right) = m$ over the field of rational functions of z, and this means that it will have rank m for almost any numerical choice of $z \in \mathbb{C}$. Furthermore, $\operatorname{rank}\left(\mathbf{A} - z\mathbf{I}_n\right) = n$ for any z that is not an eigenvalue of $\mathbf{A}$. Thus, if the system is invertible, we have $\operatorname{rank}\left(\mathbf{P}(z)\right) = n + m$ for almost any choice of z. On the other hand, if the system is not invertible, then $\operatorname{rank}(\mathbf{H}(z)) < m$ for every value of z, and thus $\operatorname{rank}(\mathbf{P}(z)) < n + m$ for every value of z. □

2.9.3 Strong Observability

To characterize strong observability of a system, we will assume without loss of generality that $\operatorname{rank}\left[\begin{smallmatrix}\mathbf{B} \\ \mathbf{D}\end{smallmatrix}\right] = m$ (otherwise, we can just use those columns of the matrix that are

linearly independent, and still be able to affect the system with the inputs in the same way).

Definition 2.8 (Invariant Zero)**.** The complex number $z_0 \in \mathbb{C}$ is called an *invariant zero* of the system (2.6) if $\text{rank}\,(\mathbf{P}(z_0)) < n+m$.

Theorem 2.8. *Consider the system* (2.6) *with* $\mathbf{x}[k] \in \mathbb{R}^n$, $\mathbf{u}[k] \in \mathbb{R}^m$ *and* $\mathbf{y}[k] \in \mathbb{R}^p$. *The system is strongly observable if and only if*

$$\text{rank}\left(\begin{bmatrix}\mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right) = n + m$$

for all $z \in \mathbb{C}$ *(i.e., the system has no invariant zeros).*

Proof. (Necessity:) Suppose that $\text{rank}(\mathbf{P}(z_0)) < n + m$ for some $z_0 \in \mathbb{C}$. Then, there exists a vector $\mathbf{v}$ in the null-space of $\mathbf{P}(z_0)$. Denote the top n components of $\mathbf{v}$ by $\mathbf{x}_0$ and the bottom m components by $\mathbf{u}_0$. This means that

$$\begin{bmatrix}\mathbf{A} - z_0\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}\begin{bmatrix}\mathbf{x}_0 \\ \mathbf{u}_0\end{bmatrix} = \mathbf{0} \Leftrightarrow \begin{matrix}\mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{u}_0 = z_0\mathbf{x}_0 \\ \mathbf{C}\mathbf{x}_0 + \mathbf{D}\mathbf{u}_0 = \mathbf{0}\end{matrix} \tag{2.21}$$

This indicates that if the initial state of the system is $\mathbf{x}_0$ and we apply the input $\mathbf{u}[0] = \mathbf{u}_0$, then $\mathbf{x}[1] = z_0\mathbf{x}_0$ and $\mathbf{y}[0] = \mathbf{0}$. Next, multiply (2.21) by z_0 on the right to obtain

$$\begin{bmatrix}\mathbf{A} - z_0\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix}\begin{bmatrix}z_0\mathbf{x}_0 \\ z_0\mathbf{u}_0\end{bmatrix} = \mathbf{0} \Leftrightarrow \begin{matrix}\mathbf{A}(z_0\mathbf{x}_0) + \mathbf{B}(z_0\mathbf{u}_0) = z_0^2\mathbf{x}_0 \\ \mathbf{C}(z_0\mathbf{x}_0) + \mathbf{D}(z_0\mathbf{u}_0) = \mathbf{0}\end{matrix}$$

This means that if we applied the input $\mathbf{u}[1] = z_0\mathbf{u}_0$, the state at time-step 2 would be $\mathbf{x}[2] = z_0^2\mathbf{x}_0$ and $\mathbf{y}[1] = 0$. Continuing in this way, we see that if we apply the input sequence $\mathbf{u}[k] = z_0^k\mathbf{u}_0$ and the initial state of the system is $\mathbf{x}_0$, the state of the system for all time-steps will satisfy $\mathbf{x}[k] = z_0^k\mathbf{x}_0$ and the output of the system will be $\mathbf{y}[k] = \mathbf{0}$. This is the same output that is obtained when the initial state of the system is $\mathbf{x}[0] = \mathbf{0}$ and $\mathbf{u}[k] = \mathbf{0}$ for all k, and so we cannot recover $\mathbf{x}_0$ from the output of the system.

(Sufficiency:) The proof of sufficiency is given in [76], and we will not cover it here. □

The proof of necessity shows that if the system has an invariant zero z_0, there is an initial state and a set of inputs such that the output is zero for all time, but the state gets multiplied by z_0 at each time-step. Now if $|z_0| < 1$, this may not be of major concern, since the state will simply decay to zero. On the other hand, if $|z_0| \geq 1$, the

state will explode (or stay constant), and we would never be able to determine this from the output. Just as we discussed the notions of detectability and stabilizability as relaxations of observability and controllability, respectively, we can also consider the notion of *strong detectability* to capture the less serious of these cases.

Definition 2.9. The linear system (2.6) is *strongly detectable* if $\mathbf{y}[k] = \mathbf{0}$ for all k implies that $\mathbf{x}[k] \to \mathbf{0}$.

In words, the above definition says that any initial state that holds the output equal to zero for all time (and thus cannot be differentiated from the all zero initial state) must eventually decay to zero. The matrix pencil test in Theorem 2.8 can be generalized to accommodate this case as follows.

Theorem 2.9. *The system* (2.6) *is strongly detectable if and only if all invariant zeros have magnitude less than* 1.

From the above theorems, note that a system can be strongly observable, strongly detectable, or invertible only if the number of outputs p is at least as large as the number of inputs m. Otherwise, the matrix pencil $\mathbf{P}(z)$ will have more columns than rows, and the maximum rank could be no greater than $n + p$, which is less than $n + m$.

Chapter 3

Observers for Linear Systems with Unknown Inputs

As discussed in the previous chapters, it is often the case that a dynamic system can be modeled as having unknown inputs (e.g., representing disturbances and faults [71]). The problem of constructing an observer for such systems has received considerable attention over the past few decades, and various methods of realizing both full and reduced-order observers have been presented in the literature (e.g., [13, 35, 41]). In this chapter, we will study this topic and provide a design procedure to construct unknown input observers. This will generalize the state-estimators that we studied in Section 2.8, and will turn out to be quite useful in diagnosing system faults, as we will see later.

3.1 Unknown Input Observer

Consider the discrete-time linear system

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] \ , \end{aligned} \tag{3.1}$$

with state vector $\mathbf{x} \in \mathbb{R}^n$, unknown input $\mathbf{u} \in \mathbb{R}^m$, output $\mathbf{y} \in \mathbb{R}^p$, and system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ of appropriate dimensions. Note that known inputs can be handled in a straightforward manner, and so we omit them for clarity of development. We also assume without loss of generality that the matrix $\left[\begin{smallmatrix}\mathbf{B}\\ \mathbf{D}\end{smallmatrix}\right]$ is full column rank. This assumption can always be enforced by an appropriate transformation and renaming of the unknown input signals.

Recall from (2.11) in Section 2.6.2 that the response of system (3.1) over $L+1$ time-steps is given by

$$\mathbf{y}[k:k+L] = \mathcal{O}_L \mathbf{x}[k] + \mathcal{J}_L \mathbf{u}[k:k+L]. \tag{3.2}$$

The matrices $\mathcal{O}_L$ and $\mathcal{J}_L$ in the above equation can be expressed in a variety of ways. We will be using the following identities in our derivations:

$$\mathcal{O}_L = \begin{bmatrix} \mathbf{C} \\ \mathcal{O}_{L-1}\mathbf{A} \end{bmatrix} \ , \tag{3.3}$$

$$\mathcal{J}_L = \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1} \end{bmatrix} \ , \tag{3.4}$$

where $\mathcal{C}_{L-1}$ is the observability matrix for the pair $(\mathbf{A}, \mathbf{B})$.

We are now ready to proceed with the construction of an observer to estimate the states in $\mathcal{S}$. Consider an observer of the form

$$\hat{\mathbf{x}}[k+1] = \mathbf{E}\hat{\mathbf{x}}[k] + \mathbf{F}\mathbf{y}[k : k+L]. \tag{3.5}$$

> **Definition 3.1.** The system (3.5) is said to be an *unknown input observer* with delay L if $\hat{\mathbf{x}}[k] - \mathbf{x}[k] \to 0$ as $k \to \infty$, regardless of the values of $\mathbf{u}[k]$.

Note that the observer given by (3.5) is of the same form as the observer studied in Section 2.8, except that (i) the input is no longer included in this equation (since it is assumed to be unknown), and (ii), we allow the use of the system outputs up to time-step $k+L$ to estimate the state $\mathbf{x}[k]$. Alternatively, this observer can be viewed as producing an estimate of the state $\mathbf{x}[k-L]$ from the outputs of the system up to time-step k (i.e., it is a *delayed* state-estimator). We will see the utility of allowing a delay in estimation in our derivation.

To choose the observer matrices $\mathbf{E}$ and $\mathbf{F}$, we examine the estimation error

$$\begin{aligned} \mathbf{e}[k+1] &\equiv \hat{\mathbf{x}}[k+1] - \mathbf{x}[k+1] \\ &= \mathbf{E}\hat{\mathbf{x}}[k] + \mathbf{F}\mathbf{y}[k : k+L] - \mathbf{A}\mathbf{x}[k] - \mathbf{B}\mathbf{u}[k] \\ &= \mathbf{E}\mathbf{e}[k] + \mathbf{F}\mathbf{y}[k : k+L] + (\mathbf{E} - \mathbf{A})\,\mathbf{x}[k] - \mathbf{B}\mathbf{u}[k] \ . \end{aligned}$$

Using (3.2), the error can then be expressed as

$$\begin{aligned} \mathbf{e}[k+1] &= \mathbf{E}\mathbf{e}[k] + \mathbf{F}\mathbf{y}[k : k+L] + (\mathbf{E} - \mathbf{A})\,\mathbf{x}[k] - \mathbf{B}\mathbf{u}[k] \\ &= \mathbf{E}\mathbf{e}[k] + (\mathbf{E} - \mathbf{A} + \mathbf{F}\mathcal{O}_L)\,\mathbf{x}[k] + \mathbf{F}\mathcal{J}_L\mathbf{u}[k : k+L] - \mathbf{B}\mathbf{u}[k] \ . \end{aligned}$$

In order to force the error to go to zero, regardless of the values of $\mathbf{x}[k]$ and the inputs, $\mathbf{E}$ must be a stable matrix and the matrix $\mathbf{F}$ must simultaneously satisfy

$$\mathbf{F}\mathcal{J}_L = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \tag{3.6}$$

$$\mathbf{E} = \mathbf{A} - \mathbf{F}\mathcal{O}_L \ . \tag{3.7}$$

The solvability of condition (3.6) is given by the following theorem.

Theorem 3.1. *There exists a matrix* $\mathbf{F}$ *that satisfies* (3.6) *if and only if*

$$rank(\mathcal{J}_L) - rank(\mathcal{J}_{L-1}) = m \ . \tag{3.8}$$

Proof. There exists a F satisfying (3.6) if and only if the matrix $\mathbf{R} \equiv \begin{bmatrix}\mathbf{B} & 0 & \cdots & 0\end{bmatrix}$ is in the space spanned by the rows of $\mathcal{J}_L$. This is equivalent to the condition

$$\text{rank} \begin{bmatrix}\mathbf{R} \\ \mathcal{J}_L\end{bmatrix} = \text{rank}(\mathcal{J}_L) \ .$$

Using (3.4), we get

$$\begin{aligned}\text{rank} \begin{bmatrix}\mathbf{R} \\ \mathcal{J}_L\end{bmatrix} &= \text{rank} \begin{bmatrix}\mathbf{B} & \mathbf{0} \\ \mathbf{D} & \mathbf{0} \\ \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} = \text{rank} \left(\begin{bmatrix}\mathbf{I} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ -\mathcal{O}_{L-1} & 0 & \mathbf{I}\end{bmatrix} \begin{bmatrix}\mathbf{B} & \mathbf{0} \\ \mathbf{D} & \mathbf{0} \\ \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} \right) \\ &= \text{rank} \left(\begin{bmatrix}\mathbf{B} & 0 \\ \mathbf{D} & 0 \\ 0 & \mathcal{J}_{L-1}\end{bmatrix} \right) \ .\end{aligned}$$

By our assumption that the matrix $\left[\begin{smallmatrix}\mathbf{B} \\ \mathbf{D}\end{smallmatrix}\right]$ has full column rank, we get

$$\text{rank} \begin{bmatrix}\mathbf{R} \\ \mathcal{J}_L\end{bmatrix} = m + \text{rank}\mathcal{J}_{L-1} \ ,$$

thereby completing the proof. □

Note that (3.8) is the condition for inversion of the inputs with known initial state and delay L, as discussed in Section 2.6.3. This is a fairly strict condition, and demonstrates the utility of a delayed observer. When designing such an observer, one can start with $L = 0$ and increase L until a value is found that satisfies (3.8). Recall from Theorem 2.4 that an upper bound on L is given by n; in other words, if (3.8) is not satisfied for $L = n$, asymptotic estimation of the states is not possible with an observer of the form (3.5). The use of delayed observers has been studied in [82, 48, 42], which generalize the investigations of zero-delay observers in [35, 41, 89].

If the system is invertible with delay L, we know that there exists a matrix $\mathbf{F}$ satisfying the linear equation (3.6). From Section A.4.2, all solutions to this equation are of the form

$$\mathbf{F} = \mathbf{S} + \mathbf{GN}, \tag{3.9}$$

where $\mathbf{S}$ is any matrix satisfying $\mathbf{S}\mathcal{J}_L = \begin{bmatrix}\mathbf{B} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix}$, $\mathbf{N}$ is any matrix whose rows form a basis for the left nullspace of $\mathcal{J}_L$, and $\mathbf{G}$ is an arbitrary (free) matrix. Substituting this into (3.7), we obtain

$$\begin{aligned}\mathbf{E} &= \mathbf{A} - \mathbf{F}\mathcal{O}_L \\ &= (\mathbf{A} - \mathbf{S}\mathcal{O}_L) - \mathbf{GN}\mathcal{O}_L.\end{aligned} \tag{3.10}$$

Since we require $\mathbf{E}$ to be a stable matrix, the pair $(\mathbf{A} - \mathbf{S}\mathcal{O}_L, \mathbf{N}\mathcal{O}_L)$ must be detectable (i.e., any eigenvalue that cannot be arbitrarily specified must be stable). In this case, we can find a corresponding matrix $\mathbf{G}$ to make $\mathbf{E}$ stable, and thereby obtain the observer matrices $\mathbf{E}$ and $\mathbf{F}$ via equations (3.9) and (3.10), respectively.

As demonstrated by the development so far, the concept of state estimation and system inversion are intimately connected. Indeed, as discussed in [48], once a state observer is constructed, one can readily obtain an estimate of the unknown inputs by first rearranging (3.1) as

$$\begin{bmatrix} \mathbf{x}[k+1] - \mathbf{A}\mathbf{x}[k] \\ \mathbf{y}[k] - \mathbf{C}\mathbf{x}[k] \end{bmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{D} \end{bmatrix} \mathbf{u}[k] \ . \tag{3.11}$$

Since $\left[\begin{smallmatrix} \mathbf{B} \\ \mathbf{D} \end{smallmatrix}\right]$ is assumed to be full column rank, there exists a matrix $\mathbf{H}$ such that

$$\mathbf{H} \begin{bmatrix} \mathbf{B} \\ \mathbf{D} \end{bmatrix} = \mathbf{I}_m \ . \tag{3.12}$$

Left-multiplying (3.11) by $\mathbf{H}$ and replacing $\mathbf{x}[k]$ with the estimated value $\hat{\mathbf{x}}[k]$, we get the estimate of the input to be

$$\hat{\mathbf{u}}[k] = \mathbf{H} \begin{bmatrix} \hat{\mathbf{x}}[k+1] - \mathbf{A}\hat{\mathbf{x}}[k] \\ \mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] \end{bmatrix} \ . \tag{3.13}$$

Since $\hat{\mathbf{x}}[k] - \mathbf{x}[k] \to 0$ as $k \to \infty$, the above estimate will asymptotically approach the true value of the input.

3.2 Design Procedure

We now summarize the design steps to construct a delayed observer for system (3.1).

1. Find the smallest L such that $\text{rank}[\mathcal{J}_L] - \text{rank}\,[\mathcal{J}_{L-1}] = m$. If the condition is not satisfied for $L = n - \text{nullity}\,[\mathbf{D}]$, it is not possible to reconstruct the entire state of the system.
2. Find any matrix $\mathbf{S}$ satisfying $\mathbf{S}\mathcal{J}_L = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$, and let $\mathbf{N}$ be any matrix whose rows form a basis for the left nullspace of $\mathcal{J}_L$.
3. If possible, choose the matrix $\mathbf{G}$ such that the eigenvalues of $\mathbf{E} = (\mathbf{A} - \mathbf{S}\mathcal{O}_L) - \mathbf{G}\mathbf{N}\mathcal{O}_L$ are stable. Set $\mathbf{F} = \mathbf{S} + \mathbf{G}\mathbf{N}$.
4. The final observer is given by equation (3.5). An estimate of the unknown inputs can be obtained from (3.13), where the matrix $\mathbf{H}$ is chosen to satisfy (3.12).

Example 3.1. Consider the system given by the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}, \ \mathbf{D} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \ .$$

We first find the smallest L for which the system is invertible. For $L = 2$ we have

$$\mathcal{J}_2 = \begin{bmatrix} \mathbf{D} & 0 & 0 \\ \mathbf{CB} & \mathbf{D} & 0 \\ \mathbf{CAB} & \mathbf{CB} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

and the first two columns of this matrix are linearly independent of each other and of all other columns (i.e., (3.8) is satisfied). Thus our observer must have a delay of 2 time-steps.

The matrices $\mathbf{S}$ and $\mathbf{N}$ satisfying $\mathbf{S}\mathcal{J}_2 = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \mathbf{0} \end{bmatrix}$ and $\mathbf{N}\mathcal{J}_2 = \mathbf{0}$ are given by

$$\mathbf{S} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$
$$\mathbf{N} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \end{bmatrix}.$$

The pair $(\mathbf{A} - \mathbf{S}\mathcal{O}_L, \mathbf{N}\mathcal{O}_L)$ is detectable, and in fact, the eigenvalues of $\mathbf{A} - \mathbf{S}\mathcal{O}_L$ are already close to zero, and thus we can simply choose $\mathbf{G} = \mathbf{0}$ to make $\mathbf{E}$ stable. This produces

$$\mathbf{E} = \mathbf{A} - \mathbf{S}\mathcal{O}_L = \begin{bmatrix} 0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \\ 1 & -1 & 0 \end{bmatrix},$$
$$\mathbf{F} = \mathbf{S} + \mathbf{GN} = \mathbf{S}.$$

The final observer is given by

$$\hat{\mathbf{x}}[k+1] = \mathbf{E}\hat{\mathbf{x}}[k] + \mathbf{F}\mathbf{y}[k : k+2].$$

To reconstruct the input, we find a matrix $\mathbf{H}$ satisfying

$$\mathbf{H}\begin{bmatrix} \mathbf{B} \\ \mathbf{D} \end{bmatrix} = \mathbf{I}_m \Rightarrow \mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The estimate of the inputs to the system are now given by

$$\hat{\mathbf{u}}[k] = \mathbf{H}\begin{bmatrix} \hat{\mathbf{x}}[k+1] - \mathbf{A}\hat{\mathbf{x}}[k] \\ \mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] \end{bmatrix}.$$

3.3 Relationship to Strong Detectability

In this section, we will show that the system will have an unknown input observer if and only if the system is strongly detectable. Specifically, consider the condition for being

able to obtain a stable $\mathbf{E}$ matrix, given by the detectability of the pair $(\mathbf{A}-\mathbf{S}\mathcal{O}_L, \mathbf{N}\mathcal{O}_L)$ in (3.10). This detectability condition can actually be stated in terms of the original system matrices as shown by the following theorem.

Theorem 3.2. *The rank condition*

$$rank\begin{bmatrix}\mathbf{A}-\mathbf{S}\mathcal{O}_L - z\mathbf{I}_n \\ \mathbf{N}\mathcal{O}_L\end{bmatrix} = n,\ \ \forall z\in\mathbb{C},\ \ |z|\geq 1$$

is satisfied if and only if

$$rank\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix} = n+m,\ \ \forall z\in\mathbb{C},\ \ |z|\geq 1\ .$$

To prove Theorem 3.2, we make use of the following lemma, which is obtained by a simple modification of a lemma from [48].

Lemma 3.1. *For any $L>0$,*

$$rank\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathcal{O}_{L-1}\mathbf{A} & \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} = rank\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{bmatrix} + rank\,(\mathcal{J}_{L-1})\,. \tag{3.14}$$

Proof. To illustrate the proof and to keep the ideas clear, we will consider the case where $L=2$. The same idea is easily extended to prove the Lemma for general L. Note that

$$\begin{aligned}
\text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathcal{O}_1\mathbf{A} & \mathcal{O}_1\mathbf{B} & \mathcal{J}_1\end{bmatrix} &= \text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{CA} & \mathbf{CB} & \mathbf{D} & \mathbf{0} \\ \mathbf{CA}^2 & \mathbf{CAB} & \mathbf{CB} & \mathbf{D}\end{bmatrix} \\
&= \text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ z\mathbf{C} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ z\mathbf{CA} & \mathbf{0} & \mathbf{CB} & \mathbf{D}\end{bmatrix} \\
&= \text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -z\mathbf{D} & \mathbf{D} & \mathbf{0} \\ z\mathbf{CA} & \mathbf{0} & \mathbf{CB} & \mathbf{D}\end{bmatrix} \\
&= \text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ z\mathbf{CA} & z\mathbf{CB} & \mathbf{CB} & \mathbf{D}\end{bmatrix}.
\end{aligned}$$

The second equality is obtained by subtracting $\mathbf{C}$ times the first row from the third row, and $\mathbf{CA}$ times the first row from the fourth row. The third equality is obtained by

subtracting z times the second row from the third row. The fourth equality is obtained by adding z times the third column to the second column. Repeating this sequence, we obtain

$$\begin{aligned}
\text{rank}\begin{bmatrix} \mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ z\mathbf{CA} & z\mathbf{CB} & \mathbf{CB} & \mathbf{D} \end{bmatrix} &= \text{rank}\begin{bmatrix} \mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ z^2\mathbf{C} & \mathbf{0} & \mathbf{CB} & \mathbf{D} \end{bmatrix} \\
&= \text{rank}\begin{bmatrix} \mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & -z^2\mathbf{D} & \mathbf{CB} & \mathbf{D} \end{bmatrix} \\
&= \text{rank}\begin{bmatrix} \mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{CB} & \mathbf{D} \end{bmatrix} \\
&= \text{rank}\begin{bmatrix} \mathbf{A}-z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} + \text{rank}\mathcal{J}_1.
\end{aligned}$$

This concludes the proof. □

We are now in place to prove Theorem 3.2.

Proof. From the definition of $\mathbf{S}$ and $\mathbf{N}$, we note that both of these matrices are in the left nullspace of the matrix $\begin{bmatrix} \mathbf{0} \\ \mathcal{J}_{L-1} \end{bmatrix}$ (i.e., the last Lm columns of $\mathcal{J}_L$). Let $\bar{\mathbf{N}}$ be a matrix whose rows form a basis for the left nullspace of $\mathcal{J}_{L-1}$. We can then write

$$\mathbf{S} = \mathbf{Q}_1 \begin{bmatrix} \mathbf{I}_p & 0 \\ 0 & \bar{\mathbf{N}} \end{bmatrix}, \ \mathbf{N} = \mathbf{Q}_2 \begin{bmatrix} \mathbf{I}_p & 0 \\ 0 & \bar{\mathbf{N}} \end{bmatrix}$$

for some matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Consider the matrix $\begin{bmatrix} \mathbf{D} \\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B} \end{bmatrix}$. As the system is invertible with delay L, this matrix has rank m. Let $\mathbf{Q}_3$ be any left inverse of this matrix. Since $\mathbf{N}\mathcal{J}_L = \mathbf{0}$, we have that

$$\mathbf{Q}_2 \begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{D} \\ \mathcal{O}_{L-1}\mathbf{B} \end{bmatrix} = \mathbf{0},$$

and thus the rows of $\mathbf{Q}_2$ form a basis for the left nullspace of $\begin{bmatrix} \mathbf{D} \\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B} \end{bmatrix}$. Thus the matrix $\begin{bmatrix} \mathbf{Q}_2 \\ \mathbf{Q}_3 \end{bmatrix}$ is square and invertible. Furthermore, since

$$\mathbf{S}\begin{bmatrix} \mathbf{D} \\ \mathcal{O}_{L-1}\mathbf{B} \end{bmatrix} = \mathbf{Q}_1 \begin{bmatrix} \mathbf{D} \\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B} \end{bmatrix} = \mathbf{B},$$

and thus $\mathbf{Q}_1 = \mathbf{B}\mathbf{Q}_3 + \mathbf{Q}_4\mathbf{Q}_2$ for some arbitrary matrix $\mathbf{Q}_4$. Next, we note that

$$\operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0}\\ \mathbf{C} & \mathbf{D} & \mathbf{0}\\ \mathcal{O}_{L-1}\mathbf{A} & \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} = \operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0}\\ \mathbf{C} & \mathbf{D} & \mathbf{0}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A} & \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B} & \mathbf{0}\\ \mathcal{O}_{L-1}\mathbf{A} & \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix},$$

where we have simply inserted a new set of rows corresponding to $\bar{\mathbf{N}}$ times the third row in the first matrix (which does not introduce any new linearly independent rows, and thus does not change the rank). Now, if we apply the same procedure as in the proof of Lemma 3.1, we can eliminate the last elements in the first two columns to obtain

$$\begin{aligned}\operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0}\\ \mathbf{C} & \mathbf{D} & \mathbf{0}\\ \mathcal{O}_{L-1}\mathbf{A} & \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} &= \operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & \mathbf{0}\\ \mathbf{C} & \mathbf{D} & \mathbf{0}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A} & \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathcal{J}_{L-1}\end{bmatrix}\\ &= \operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A} & \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B}\end{bmatrix} + \operatorname{rank}\left(\mathcal{J}_{L-1}\right).\end{aligned}$$

Using the above definitions of $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3$ and $\mathbf{Q}_4$, we obtain

$$\begin{aligned}\operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A} & \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B}\end{bmatrix} &= \operatorname{rank}\left(\begin{bmatrix}\mathbf{I}_n & 0\\ 0 & \mathbf{Q}_2\\ 0 & \mathbf{Q}_3\end{bmatrix}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A} & \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{B}\end{bmatrix}\right)\\ &= \operatorname{rank}\left(\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B}\\ \mathbf{Q}_2\begin{bmatrix}\mathbf{C}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A}\end{bmatrix} & \mathbf{0}\\ \mathbf{Q}_3\begin{bmatrix}\mathbf{C}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A}\end{bmatrix} & \mathbf{I}_m\end{bmatrix}\right)\\ &= \operatorname{rank}\left(\begin{bmatrix}\mathbf{A}-\mathbf{S}\mathcal{O}_L-z\mathbf{I}_n & \mathbf{0}\\ \mathbf{Q}_2\begin{bmatrix}\mathbf{C}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A}\end{bmatrix} & \mathbf{0}\\ \mathbf{Q}_3\begin{bmatrix}\mathbf{C}\\ \bar{\mathbf{N}}\mathcal{O}_{L-1}\mathbf{A}\end{bmatrix} & \mathbf{I}_m\end{bmatrix}\right)\\ &= \operatorname{rank}\left(\begin{bmatrix}\mathbf{A}-\mathbf{S}\mathcal{O}_L z\mathbf{I}_n & \mathbf{0}\\ \mathbf{N}\mathcal{O}_L & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_m\end{bmatrix}\right).\end{aligned}$$

The second last equation is obtained by subtracting $\mathbf{Q}_4$ times the second row and $\mathbf{B}$ times the last row from the first row. We now have

$$\operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B} & 0\\ \mathbf{C} & \mathbf{D} & 0\\ \mathcal{O}_{L-1}\mathbf{A} & \mathcal{O}_{L-1}\mathbf{B} & \mathcal{J}_{L-1}\end{bmatrix} = m + \operatorname{rank}\left(\mathcal{J}_{L-1}\right) + \operatorname{rank}\begin{bmatrix}\mathbf{A}-\mathbf{S}\mathcal{O}_L-z\mathbf{I}_n\\ \mathbf{N}\mathcal{O}_L\end{bmatrix}.$$

Using Lemma 3.1, we get

$$\operatorname{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I}_n & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix} = m + \operatorname{rank}\begin{bmatrix}\mathbf{A}-\mathbf{S}\mathcal{O}_L - z\mathbf{I}_n\\ \mathbf{N}\mathcal{O}_L\end{bmatrix} \ ,$$

and thus the matrix on the left has rank $n+m$ for all $z \in \mathbb{C}, |z| \geq 1$ if and only if the matrix on the right has rank n for all such z. □

From Theorem 3.1, Theorem 3.2, and our discussion so far, we immediately obtain the following theorem.

Theorem 3.3. *The system* (3.1) *has an observer with delay L if and only if*

1. $\mathit{rank}(\mathcal{J}_L) - \mathit{rank}(\mathcal{J}_{L-1}) = m$,
2. $\mathit{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix} = n+m, \ \forall z \in \mathbb{C}, \ |z| \geq 1.$

Recall that the first condition in the above theorem means that the system is invertible with delay $L+1$. However, the second condition was shown in Theorem 2.7 of Section 2.9.2 (and [60]) to be sufficient for the existence of an inverse. This leads to the following theorem.

Theorem 3.4. *The system $\mathcal{S}$ in* (3.1) *has an observer (possibly with delay) if and only if*

$$\mathit{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix} = n+m, \ \forall z \in \mathbb{C}, \ |z| \geq 1 \ .$$

From Section 2.9.3, we see that this is the condition for the system to be *strongly detectable*, and thus strong detectability and the ability to construct an unknown input observer are equivalent.

Chapter 4

Fault-Detection and Isolation Schemes

In the last chapter, we saw how to design observers for systems with unknown inputs; if the system is strongly detectable, then one can recover the state of the system *and* the unknown inputs (e.g., faults) via such an observer. However, if there are a large number of potential faults in the system, treating them all as unknown inputs may cause the system to no longer be strongly detectable. In such cases, if we assume that only a small number of all potential faults can occur at any given time, one can tailor observer-based schemes to detect and identify such faults. We will examine these schemes here. The material in this section follows Chapters 1-3 of [63], the survey paper [21], and the paper [12].

4.1 Sensor Fault Detection

Consider the system

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k],$$

where $\mathbf{u}$ is a known input. Suppose that the system has p outputs, given by the vector

$$\mathbf{y}[k] = \begin{bmatrix} y_1[k] \\ y_2[k] \\ \vdots \\ y_p[k] \end{bmatrix}.$$

If sensor i is faulty at a given time-step k, the output provided by it can be modeled as $y_i[k] = \mathbf{C}_i\mathbf{x}[k] + f_i[k]$, where $\mathbf{C}_i$ is a row vector and $f_i[k]$ is an additive error. Thus the overall output of the system is given by

$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{f}[k],$$

where $\mathbf{f}[k]$ is the vector of all additive errors at time-step k. Our objective in this section will be to detect and identify any single sensor fault (i.e., determine which, if any, component of $\mathbf{f}[k]$ is nonzero). The architecture that we will be using to achieve this is shown below:

The main tool to perform diagnosis will be a state-estimator (studied in Section 2.8). Specifically, consider the output of any given sensor $y_i[k] = \mathbf{C}_i\mathbf{x}[k] + f_i[k]$, and construct a state-estimator of the form

$$\begin{aligned}\hat{\mathbf{x}}[k+1] &= \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{L}(y_i[k] - \mathbf{C}_i\hat{\mathbf{x}}[k]) \\ &= (\mathbf{A} - \mathbf{L}\mathbf{C}_i)\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{L}y_i[k].\end{aligned}$$

As before, define the estimation error $\mathbf{e}[k] = \mathbf{x}[k] - \hat{\mathbf{x}}[k]$, and examine

$$\mathbf{e}[k+1] = \mathbf{x}[k+1] - \hat{\mathbf{x}}[k+1] = (\mathbf{A} - \mathbf{L}\mathbf{C}_i)\mathbf{e}[k] - \mathbf{L}f_i[k].$$

Note that the estimation error is now affected by the sensor fault $f_i[k]$. Recall that if the pair $(\mathbf{A}, \mathbf{C}_i)$ is observable, then one can choose $\mathbf{L}$ so that the matrix $\mathbf{A} - \mathbf{L}\mathbf{C}_i$ has all eigenvalues at any desired locations. Specifically, one could choose all eigenvalues to be at zero, which would cause $\mathbf{e}[k] \to 0$ in n time-steps if $f_i[k]$ is zero over that time (i.e., the i–th sensor is perfectly reliable over the first n time-steps). Given the estimated state $\hat{\mathbf{x}}[k]$, define the estimated output $\hat{\mathbf{y}}[k] = \mathbf{C}\hat{\mathbf{x}}[k]$, and note that $\mathbf{y}[k] - \hat{\mathbf{y}}[k] \to 0$ if $f_i[k]$ is zero. The j–th component of $\hat{\mathbf{y}}[k]$, denoted by $\hat{y}_j[k] = \mathbf{C}_j\hat{\mathbf{x}}[k]$ gives the estimated output of the j–th sensor.

Define the $p-1$ *residual* signals

$$r_j[k] = y_j[k] - \hat{y}_j[k], \;\; j = 1, 2, \ldots, p, \;\; j \neq i.$$

As noted above, if sensor i is perfect then $\hat{y}_j[k] \to \mathbf{C}_j\mathbf{x}[k]$. However, if sensor j has a fault, then $y_j[k] = \mathbf{C}_j\mathbf{x}[k] + f_j[k]$, and thus $r_j[k] = \mathbf{C}_j\mathbf{x}[k] + f_j[k] - \mathbf{C}_j\hat{\mathbf{x}}[k] \to f_j[k]$. On the other hand, if sensor i itself is faulty, then many (and perhaps all) of the estimated outputs will not match the actual outputs provided by the other (correct) sensors. The detection logic is therefore as follows:

> Suppose that at most one sensor can be faulty at any time-step k. If $r_j[k] \neq 0$ for only one $j \in \{1, 2, \ldots, i-1, i+1, \ldots, p\}$, then sensor j is faulty. If $r_j[k] \neq 0$ for all $j \in \{1, 2, \ldots, i-1, i+1, \ldots, p\}$, then sensor i is faulty.

Example 4.1. Consider the system

$$\mathbf{A} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

We will assume for simplicity that the known inputs are zero (if not they can be handled as in Section 2.8). Consider sensor 1, which provides the output $y_1[k] = \mathbf{C}_1\mathbf{x}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}\mathbf{x}[k]$ under fault-free circumstances. One can verify that the pair $(\mathbf{A}, \mathbf{C}_1)$ is observable, and thus we can use just this output to construct a state estimator of the form

$$\hat{\mathbf{x}}[k+1] = \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{L}(y_i[k] - \mathbf{C}_i\hat{\mathbf{x}}[k]) = (\mathbf{A} - \mathbf{L}\mathbf{C}_i)\hat{\mathbf{x}}[k] + \mathbf{L}y_i[k].$$

The estimation error is given by $\mathbf{e}[k+1] = (\mathbf{A} - \mathbf{L}\mathbf{C}_i)\mathbf{e}[k]$ (when the fault is zero), and we choose the matrix $\mathbf{L}$ so that all eigenvalues of $\mathbf{A} - \mathbf{L}\mathbf{C}_i$ are at zero; this is achieved via

$$\mathbf{L} = \begin{bmatrix} 2 \\ 1.5 \\ 2.5 \\ 2 \end{bmatrix}.$$

With this matrix, we find that $(\mathbf{A} - \mathbf{L}\mathbf{C}_i)^4 = \mathbf{0}$, and thus $\mathbf{e}[4] = (\mathbf{A} - \mathbf{L}\mathbf{C}_i)^4\mathbf{e}[0] = \mathbf{0}$ if there are no faults during the first 5 time-steps.

Now construct the output estimates $\hat{y}_2[k] = \mathbf{C}_2\hat{\mathbf{x}}[k]$ and $\hat{y}_3[k] = \mathbf{C}_3\hat{\mathbf{x}}[k]$, and the residuals

$$r_2[k] = y_2[k] - \hat{y}_2[k], \;\; r_3[k] = y_3[k] - \hat{y}_3[k].$$

If there are no faults in the first five time-steps, we have $r_2[k] = 0$ and $r_3[k] = 0$ after $k = 5$. However, now suppose that there is a fault in sensor 2 starting at time-step 20, with additive error $f_2[k] = 2$. The residual $r_2[k]$ then becomes equal to 2 after time-step 20, while $r_3[k]$ stays zero. Thus, we can determine that sensor 2 is faulty.

4.2 Robust Fault Detection and Identification

We will now consider systems that are affected by faults, together with disturbances. Such systems can be modeled as follows:

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}_d\mathbf{d}[k] + \mathbf{B}_f\mathbf{f}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}_d\mathbf{d}[k] + \mathbf{D}_f\mathbf{f}[k]. \end{aligned} \tag{4.1}$$

where $\mathbf{d} \in \mathbb{R}^{m_d}$ is the disturbance, and $\mathbf{f} \in \mathbb{R}^{m_f}$ is the fault. We have omitted known inputs in the above equation, but they can easily be handled. The values of $\mathbf{d}[k]$ and $\mathbf{f}[k]$ are *unknown* at each time-step; however, $\mathbf{d}[k]$ is assumed to be active at all time, whereas $\mathbf{f}[k]$ is assumed to be intermittently nonzero (corresponding to transient faults), nonzero only after some time, or sparse (corresponding to only a few of the faults being active). Note that the above model is able to capture actuator faults, plant faults, and sensor faults. The output of the system over $L+1$ time-steps is given by

$$\mathbf{y}[k:k+L] = \mathcal{O}_L \mathbf{x}[k] + \mathcal{J}_L^d \mathbf{d}[k:k+L] + \mathcal{J}_L^f \mathbf{f}[k:k+L], \tag{4.2}$$

where $\mathcal{J}_L^d$ and $\mathcal{J}_L^f$ are the invertibility matrices for the systems $(\mathbf{A}, \mathbf{B}_d, \mathbf{C}, \mathbf{D}_d)$ and $(\mathbf{A}, \mathbf{B}_f, \mathbf{C}, \mathbf{D}_f)$, respectively. As described in the last section, if the system

$$\left(\mathbf{A}, \begin{bmatrix}\mathbf{B}_d & \mathbf{B}_f\end{bmatrix}, \mathbf{C}, \begin{bmatrix}\mathbf{D}_d & \mathbf{D}_f\end{bmatrix}\right)$$

is strongly detectable, one can construct an unknown input observer for the system and recover both $\mathbf{f}[k]$ and $\mathbf{d}[k]$, regardless of their values. However, when there are a large number of possible faults, decoupling them all simultaneously will often not be possible. In such cases, one can operate under the assumption that only one (or a few) of the faults will be active. Since the disturbances are assumed to be present all the time (e.g., wind on an airplane), our objective will be to construct an observer to decouple only the disturbances, and then use the estimation error to infer the presence of any faults.

4.2.1 Fault Detection: Single Observer Scheme

Here, we design an unknown input observer to only decouple the disturbances, *assuming* that the faults are zero. When we run the resulting state-estimator, and the fault is nonzero, that will hopefully show up as a difference in the estimated outputs and the actual outputs, from which we can detect the presence of the fault.

To this end, consider the estimator

$$\hat{\mathbf{x}}[k+1] = \mathbf{E}\hat{\mathbf{x}}[k] + \mathbf{F}\mathbf{y}[k:k+L]$$

where $\mathbf{E}$ and $\mathbf{F}$ are chosen so that $\mathbf{e}[k] = \hat{\mathbf{x}}[k] - \mathbf{x}[k] \to 0$, regardless of $\mathbf{d}[k]$, and when $\mathbf{f}[k] = 0$ for all k. This can be done as long as $(\mathbf{A}, \mathbf{B}_d, \mathbf{C}, \mathbf{D}_d)$ is strongly detectable. More specifically,

$$\begin{aligned}
\mathbf{e}[k+1] &\equiv \hat{\mathbf{x}}[k+1] - \mathbf{x}[k+1] \\
&= \mathbf{E}\hat{\mathbf{x}}[k] + \mathbf{F}\mathbf{y}[k:k+L] - \mathbf{A}\mathbf{x}[k] - \mathbf{B}_d\mathbf{d}[k] - \mathbf{B}_f\mathbf{f}[k] \\
&= \mathbf{E}\mathbf{e}[k] + \mathbf{F}\mathbf{y}[k:k+L] + (\mathbf{E}-\mathbf{A})\,\mathbf{x}[k] - \mathbf{B}_d\mathbf{d}[k] - \mathbf{B}_f\mathbf{f}[k] \ .
\end{aligned}$$

Using (4.2), the error can then be expressed as

$$\begin{aligned}
\mathbf{e}[k+1] &= \mathbf{E}\mathbf{e}[k] + \mathbf{F}\mathbf{y}[k:k+L] + (\mathbf{E}-\mathbf{A})\,\mathbf{x}[k] - \mathbf{B}_d\mathbf{d}[k] - \mathbf{B}_f\mathbf{f}[k] \\
&= \mathbf{E}\mathbf{e}[k] + (\mathbf{E}-\mathbf{A}+\mathbf{F}\mathcal{O}_L)\,\mathbf{x}[k] + \mathbf{F}\mathcal{J}_L^d\mathbf{d}[k:k+L] - \mathbf{B}_d\mathbf{d}[k] \\
&\quad + \mathbf{F}\mathcal{J}_L^f\mathbf{f}[k:k+L] - \mathbf{B}_f\mathbf{f}[k] \ .
\end{aligned}$$

As in the last chapter, if the system $(\mathbf{A}, \mathbf{B}_d, \mathbf{C}, \mathbf{D}_d)$ is strongly detectable, we can choose $\mathbf{E}$ and $\mathbf{F}$ so that $\mathbf{F}\mathcal{J}_L^d = \begin{bmatrix}\mathbf{B}_d & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix}$ and $\mathbf{E} = \mathbf{A} - \mathbf{F}\mathcal{O}_L$ is stable. This produces

$$\mathbf{e}[k+1] = (\mathbf{A} - \mathbf{F}\mathcal{O}_L)\mathbf{e}[k] + \mathbf{F}\mathcal{J}_L^f \mathbf{f}[k : k+L] - \mathbf{B}_f \mathbf{f}[k], \tag{4.3}$$

from which $\mathbf{e}[k] \to 0$ if $\mathbf{f}[k] = 0$ for all k. We will now define a residual function

$$\mathbf{r}[k] = \mathbf{H}_1 \hat{\mathbf{x}}[k] + \mathbf{H}_2 \mathbf{y}[k : k+L]$$

that is supposed to satisfy the property that $\mathbf{r}[k] = 0$ if $\mathbf{f}[k] = 0$, and $\mathbf{r}[k] \neq 0$ otherwise. Expanding the above expression, we obtain

$$\mathbf{r}[k] = \mathbf{H}_1\mathbf{e}[k] + \mathbf{H}_1\mathbf{x}[k] + \mathbf{H}_2\mathcal{O}_L\mathbf{x}[k] + \mathbf{H}_2\mathcal{J}_L^d\mathbf{d}[k : k+L] + \mathbf{H}_2\mathcal{J}_L^f\mathbf{f}[k : k : L].$$

When $\mathbf{f}[k] = 0$, we can choose $\mathbf{H}_1 = -\mathbf{H}_2\mathcal{O}_L$, and $\mathbf{H}_2$ to be a matrix whose rows form a basis for the left nullspace of $\mathcal{J}_L^d$, yielding

$$\mathbf{r}[k] = \mathbf{H}_1\mathbf{e}[k] + \mathbf{H}_2\mathcal{J}_L^f\mathbf{f}[k : k : L]. \tag{4.4}$$

Equations (4.3) and (4.4) together form the *residual generator*, and if $\mathbf{f}[k] = 0$, we have $\mathbf{e}[k] \to 0$ and $\mathbf{r}[k] \to 0$. However, if $\mathbf{f}[k]$ becomes nonzero after some time, then we would expect $\mathbf{e}[k]$ and $\mathbf{r}[k]$ to be nonzero, from which we can detect the presence of the fault.

Remark 4.1. Note that this method does not guarantee that all possible faults will be detected. Since the system is assumed to not be strongly detectable (otherwise we could design a UIO for the entire system, including faults), there is an input and a set of faults and disturbances such that the output is zero for all time. Clearly such faults and disturbances cannot be detected; the above method assumes that the faults to the system are not applied in this "worst case" manner, so that the output of the system does in fact indicate the presence of a fault (via the state-estimator and the residual).

Example 4.2. Consider the system

$$\mathbf{x}[k+1] = \underbrace{\frac{1}{2}\begin{bmatrix}1 & 1 & 0 & 1\\ 0 & 1 & 1 & 0\\ 1 & 0 & 1 & 1\\ 0 & 0 & 1 & 1\end{bmatrix}}_{\mathbf{A}}\mathbf{x}[k] + \underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 1\\ 1 & 1\end{bmatrix}}_{\mathbf{B}_f}\mathbf{f}[k] + \underbrace{\begin{bmatrix}0\\ 1\\ 0\\ 1\end{bmatrix}}_{\mathbf{B}_d}\mathbf{d}[k]$$

$$\mathbf{y}[k] = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{bmatrix}\mathbf{x}[k].$$

Using the design procedure in Chapter 3, we construct an unknown input observer for the system $(\mathbf{A}, \mathbf{B}_d, \mathbf{C}, \mathbf{0})$, producing

$$\hat{\mathbf{x}}[k+1] = \underbrace{\begin{bmatrix}0 & 0.3077 & -0.1538 & 0.6923 & -0.3077 & 0.3077\\ 0 & 0.1538 & -0.0769 & -0.1538 & 0.8462 & 0.1538\\ 0 & -0.3077 & 0.1538 & 0.3077 & -0.6923 & 0.6923\\ -0.5 & -0.2308 & -0.3846 & 0.2308 & 0.2308 & 0.7692\end{bmatrix}}_{\mathbf{F}}\mathbf{y}[k : k+1].$$

Note that the $\mathbf{E}$ matrix is zero in this case (we placed all of the eigenvalues at 0). Next, to design the residual (4.4), we choose the matrix $\mathbf{H}_2$ to form a basis for the left nullspace of

$$\mathcal{J}_1^d = \begin{bmatrix} \mathbf{D}_d & 0 \\ \mathbf{C}\mathbf{B}_d & \mathbf{D}_d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix},$$

which gives

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Finally, we have

$$\mathbf{H}_1 = -\mathbf{H}_2\mathcal{O}_1 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 \\ -0.5 & -0.5 & 0 & -0.5 \\ -0.5 & 0 & -0.5 & -0.5 \end{bmatrix}.$$

If there are no faults in the system, the residual $\mathbf{r}[k] = \mathbf{H}_1\hat{\mathbf{x}}[k] + \mathbf{H}_2\mathbf{y}[k : k+1]$ is found to be zero. However, if one or both of the faults become active (say, equal to 2 over some period of time), we find that the residual also becomes nonzero over that time-period, and thus we are able to detect the fault.

4.2.2 Single Fault Identification: Bank of Observers

The method described in the previous section used a single observer to decouple the disturbances, and a nonzero residual was used to infer the presence of a fault. However, the above scheme generally does not allow us to identify which of the possible faults caused the nonzero residual (i.e., which entry of $\mathbf{f}[k]$ is nonzero). To identify faults, we will use a scheme similar to that described at the beginning of the chapter for sensor faults. Specifically, we will design a *bank* of observers, each of which will provide a residual that will allow us to identify a single fault at a time.

First, recall from Section 2.9.3 that a system is strongly detectable only if the number of unknown inputs is less than the number of outputs. Let m_f be the number of possible faults, and let m_d be the number of disturbances. Now we will design m_f different observers as follows. For observer i, we group fault i with the disturbances, and treat it as an unknown input. Specifically, let $\mathbf{B}_f^i$ denote the i–th column of $\mathbf{B}_f$, and let $\mathbf{B}_f^{-i}$ denote all other columns. Similarly, let $f_i[k]$ denote the i–th component of $\mathbf{f}[k]$ and let

$\mathbf{f}_{-i}[k]$ denote all other components. Rearranging (4.1), we can write

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}_f^{-i}\mathbf{f}_{-i}[k] + \begin{bmatrix}\mathbf{B}_d & \mathbf{B}_f^i\end{bmatrix}\begin{bmatrix}\mathbf{d}[k] \\ f_i[k]\end{bmatrix}$$
$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{D}_f^{-i}\mathbf{f}_{-i}[k] + \begin{bmatrix}\mathbf{D}_d & \mathbf{D}_f^i\end{bmatrix}\begin{bmatrix}\mathbf{d}[k] \\ f_i[k]\end{bmatrix}.$$

If the system $(\mathbf{A}, \begin{bmatrix}\mathbf{B}_d & \mathbf{B}_f^i\end{bmatrix}, \mathbf{C}, \begin{bmatrix}\mathbf{D}_d & \mathbf{D}_f^i\end{bmatrix})$ is strongly detectable, we can create an unknown input observer for this system, treating the vector $\begin{bmatrix}\mathbf{d}[k] \\ f_i[k]\end{bmatrix}$ as the unknown inputs. Furthermore, we can design a *residual* $\mathbf{r}_i[k]$ for this system; let the residual generator be given by

$$\hat{\mathbf{x}}^i[k+1] = \mathbf{E}^i\hat{\mathbf{x}}^i[k] + \mathbf{F}^i\mathbf{y}[k:k+L]$$
$$\mathbf{r}_i[k] = \mathbf{H}_{i1}\hat{\mathbf{x}}^i[k] + \mathbf{H}_{i2}\mathbf{y}[k:k+L].$$

As in the last section, if $\mathbf{H}_{i1}$ and $\mathbf{H}_{i2}$ are chosen appropriately, we will have $\mathbf{r}_i[k] = 0$ if $\mathbf{f}_{-i}[k] = 0$, and $\mathbf{r}_i[k] \neq 0$ otherwise. Thus, we can determine which faults are active simply by examining which of the residuals are nonzero. Specifically, residual $\mathbf{r}_i[k]$ depends on all faults except $f_i[k]$. The detection logic is as follows.

> If all residuals except $\mathbf{r}_i[k]$ are nonzero, and $\mathbf{r}_i[k]$ is zero, then fault i has occurred.

Note that only one fault can be identified via the above scheme.

4.2.3 Multiple Fault Identification: Dedicated Observers

If we would like to be able to simultaneously identify multiple faults, we can extend the scheme in the previous section as follows. As in the last section, we will design m_f different observers. For observer i, we group all faults except fault i with the disturbances, and treat them all as unknown inputs. Specifically, let $\mathbf{B}_f^i$ denote the i–th column of $\mathbf{B}_f$, and let $\mathbf{B}_f^{-i}$ denote all other columns. Similarly, let $f_i[k]$ denote the i–th component of $\mathbf{f}[k]$ and let $\mathbf{f}_{-i}[k]$ denote all other components. Rearranging (4.1), we can write

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}_f^i f_i[k] + \begin{bmatrix}\mathbf{B}_d & \mathbf{B}_f^{-i}\end{bmatrix}\begin{bmatrix}\mathbf{d}[k] \\ \mathbf{f}_{-i}[k]\end{bmatrix}$$
$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{D}_f^i f_i[k] + \begin{bmatrix}\mathbf{D}_d & \mathbf{D}_f^{-i}\end{bmatrix}\begin{bmatrix}\mathbf{d}[k] \\ \mathbf{f}_{-i}[k]\end{bmatrix}.$$

If the system $(\mathbf{A}, \begin{bmatrix}\mathbf{B}_d & \mathbf{B}_f^{-i}\end{bmatrix}, \mathbf{C}, \begin{bmatrix}\mathbf{D}_d & \mathbf{D}_f^{-i}\end{bmatrix})$ is strongly detectable, we can create an unknown input observer for this system, treating the vector $\begin{bmatrix}\mathbf{d}[k] \\ \mathbf{f}_{-i}[k]\end{bmatrix}$ as the unknown

inputs. Furthermore, we can design a *residual* $\mathbf{r}_i[k]$ for this system; let the residual generate for the system be given by

$$\begin{aligned}\hat{\mathbf{x}}^i[k+1] &= \mathbf{E}^i\hat{\mathbf{x}}^i[k] + \mathbf{F}^i\mathbf{y}[k:k+L] \\ \mathbf{r}_i[k] &= \mathbf{H}_{i1}\hat{\mathbf{x}}^i[k] + \mathbf{H}_{i2}\mathbf{y}[k:k+L].\end{aligned}$$

As in the last section, if $\mathbf{H}_{i1}$ and $\mathbf{H}_{i2}$ are chosen appropriately, we will have $\mathbf{r}_i[k] = 0$ if $f_i[k] = 0$, and $\mathbf{r}_i[k] \neq 0$ otherwise. Thus, we can determine which faults are active simply by examining which of the residuals are nonzero.

> If residual $\mathbf{r}_i[k]$ is nonzero, then fault i has occurred.

It is instructive to compare the scheme in this section to the previous one. Note that in order to identify multiple faults, we had to group all but one of the faults into the unknown inputs, together with the disturbances. This could potentially add a large number of unknown inputs to the system, preventing it from being strongly detectable. In contrast, the scheme in the previous section only adds a *single* fault to the unknown inputs for each observer. Thus, the former scheme can be applied to a larger class of systems than the scheme described in this section.

4.3 Parity Space Methods

In the last two sections, we used observer-based methods to generate residuals for fault detection and identification. One can also try to diagnose faults directly from the outputs of the system, without constructing an observer (this would be useful, for example, if the system is not strongly detectable). This is called the *parity space method*.

First, note that the output of the system (4.1) over any $L+1$ time-steps is given by (4.2). In this equation, the quantities $\mathbf{x}[k], \mathbf{f}[k:k+L]$ and $\mathbf{d}[k:k+L]$ are all unknown (assuming that an unknown input observer for $\mathbf{x}[k]$ has not been constructed). However, for the purposes of fault diagnosis, we are only interested in recovering information about $\mathbf{f}[k:k+L]$, so we would like to eliminate the effect of the other unknown quantities on the output. To do this, define the matrix $\mathbf{V}$ such that the rows form a basis for the left nullspace of $\begin{bmatrix}\mathcal{O}_L & \mathcal{J}_L^d\end{bmatrix}$.

> **Definition 4.1.** The space spanned by the rows of $\mathbf{V}$ is called the *parity space* of the system.

If we multiply the output of the system by $\mathbf{V}$, we obtain

$$\begin{aligned}\underbrace{\mathbf{V}\mathbf{y}[k:k+L]}_{\mathbf{r}[k]} &= \mathbf{V}\mathcal{O}_L\mathbf{x}[k] + \underbrace{\mathbf{V}\mathcal{J}_L^f}_{\mathbf{P}}\mathbf{f}[k:k+L] + \mathbf{V}\mathcal{J}_L^d\mathbf{d}[k:k+L] \\ &= \mathbf{P}\mathbf{f}[k:k+L].\end{aligned}$$

The quantity $\mathbf{r}[k]$ is the residual, and is used to diagnose the presence of faults. In this context, the residual is also known as a *syndrome* in the coding-theory literature, which we will study later. The residual lies in the range-space of the matrix $\mathbf{P}$, and characterizes the (isolated) effects of the faults on the output over $L+1$ time-steps. If column i of $\mathbf{P}$ is nonzero, then if the i-th element of $\mathbf{f}[k:k+L]$ is nonzero and everything else is zero, the residual will be nonzero, which means that the fault will be detected.

Similarly, if column i is linearly independent of any other single column, then one can identify that the i–th element of $\mathbf{f}[k:k+L]$ is nonzero, as long as only one element of $\mathbf{f}[k:k+L]$ is nonzero. To see this, let $\mathbf{p}_i$ denote the i–th column of $\mathbf{P}$, and let f_i denote the value of the i–th element of $\mathbf{f}[k:k+L]$. If f_i is the only nonzero element of $\mathbf{f}[k:k+L]$, the residual is given by $\mathbf{r}[k] = \mathbf{p}_i f_i$. This means that the residual is a scaled version of $\mathbf{p}_i$. The only way we could not identify that f_i is the nonzero element is if there is some other nonzero element of $\mathbf{f}[k:k+L]$, say f_j, that could produce the same residual. This would mean that

$$\mathbf{r}[k] = \mathbf{p}_i f_i = \mathbf{p}_j f_j \;\; \Rightarrow \mathbf{p}_i f_i - \mathbf{p}_j f_j = 0.$$

This means that column $\mathbf{p}_i$ and $\mathbf{p}_j$ would have be linearly dependent, which is not the case by assumption.

More generally, if a given set of t columns of $\mathbf{P}$ are linearly independent of any other set of t columns, then we could uniquely identify the faults corresponding to those t columns (this can be shown by following an identical argument to the case where $t=1$ considered above).

Example 4.3. Consider the system

$$\begin{aligned}\mathbf{x}[k+1] &= \underbrace{\begin{bmatrix}1&1&0&1\\0&1&1&0\\1&0&1&1\\0&0&1&1\end{bmatrix}}_{\mathbf{A}}\mathbf{x}[k] + \underbrace{\begin{bmatrix}1&0\\0&1\\0&1\\1&1\end{bmatrix}}_{\mathbf{B}_f}\mathbf{f}[k] + \underbrace{\begin{bmatrix}0\\1\\0\\1\end{bmatrix}}_{\mathbf{B}_d}\mathbf{d}[k] \\ \mathbf{y}[k] &= \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&1&1&0\end{bmatrix}\mathbf{x}[k].\end{aligned}$$

Consider the output of the system over 4 time-steps (i.e., $L=3$). A basis $\mathbf{V}$ for the left

nullspace of $\begin{bmatrix}\mathcal{O}_3 & \mathcal{J}_3^d\end{bmatrix}$ is found, and is used to calculate

$$\mathbf{P} = \mathbf{V}\mathcal{J}_L^f = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 2 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Note that the last two columns of $\mathbf{P}$ are zero, so we will not be able to detect or identify any faults in the last two elements of $\mathbf{f}[k : k+3]$, or equivalently, in $\mathbf{f}[3]$. However, all other columns are nonzero, so if any single element of $\mathbf{f}[k : k+2]$ (i.e., all but the bottom two elements) is nonzero, this will be detected.

Next, note that the first column is just the negative of the second column. Thus, if $\mathbf{f}[k]$ is of the form $\begin{bmatrix} a \\ a \end{bmatrix}$ for any a, we would not be able to detect this behavior. Thus we are not guaranteed to be able to detect any *two* faults in $\mathbf{f}[k : k+3]$ (in the worst case). However, note that columns $3, 4, 5$ and 6 are linearly independent of all other columns. Thus, any nonzero entries in the third, fourth, fifth and sixth elements of $\mathbf{f}[k : k+3]$ are guaranteed to be identified (these elements correspond to $\mathbf{f}[k+1]$ and $\mathbf{f}[k+2]$).

For example, suppose that the residual is given by

$$\mathbf{r}[k] = \begin{bmatrix} 0 \\ 3 \\ 0 \\ 6 \\ 6 \end{bmatrix}.$$

This is a multiple of the third column, and thus we know that the third element of $\mathbf{f}[k : k+3]$ is nonzero, with value 3.

For this example, if we assume that there is at most one fault in any $L+1$ time-steps, we can uniquely identify the fault if it occurs in $\mathbf{f}[k+1]$ or $\mathbf{f}[k+2]$.

Chapter 5

Reliable System Design

In the previous sections, we have been concerned with detecting and identifying faults that occur in a given system. We now turn our attention to *constructing* reliable systems. We will consider various fault models, and describe schemes that can be used to handle each of those models. We will start our investigation by considering an intuitive majority voting scheme.

5.1 Majority Voting with Unreliable Components

As described in Chapter 1, majority voting is perhaps the most direct method of implementing fault-tolerance. Specifically, suppose that we have a certain component that could malfunction. In order to obtain reliable behavior, one could simply replicate this component, and then use some logic to compare the outputs to decide which one to use. For example, consider designing a controller for a given system. In Section 2.8, we saw that we could design a state-estimator for the system, and then use observer feedback to control the system. The architecture is as follows:

The controller in this case is given by the state-estimator together with the feedback input. Now, if the controller experiences a fault, one could potentially apply an incorrect (or destabilizing) input to the system. To prevent this, we can replicate the controller three times, and then simply use a majority voter to apply the input that is specified by

the majority of the controllers; this is guaranteed to work if at most one of the controllers is assumed to be faulty at any given time-step. More generally, if we want to tolerate up to f faulty controllers at any given time-step, we should use a bank of $2f+1$ redundant controllers, and apply the value specified by at least $f+1$ of them.

The above procedure works if one considers that a bounded number of faults occur, and that the majority voter itself is reliable. However, what happens if the components are faulty with a certain probability, and the voter itself can choose an incorrect value with a certain probability? This problem was studied in the seminal paper [92] by John Von Neumann, and we will discuss the main ideas here.

The paper considers the case where each component outputs a binary value 0 or 1. However, the component can be faulty with probability η, in which case it outputs a 0 when it should have output a 1, and vice versa. Note that there is a possibility that two or more of the components can be faulty, under this model. Specifically, the probability that at least two components are faulty is given by

$$\theta = \binom{3}{2}\eta^2(1-\eta) + \eta^3 = 3\eta^2 - 2\eta^3. \tag{5.1}$$

If the majority voter is operating correctly, it should output a 1 if at least two of its inputs are 1, and 0 otherwise. However, the voter itself can be faulty with probability ϵ, whereby it outputs a 1 when the majority of its inputs are 0, and vice versa. We will assume throughout that $\eta \leq 1/2$ and $\epsilon \leq 1/2$, because otherwise, the corresponding component is "wrong more often than right", and thus we can simply flip the value that it provides us to obtain a value that is incorrect with probability $1-\eta < 1/2$ and $1-\epsilon < 1/2$.

The output of the entire mechanism (the three redundant components and the majority voter together) can be incorrect with a certain probability, given by

$$\begin{aligned}\eta^* &= \epsilon(1-\theta) + (1-\epsilon)\theta \\ &= \epsilon + \theta(1-2\epsilon).\end{aligned}$$

Since $\epsilon < 1/2$, we see that $1-2\epsilon > 0$, and thus $\eta^* > \epsilon$; this is intuitive, since one cannot expect to obtain a measure of reliability greater than the reliability of the last component in the chain. The above expression shows that the probability of a faulty output exceeds the minimum possible probability ϵ by the quantity $\theta(1-2\epsilon)$. To investigate η^* more carefully, use (5.1) in the above expression to obtain

$$\eta^* = \epsilon + (3\eta^2 - 2\eta^3)(1-2\epsilon).$$

Note that if $\epsilon = \frac{1}{2}$, we have $\eta^* = \epsilon = \frac{1}{2}$, regardless of η. This is intuitive: if the majority voter is completely arbitrary (i.e., it is correct or incorrect with equal probability), the output of the majority voter can never be trusted, regardless of how reliable the individual components themselves are. A plot of η^* versus η for various nonzero values of ϵ is shown in Figure 5.1.

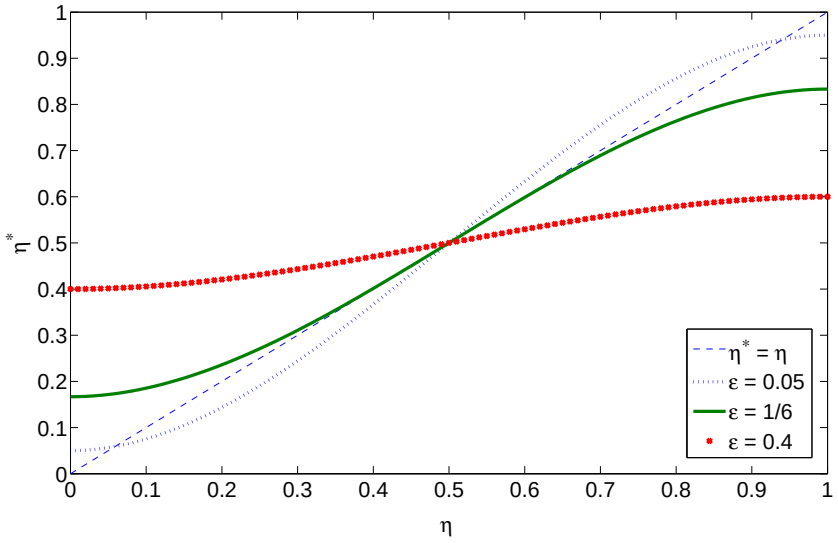

Figure 5.1: Variation of η^* versus η for various values of ϵ.

The straight diagonal line in the figure represents the case where $\eta^* = \eta$ (i.e., the probability of error at the output of the majority-voting system is equal to the probability of error of any of the individual components). Clearly, we would like to do better than this, because otherwise we are gaining nothing from the majority voting system. Next, note that for small values of ϵ, this is in fact possible. For example, when $\epsilon = 0.05$ and $0.059 < \eta < 0.5$, we have $\eta^* < \eta$. For $\eta < 0.059$, however, we have $\eta^* > \eta$; this is because the unreliable majority voter becomes the bottleneck in this case. The main point is that for small ϵ, the majority voter scheme does, in fact, reduce the probability of error at the output, compared to any single component on its own.

Now consider the case where $\epsilon = \frac{1}{6}$ in Figure 5.1. Here, the curve η^* is always above η for $0 \leq \eta < 0.5$, and crosses only at $\eta = 0.5$. Thus, for $\epsilon = \frac{1}{6}$, the majority voter does not do better than the single component acting on its own. The same is true for $\epsilon > \frac{1}{6}$, as shown in the figure.

Based on the above analysis, we see that we will get an improvement via the majority voting scheme if and only if the curve $\eta^* = \epsilon + (3\eta^2 - 2\eta^3)(1 - 2\epsilon)$ crosses the curve $\eta^* = \eta$ more than once. To find the set of ϵ's for which this happens, we set the two

expressions equal to each other to find the points of intersection:

$$\epsilon + (3\eta^2 - 2\eta^3)(1-2\epsilon) = \eta \;\Leftrightarrow\; \underbrace{2(1-2\epsilon)\eta^3 - 3(1-2\epsilon)\eta^2 + \eta - \epsilon}_{f(\eta)} = 0.$$

As indicated by the figures above, $\eta = \frac{1}{2}$ is always a root of $f(\eta)$. Factoring out $\eta - \frac{1}{2}$, we obtain

$$f(\eta) = 2\left(\eta - \frac{1}{2}\right)((1-2\epsilon)\eta^2 - (1-2\epsilon)\eta + \epsilon).$$

Thus, the other intersections are given by the roots of $(1-2\epsilon)\eta^2 - (1-2\epsilon)\eta + \epsilon$; using the quadratic formula, we obtain

$$\eta = \frac{(1-2\epsilon) \pm \sqrt{(1-2\epsilon)^2 - 4(1-2\epsilon)\epsilon}}{2(1-2\epsilon)} = \frac{1}{2} \pm \frac{1}{2}\sqrt{\frac{1-6\epsilon}{1-2\epsilon}}.$$

Since $1 - 2\epsilon > 0$ (by assumption), we see that $f(\eta)$ will have three (real) roots if and only if $1 - 6\epsilon > 0$, or equivalently, $\epsilon < \frac{1}{6}$. Thus, we have analytically verified the bound that we observed in the figure. Thus, we obtain the following result.

> Suppose that each component is faulty with a probability η, and that the majority voter is faulty with probability ϵ (with both $0 \leq \eta \leq \frac{1}{2}$ and $0 \leq \epsilon \leq \frac{1}{2}$). Then the output of a majority voting system with three redundant components is faulty with probability
>
> $$\eta^* = \epsilon + (3\eta^2 - 2\eta^3)(1-2\epsilon).$$
>
> Furthermore, it is possible to have $\eta^* < \eta$ only if $\epsilon < \frac{1}{6}$.

The above result shows that we can reduce the probability of a fault via a majority voting scheme (only $\epsilon < \frac{1}{6}$); furthermore, we argued earlier that the probability of a fault can never fall below ϵ (the probability that the majority voter itself is faulty). We can now ask the question: how close can we bring the fault probability to ϵ? To answer this question, suppose that $\epsilon < \frac{1}{6}$, and consider the following architecture:

The output of each of the first line of majority voters is incorrect with probability $\eta^* < \eta$. We can now repeat the above analysis to see that the output of the last majority voter is faulty with probability $\eta^{**} < \eta^* < \eta$. Essentially, every time we repeat this process, we 'slide' down the curve $\epsilon+(3\eta^2-2\eta^3)(1-2\epsilon)$ to the point where $\epsilon+(3\eta^2-2\eta^3)(1-2\epsilon) = \eta$, given by

$$\frac{1}{2} - \frac{1}{2}\sqrt{\frac{1-6\epsilon}{1-2\epsilon}}.$$

This defines the lower bound on the fault probability, using a triple-modular-redundancy architecture of the form studied in this section. Note that this lower bound is achievable to an arbitrary degree of precision by creating a long enough chain of majority voters. The price paid for this, of course, is an exponential growth in the number of components required (3^l, where l is the length of the chain), and thus this approach is infeasible in practice, but the bound (and the technique use to derive it) is useful.

Note that if $\epsilon > \frac{1}{6}$, repeated majority voters will *not* help; at every time, the probability of error at the output of each majority voter is worse than the probability of error in any of the inputs, which drives the error probability closer and closer to $\frac{1}{2}$, meaning that one could never obtain any useful information from the output of the last majority voter.

5.2 A Coding-Theory Approach to Reliable Controller Design

In the last section, we considered a majority voting scheme to improve the reliability of faulty components (such as controllers in a feedback loop). This mechanism is simple to implement, and is able to handle arbitrary failures in the controllers (i.e., it does not matter why the output is incorrect). However, we saw that this required a great deal of redundancy. In this section, we will consider a different approach to protecting controllers against a certain number of faults that affect their internal state by using coding-theory to add a small amount of redundancy to the controller itself. This approach was devised in [30, 32], and the material in this section is from those papers.

5.2.1 Background on Parity-Check Codes

We will start by reviewing some background on coding schemes for reliable information transmission. Consider the following simple example. Suppose that we wish to send the string of numbers $(m_1, m_2, m_3, m_4, m_5) = (5, 21, 13, 14, 19)$ across a channel. For convenience, we will represent this string by a vector

$$\mathbf{m} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}.$$

Suppose that the channel can introduce faults into the transmission, where it changes some of the numbers. One option to make sure that we receive the message reliably is to send the message three times, i.e., send a vector of the form

$$\bar{\mathbf{m}} = \begin{bmatrix} \mathbf{m} \\ \mathbf{m} \\ \mathbf{m} \end{bmatrix}.$$

Then, as long as the same number is not corrupted in two or more of the subvectors, we could recover the numbers correctly – this is just a majority voting scheme, as before.

However, now suppose that we consider a model for the channel where it only corrupts one of the numbers in the string. As a first case, suppose that we are only interested in detecting such a corruption. One option is certainly to send the string twice, but this is much more redundancy than we need (since we are only interested in detecting a single corruption). Instead of sending the entire message twice, let us instead send a single additional number with the message. Specifically, suppose that we calculate the sum of the numbers in the original message, given by $s_1 = 5 + 21 + 13 + 14 + 19 = 72$ and send the message

$$\bar{\mathbf{m}} = \begin{bmatrix} \mathbf{m} \\ s_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}.$$

Now suppose that one of the numbers is corrupted in transmission; for example, let us say that 21 is changed to 25. The received message is

$$\bar{\mathbf{m}}_f = \begin{bmatrix} 5 \\ 25 \\ 13 \\ 14 \\ 19 \\ 72 \end{bmatrix} = \bar{\mathbf{m}} + \underbrace{\begin{bmatrix} 0 \\ 4 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{f}}.$$

At the receiver, we know that the last number must be equal to the sum of the other numbers. However, the sum of the first five numbers in the received message is now 76, and this does not match with the last number; thus, we can detect than an error has occurred. More formally, we know that for any correct message $\bar{\mathbf{m}}$, it must be the case that

$$\mathbf{r} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & -1 \end{bmatrix}}_{\mathbf{P}} \bar{\mathbf{m}} = 0.$$

Note that the vector $\mathbf{P}$ is a basis for the left nullspace of $\mathbf{G}$. For the erroneous message, we have

$$\mathbf{r} = \mathbf{P}\bar{\mathbf{m}}_f = \mathbf{P}(\bar{\mathbf{m}} + \mathbf{f}) = \mathbf{P}\mathbf{f}.$$

Suppose a single element of $\mathbf{f}$, say f_i is nonzero; this produces $\mathbf{r} = f_i$ if $1 \leq i \leq 5$ and $\mathbf{s} = -f_i$ if $i = 6$. Thus, $\mathbf{r}$ will be nonzero if there is a single fault. However, we cannot determine which of the numbers were corrupted, since any of them could have had an error with value f_i added to it. In the language of fault diagnosis from Chapter 4, the number $\mathbf{r}$ is the *residual* for this message. As mentioned in the last chapter, the quantity $\mathbf{r}$ is also called the *syndrome* in coding theory [5].

To identify an error, we would have to add more redundancy to the message. Specifically, suppose that we add another number to the message that is generated by taking some linear combination of the numbers, i.e., $s_2 = \alpha_1 m_1 + \alpha_2 m_2 + \alpha_3 m_3 + \alpha_4 m_4 + \alpha_5 m_5$. The sent message is then

$$\bar{\mathbf{m}} = \begin{bmatrix} \mathbf{m} \\ s_1 \\ s_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}$$

and again, the received message is

$$\bar{\mathbf{m}}_f = \bar{\mathbf{m}} + \mathbf{f},$$

where $\mathbf{f}$ is an additive error that affects one of the numbers in the message. Once again, let $\mathbf{P}$ be a matrix whose rows form a basis for the left nullspace of $\mathbf{G}$; one such matrix is

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & -1 & 0 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & 0 & -1 \end{bmatrix}.$$

Thus, the residual (or syndrome) is given by

$$\mathbf{r} = \mathbf{P}\bar{\mathbf{m}}_f = \mathbf{PGm} + \mathbf{Pf} = \mathbf{Pf}.$$

As in the parity-space fault diagnosis method studied in Section 4.3, the columns of $\mathbf{P}$ determine whether we can identify a nonzero element of $\mathbf{f}$. Specifically, we can uniquely identify which element is nonzero if the residual can only be generated by a single column of $\mathbf{P}$. This is accomplished if any *two* columns of $\mathbf{P}$ are linearly independent. To see why, suppose that this condition holds, but that two different elements of $\mathbf{f}$, say f_i and f_j, can produce the same residual, i.e.,

$$\mathbf{r} = \mathbf{p}_i f_i = \mathbf{p}_j f_j$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the i–th and j–th columns of $\mathbf{P}$, respectively. However, this would imply that $\mathbf{p}_i$ and $\mathbf{p}_j$ are linearly dependent, which contradicts the assumption.

So, if $\alpha_1, \alpha_2, \ldots, \alpha_5$ are chosen so that any two columns of $\mathbf{P}$ are linearly independent, we can uniquely identify any single fault that affects the message. For example, one can verify that this is achieved by choosing the α_i's to all be different.

More generally, suppose that we wish to send k numbers (given by the vector $\mathbf{m} \in \mathbb{R}^k$, and we generate the transmitted (or redundant) message as follows:

$$\bar{\mathbf{m}} = \underbrace{\begin{bmatrix} \mathbf{I}_k \\ \mathbf{H} \end{bmatrix}}_{\mathbf{G}} \mathbf{m}, \tag{5.2}$$

where $\mathbf{H}$ is a $d \times k$ matrix (i.e., we are adding d redundant numbers to the message). The matrix $\mathbf{P}$ whose rows form a basis for the left nullspace of $\mathbf{G}$ is then given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{H} & -\mathbf{I}_d \end{bmatrix}. \tag{5.3}$$

The matrix $\mathbf{G}$ is called the *generator* matrix, and the matrix $\mathbf{P}$ is called the *parity check* matrix.

Theorem 5.1. *Let t be a nonnegative integer such that any $2t$ columns of $\mathbf{P}$ are linearly independent. Then:*

- *As long as no more than $2t$ faults occur, they can be detected.*
- *As long as no more than t faults occur, they call be identified.*

Proof. Suppose that $2t$ or fewer faults occur, so that at most $2t$ elements of $\mathbf{f}$ are nonzero. Thus, the residual is given by

$$\mathbf{r} = \mathbf{P}\mathbf{f}$$

which is a linear combination of at most $2t$ columns from $\mathbf{P}$. Since any $2t$ columns are linearly independent, we see that the residual must be nonzero, and thus these faults can be detected.

Now suppose instead that t or fewer faults occur, so that at most t elements of $\mathbf{f}$ are nonzero. Again, the residual is given by $\mathbf{r} = \mathbf{P}\mathbf{f}$, which is now a linear combination of t columns of $\mathbf{P}$. We can identify which elements are nonzero if the residual could not have been generated by any other set of t nonzero elements. Specifically, suppose that two different sets of t nonzero elements (i.e., two different sets of t faults) can produce the same residual. Let the fault vectors for these two cases be given by $\mathbf{f}$ and $\bar{\mathbf{f}}$, respectively, where each of those vectors have no more than t nonzero entries. Then if it is the case that

$$\mathbf{r} = \mathbf{P}\mathbf{f} = \mathbf{P}\bar{\mathbf{f}},$$

then, it must be the case that $\mathbf{P}(\mathbf{f} - \bar{\mathbf{f}}) = \mathbf{0}$. However, the vector $\mathbf{f} - \bar{\mathbf{f}}$ has at most $2t$ nonzero entries, and thus $\mathbf{P}(\mathbf{f} - \bar{\mathbf{f}})$ is a linear combination of at most $2t$ columns of $\mathbf{P}$. Since any $2t$ columns of $\mathbf{P}$ are linearly independent, the only way for $\mathbf{P}(\mathbf{f} - \bar{\mathbf{f}}) = \mathbf{0}$ is if $\mathbf{f} - \bar{\mathbf{f}} = 0$, i.e., $\mathbf{f} = \bar{\mathbf{f}}$. In other words, as long as there are no more than t faults, every residual *uniquely* determines which set of t faults occurred. □

Thus, in order to design a message that is resilient to t faults, one must choose the matrix $\mathbf{H}$ in (5.2) so that any $2t$ columns of $\mathbf{P}$ in (5.3) are linearly independent. Note that since $\mathbf{P}$ has d rows, we need $d \geq 2t$ in order for this to be possible (i.e., we need to add at least $2t$ redundant numbers to the message). Under this condition, there are various ways to choose $\mathbf{H}$ to achieve the desired linear independence. We will see one such scheme in the next section, when we apply these concepts to designing reliable controllers.

5.2.2 Reliable Controller Design

Suppose that we consider a controller that is a dynamical system of the form

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k], \end{aligned} \tag{5.4}$$

where the state $\mathbf{x} \in \mathbb{R}^n$. In the previous chapters, we used this notation to represent the plant that we were trying to control, but for convenience, we will use the same notation here for the controller. Note that the observer-feedback based controllers from Section 2.8 are of this form. These controllers are simply dynamical systems that are implemented in a computer: each of the state-variables would be stored in a register, for example. However, digital computers can potentially experience faults; they might incorrectly perform a computation due to some error, or the contents of certain registers might get corrupted due to environmental factors, such as alpha particles in aircraft [56, 26]. In this section, we will focus on errors that affect the state-variables in the controller, where at some time-step k, some state $x_i[k]$ gets changed (erroneously) to some other state $x_i[k] + f[k]$, for some additive error $f[k]$. Since we are assuming that we have access to the entire state of the controller, we will disregard the output of the controller $\mathbf{y}[k]$ in our investigation, and study how we can add redundancy to the controller to detect and identify such state errors.

If we have access to the whole state, why we do not simply just verify that the state is at the correct value by checking whether $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k]$ at each k. However, this essentially amounts to performing the entire state update calculation a second time, which is equivalent to modular redundancy. The goal here is not to replicate the entire controller, but to add just enough additional redundancy to detect and identify a fixed number of state errors. To do this, we will be using the techniques from the previous section on parity check codes to construct a *redundant controller* of the form

$$\bar{\mathbf{x}}[k+1] = \bar{\mathbf{A}}\bar{\mathbf{x}}[k] + \bar{\mathbf{B}}\mathbf{u}[k], \tag{5.5}$$

where the state vector $\bar{\mathbf{x}}$ is of dimension $n + d$ for some nonnegative integer d. This controller should be designed so that it provides the same functionality as the original controller, but with some additional redundancy. To capture this, the following definition was used in [32].

Definition 5.1. The controller (5.5) is a *redundant implementation* of the controller (5.4) if, for all time-steps k, we have

$$\bar{\mathbf{x}}[k] = \mathbf{G}\mathbf{x}[k], \tag{5.6}$$

$$\mathbf{x}[k] = \mathbf{L}\bar{\mathbf{x}}[k], \tag{5.7}$$

for some generator matrix $\mathbf{G}$ and some decoder matrix $\mathbf{L}$.

The above definition says that the state of the redundant controller should be obtained as a redundant version of the state of the original controller (i.e., it must lie in the column space of the matrix $\mathbf{G}$), and that the state of the original controller should also be obtained from state of the redundant controller (so that the same functionality is maintained). As in the case of parity check codes discussed in the previous section, we can define a *parity check matrix* $\mathbf{P}$ such that the rows form a basis for the left nullspace of $\mathbf{G}$. Under normal (fault-free) conditions, we must have $\mathbf{P}\bar{\mathbf{x}}[k] = \mathbf{0}$ for all k. To see what these conditions imply for the matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, consider the following result.

Theorem 5.2. *The controller* (5.5) *is a redundant implementation of* (5.4) *if and only if there exists a similarity transformation matrix* $\mathbf{T}$ *such that*

$$\mathbf{T}\bar{\mathbf{A}}\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}, \ , \mathbf{T}\bar{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}, \tag{5.8}$$

where $\mathbf{A}_{12}$ *and* $\mathbf{A}_{22}$ *are some* $n \times d$ *and* $d \times d$ *matrices that are free to choose. Furthermore,*

$$\mathbf{TG} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{0} \end{bmatrix}, \ \mathbf{LT}^{-1} = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \end{bmatrix}, \ \mathbf{PT}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_d \end{bmatrix}.$$

Proof. From (5.6) and (5.7), we have that

$$\mathbf{x}[k] = \mathbf{L}\bar{\mathbf{x}}[k] = \mathbf{LG}\mathbf{x}[k],$$

and this must hold for all $\mathbf{x}[k]$. This means that $\mathbf{LG} = \mathbf{I}_n$. Let $\mathbf{T}$ be a matrix of the form

$$\mathbf{T} = \begin{bmatrix} \mathbf{L} \\ \mathbf{N}_G \end{bmatrix},$$

where the rows of the $d \times (n + d)$ matrix $\mathbf{N}_G$ form a basis for the left nullspace of $\mathbf{G}$. Then, we have

$$\mathbf{TG} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{0} \end{bmatrix}, \ \mathbf{LT}^{-1} = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \end{bmatrix} \mathbf{TT}^{-1} = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \end{bmatrix}.$$

Furthermore, since the rows of $\mathbf{P}$ form a basis for the left nullspace of $\mathbf{G}$, we see that this is achieved by choosing $\mathbf{PT}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_d \end{bmatrix}$.

Now perform a similarity transformation on the controller (5.5) by defining $\bar{\mathbf{x}}_r[k] = \mathbf{T}\bar{\mathbf{x}}[k]$. This produces

$$\begin{aligned} \bar{\mathbf{x}}_r[k+1] &= \mathbf{T}\bar{\mathbf{A}}\mathbf{T}^{-1}\bar{\mathbf{x}}_r[k] + \mathbf{T}\bar{\mathbf{B}}\mathbf{u}[k] \\ &= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \bar{\mathbf{x}}_r[k] + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix} \mathbf{u}[k]. \end{aligned} \tag{5.9}$$

Alternatively, note that

$$\bar{\mathbf{x}}_r[k] = \mathbf{T}\bar{\mathbf{x}}[k] = \mathbf{TG}\mathbf{x}[k] = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{0} \end{bmatrix} \mathbf{x}[k],$$

so that

$$\bar{\mathbf{x}}_r[k+1] = \begin{bmatrix} \mathbf{x}[k+1] \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{0} \end{bmatrix}.$$

Substituting this into (5.9), we have

$$\begin{bmatrix} \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}[k] \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix} \mathbf{u}[k].$$

This equation has to hold for all possible values of $\mathbf{x}[k]$ and $\mathbf{u}[k]$; in particular, if we consider $\mathbf{u}[k] = \mathbf{0}$, the bottom row of the above system says that $\mathbf{0} = \mathbf{A}_{12}\mathbf{x}[k]$, and thus it must be that $\mathbf{A}_{12} = \mathbf{0}$. Now, if we allow $\mathbf{u}[k]$ to be nonzero, we see that

$$\mathbf{0} = \mathbf{A}_{22}\mathbf{0} + \mathbf{B}_2\mathbf{u}[k] = \mathbf{B}_2\mathbf{u}[k],$$

and thus $\mathbf{B}_2 = \mathbf{0}$. However, the above equation does not constrain matrices $\mathbf{A}_{12}$ or $\mathbf{A}_{22}$ in any way. □

The matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$ are free for us to choose, and characterize all of the freedom that we have in designing a redundant implementation. Specifically, we will show how to choose these matrices in order to allow detection and identification of a fixed number of faults.

Detecting and Identifying Faults

Suppose that we have constructed the controller (5.5) so that it is a redundant implementation of the original controller (5.4) (i.e., it satisfies the conditions in Definition 5.1), and that at some time-step $k-1$, some of the states of the redundant controller becomes affected by faults as they are being updated. This produces a *faulty* state

$$\begin{aligned} \bar{\mathbf{x}}_f[k] &= \bar{\mathbf{A}}\bar{\mathbf{x}}[k] + \bar{\mathbf{B}}\mathbf{u}[k] + \mathbf{f}[k] \\ &= \bar{\mathbf{x}}[k] + \mathbf{f}[k] \\ &= \mathbf{G}\mathbf{x}[k] + \mathbf{f}[k], \end{aligned} \tag{5.10}$$

where $\mathbf{f}_i$ is a vector with a single nonzero element in its i–th entry, and zeros everywhere else. Now, let $\mathbf{P}$ be a matrix whose rows form a basis for the left nullspace of $\mathbf{G}$ (note that $\mathbf{P}$ has dimension $\mathbf{d} \times (n+d)$). We can now calculate the residual (or syndrome)

$$\mathbf{r}[k] = \mathbf{P}\bar{\mathbf{x}}_f[k] = \mathbf{P}\mathbf{G}\mathbf{x}[k] + \mathbf{P}\mathbf{f}[k] = \mathbf{P}\mathbf{f}[k].$$

Thus, the residual is a linear combination of all columns of $\mathbf{P}$ corresponding to the entries of $\mathbf{f}[k]$ that are nonzero. As in the previous section, if we want to be able to detect any $2t$ state-transition faults, or identify any t state-transition faults, we need to any $2t$ columns of the parity check matrix $\mathbf{P}$ to be linearly independent. Rather than focusing on designing the system with only this in mind, we will instead consider the following objective. Instead of checking for errors in the state at *every* time-step, we would instead like to only check for errors periodically, say, every N time-steps for some

N. This would reduce the burden on the checking mechanism, and perhaps allow it to operate more reliably.

To this end, we will break up the operation of the controller into periods of N time-steps. At the end of each N time-steps, we will examine the state and try to identify any errors that occurred during the last period, and then correct all of those errors so the state at the start of the next period is what it should be. For this reason, we can just consider what happens in the first N time-steps, and the same analysis will hold for all subsequent periods.

Suppose that the redundant controller is initialized with state $\bar{\mathbf{x}}[0] = \mathbf{G}\mathbf{x}[0]$ for some appropriately chosen $\mathbf{G}$ (which we will describe later). Any faults that occur at time-step k are captured by the vector $\mathbf{f}[k]$, so that the faulty state at time-step k gets updated according to equation (5.10). The faulty state at the end of N time-steps is then given by

$$\begin{aligned}\bar{\mathbf{x}}_f[N] &= \bar{\mathbf{A}}^N\bar{\mathbf{x}}[0] + \begin{bmatrix}\bar{\mathbf{A}}^{N-1}\bar{\mathbf{B}} & \bar{\mathbf{A}}^{N-2}\bar{\mathbf{B}} & \cdots & \bar{\mathbf{B}}\end{bmatrix}\mathbf{u}[0:N-1] \\ &\quad + \begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix}\mathbf{f}[0:N-1] \\ &= \bar{\mathbf{x}}[N] + \begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix}\mathbf{f}[0:N-1] \\ &= \mathbf{G}\mathbf{x}[N] + \begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix}\mathbf{f}[0:N-1].\end{aligned}$$

If we perform a parity check on this state, we obtain the residual

$$\mathbf{r}[N] = \mathbf{P}\bar{\mathbf{x}}_f[N] = \underbrace{\mathbf{P}\begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix}}_{\mathbf{R}}\mathbf{f}[0:N-1]. \tag{5.11}$$

Thus, the residual is a linear combination of the columns $\mathbf{R}$. Just as before, if we want to be able to identify up to t errors in a period of N time-steps, we want to make sure that any $2t$ columns of this matrix are linearly independent. The objective will be to choose the redundant dynamics $\mathbf{A}_{22}$ and $\mathbf{A}_{12}$ so that this is the case. The following result shows how these matrices show up in the matrix $\mathbf{R}$.

Lemma 5.1. *If controller* (5.5) *is a redundant implementation of* (5.4)*, then for any* k*,* $\mathbf{P}\bar{\mathbf{A}}^k = \mathbf{A}_{22}^k\mathbf{P}$*.*

Proof. From Theorem 5.2, we see that there exists a matrix $\mathbf{T}$ such that (5.8) holds. Furthermore, $\mathbf{T}\mathbf{G} = \begin{bmatrix}\mathbf{I}_n \\ \mathbf{0}\end{bmatrix}$. Since $\mathbf{P}\mathbf{G} = \mathbf{0}$, we see that we can choose $\mathbf{P}\mathbf{T}^{-1} = \begin{bmatrix}\mathbf{0} & \mathbf{I}_d\end{bmatrix}$. Thus, we have

$$\begin{aligned}\mathbf{P}\bar{\mathbf{A}}^k &= \mathbf{P}\mathbf{T}^{-1}\mathbf{T}\left(\mathbf{T}^{-1}\begin{bmatrix}\mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22}\end{bmatrix}\mathbf{T}\right)^k \\ &= \begin{bmatrix}\mathbf{0} & \mathbf{I}_d\end{bmatrix}\begin{bmatrix}\mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22}\end{bmatrix}^k\mathbf{T} \\ &= \begin{bmatrix}\mathbf{0} & \mathbf{I}_d\end{bmatrix}\begin{bmatrix}\mathbf{A}^k & * \\ \mathbf{0} & \mathbf{A}_{22}^k\end{bmatrix}\mathbf{T},\end{aligned}$$

where $*$ represents an unimportant matrix. This produces

$$\begin{aligned}\mathbf{P}\bar{\mathbf{A}}^k &= \begin{bmatrix}\mathbf{0} & \mathbf{A}_{22}^k\end{bmatrix}\mathbf{T} \\ &= \mathbf{A}_{22}^k\begin{bmatrix}\mathbf{0} & \mathbf{I}_d\end{bmatrix}\mathbf{T} \\ &= \mathbf{A}_{22}^k\mathbf{P}.\end{aligned}$$

This produces the desired result. □

The above lemma indicates that $\mathbf{R}$ is

$$\begin{aligned}\mathbf{R} &= \mathbf{P}\begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{A}_{22}^{N-1}\mathbf{P} & \mathbf{A}_{22}^{N-2}\mathbf{P} & \cdots & \mathbf{P}\end{bmatrix}.\end{aligned}$$

Thus, the ability to identify errors depends entirely on the matrices $\mathbf{P}$ and $\mathbf{A}_{22}$, and not on $\mathbf{A}_{12}$. We will now focus on choosing $\mathbf{A}_{22}$ and $\mathbf{P}$ so that any $2t$ columns of the matrix $\mathbf{R}$ are linearly independent.

Designing the Redundant Dynamics

To design $\mathbf{P}$ and $\mathbf{A}_{22}$, we will find it convenient to consider matrices of the following form.

Definition 5.2. For any real numbers $p_1, p_2, p_3, \ldots, p_s$, the $2t \times s$ matrix

$$\mathbf{V}(p_1, p_2, \ldots, p_s) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ p_1 & p_2 & p_3 & \cdots & p_s \\ p_1^2 & p_2^2 & p_3^2 & \cdots & p_s^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_1^{2t-1} & p_2^{2t-1} & p_3^{2t-1} & \cdots & p_s^{2t-1} \end{bmatrix}$$

is called a *Vandermonde* matrix.

The following result about Vandermonde matrices will be key.

Lemma 5.2. *If $s \geq 2t$, any $2t$ columns of the above matrix will be linearly independent if and only if all of the p_i's are distinct.*

The proof of the above lemma can be found in standard linear algebra books [5]. This result will be very useful to us; our strategy will be to make the matrix $\mathbf{R}$ have the form of a Vandermonde matrix, where every column corresponds to a different real number. Then, any $2t$ columns of $\mathbf{R}$ will be linearly independent. To achieve this, we choose $\mathbf{P}$ and $\mathbf{A}_{22}$ as follows.

1. If we want to detect and identify t errors in any period of N time-steps, the matrix $\mathbf{A}_{22}$ should have dimensions $2t \times 2t$, and $\mathbf{P}$ should have dimensions $2t \times (n+2t)$ (i.e., $d = 2t$ extra state variables should be added).

2. Choose $n + 2t + 1$ real numbers $p, p_1, p_2, \ldots, p_{n+2t}$, and form the following matrices

$$\begin{aligned} \Lambda &= \text{diag}(1, p, p^2, \cdots, p^{2t-2}, p^{2t-1}) \\ \mathbf{M} &= \mathbf{V}(p_{n+1}, p_{n+2}, \cdots, p_{n+2t}) \\ \mathbf{H} &= -\mathbf{M}^{-1}\mathbf{V}(p_1, p_2, \cdots, p_n). \end{aligned} \tag{5.12}$$

3. Form the matrices

$$\mathbf{A}_{22} = \mathbf{M}^{-1}\Lambda\mathbf{M}, \;\; \mathbf{P} = \begin{bmatrix} -\mathbf{H} & \mathbf{I}_{2t} \end{bmatrix}. \tag{5.13}$$

Using the matrices defined by the above procedure, we have

$$\begin{aligned} \mathbf{A}_{22}^k\mathbf{P} &= \mathbf{M}^{-1}\Lambda^k\mathbf{M}\begin{bmatrix} -\mathbf{H} & \mathbf{I}_{2t} \end{bmatrix} \\ &= \mathbf{M}^{-1}\Lambda^k\begin{bmatrix} -\mathbf{M}\mathbf{H} & \mathbf{M} \end{bmatrix} \\ &= \mathbf{M}^{-1}\Lambda^k\begin{bmatrix} \mathbf{V}(p_1, p_2, \cdots, p_n) & \mathbf{V}(p_{n+1}, p_{n+2}, \cdots, p_{n+2t}) \end{bmatrix} \\ &= \mathbf{M}^{-1}\Lambda^k\mathbf{V}(p_1, p_2, p_3, \cdots, p_{n+2t}). \end{aligned}$$

It is easy to verify that

$$\Lambda^k\mathbf{V}(p_1, p_2, p_3, \cdots, p_{n+2t}) = \mathbf{V}(p^k p_1, p^k p_2, p^k p_3, \cdots, p^k p_{n+2t}),$$

so that

$$\mathbf{A}_{22}^k\mathbf{P} = \mathbf{M}^{-1}\mathbf{V}(p^k p_1, p^k p_2, p^k p_3, \cdots, p^k p_{n+2t}).$$

Thus, the matrix $\mathbf{R}$ is given by

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} \mathbf{A}_{22}^{N-1}\mathbf{P} & \mathbf{A}_{22}^{N-2}\mathbf{P} & \cdots & \mathbf{P} \end{bmatrix} \\ &= \mathbf{M}^{-1}\begin{bmatrix} \mathbf{V}(p^{N-1}p_1, \cdots, p^{N-1}p_{n+2t}) & \mathbf{V}(p^{N-2}p_1, \cdots, p^{N-2}p_{n+2t}) & \cdots & \mathbf{V}(p_1, \cdots, p_{n+2t}) \end{bmatrix} \\ &= \mathbf{M}^{-1}\mathbf{V}(p^{N-1}p_1, \cdots, p^{N-1}p_{n+2t}, p^{N-2}p_1, \cdots, p^{N-2}p_{n+2t}, \cdots, p_1, \cdots, p_{n+2t}). \end{aligned}$$

If the numbers $p, p_1, \ldots, p_{n+2t}$ are chosen so that $p^k p_i \neq p^{k'} p_j$ for any $0 \leq k < k' \leq N-1$, then Lemma 5.2 indicates that any $2t$ columns of

$$\mathbf{V}(p^{N-1}p_1, \cdots, p^{N-1}p_{n+2t}, p^{N-2}p_1, \cdots, p^{N-2}p_{n+2t}, \cdots, p_1, \cdots, p_{n+2t})$$

will be invertible. Furthermore, since multiplying a set of linearly independent columns on the left by an invertible matrix does not change the linear independence of those

columns, we see that any $2t$ columns of $\mathbf{R}$ will be linearly independent, allowing us to identify up to t errors.

Finally, since $\mathbf{P} = \begin{bmatrix}\mathbf{0} & \mathbf{I}_d\end{bmatrix}\mathbf{T}$, we see that the last d rows of $\mathbf{T}$ must be of the form $\begin{bmatrix}-\mathbf{H} & \mathbf{I}_{2t}\end{bmatrix}$. Furthermore, since $\mathbf{L} = \begin{bmatrix}\mathbf{I}_n & \mathbf{0}\end{bmatrix}\mathbf{T}$, we see that the top n rows of $\mathbf{T}$ have the form $\begin{bmatrix}\mathbf{I}_n & \mathbf{0}\end{bmatrix}$. Thus, the similarity transformation matrix is given by

$$\mathbf{T} = \begin{bmatrix}\mathbf{I}_n & \mathbf{0} \\ -\mathbf{H} & \mathbf{I}_{2t}\end{bmatrix}.$$

We can now design the redundant implementation as

$$\begin{aligned}\bar{\mathbf{x}}[k+1] &= \mathbf{T}^{-1}\begin{bmatrix}\mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22}\end{bmatrix}\mathbf{T}\bar{\mathbf{x}}[k] + \mathbf{T}^{-1}\begin{bmatrix}\mathbf{B} \\ \mathbf{0}\end{bmatrix}\mathbf{u}[k] \\ &= \begin{bmatrix}\mathbf{I}_n & \mathbf{0} \\ \mathbf{H} & \mathbf{I}_{2t}\end{bmatrix}\begin{bmatrix}\mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{I}_n & \mathbf{0} \\ -\mathbf{H} & \mathbf{I}_{2t}\end{bmatrix} + \begin{bmatrix}\mathbf{I}_n & \mathbf{0} \\ \mathbf{H} & \mathbf{I}_{2t}\end{bmatrix}\begin{bmatrix}\mathbf{B} \\ \mathbf{0}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{A} - \mathbf{A}_{12}\mathbf{H} & \mathbf{A}_{12} \\ \mathbf{HA} - \mathbf{HA}_{12}\mathbf{H} - \mathbf{A}_{22}\mathbf{H} & \mathbf{HA}_{12} + \mathbf{A}_{22}\end{bmatrix}\bar{\mathbf{x}}[k] + \begin{bmatrix}\mathbf{B} \\ \mathbf{HB}\end{bmatrix}\mathbf{u}[k].\end{aligned}$$

Note that $\mathbf{A}_{12}$ can be *arbitrary* in the above implementation; for simplicity, we can choose $\mathbf{A}_{12}$ to be the zero matrix. The above redundant controller is initialized with

$$\bar{\mathbf{x}}[0] = \mathbf{G}\mathbf{x}[0] = \mathbf{T}^{-1}\begin{bmatrix}\mathbf{I}_n \\ \mathbf{0}\end{bmatrix}\mathbf{x}[0] = \begin{bmatrix}\mathbf{I}_n \\ \mathbf{H}\end{bmatrix}\mathbf{x}[0].$$

Summary of Design Procedure

The design procedure described in the previous sections is now summarized.

1. To identify t errors, set $d = 2t$.
2. Choose matrices $\Lambda, \mathbf{H}$ and $\mathbf{M}$ according to (5.12), and form matrices $\mathbf{A}_{22}$ and $\mathbf{P}$ according to (5.13).
3. Form the redundant controller (choosing $\mathbf{A}_{12} = \mathbf{0}$)

   $$\bar{\mathbf{x}}[k+1] = \begin{bmatrix}\mathbf{A} & \mathbf{0} \\ \mathbf{HA} - \mathbf{A}_{22}\mathbf{H} & \mathbf{A}_{22}\end{bmatrix}\bar{\mathbf{x}}[k] + \begin{bmatrix}\mathbf{B} \\ \mathbf{HB}\end{bmatrix}\mathbf{u}[k],$$

   with

   $$\bar{\mathbf{x}}[0] = \begin{bmatrix}\mathbf{I}_n \\ \mathbf{H}\end{bmatrix}\mathbf{x}[0].$$

4. Run the controller. Every N time-steps, perform a parity check on the (possibly) faulty state $\bar{\mathbf{x}}_f[N]$ and calculate the residual

   $$\mathbf{r}[N] = \mathbf{P}\bar{\mathbf{x}}_f[N] = \mathbf{R}\mathbf{f}[0 : N-1].$$

   Find the t (or fewer) columns of $\mathbf{R}$ that produce the residual, and solve for $\mathbf{f}[0 : N-1]$ (this will have at most t nonzero entries).

5. Find the correct value of the state at time-step N as

$$\bar{\mathbf{x}}[N] = \bar{\mathbf{x}}_f[N] - \begin{bmatrix}\bar{\mathbf{A}}^{N-1} & \bar{\mathbf{A}}^{N-2} & \cdots & \mathbf{I}_n\end{bmatrix} \mathbf{f}[0:N-1].$$

Chapter 6

Control Over Packet-Dropping Channels

So far, we have considered the issue of reliability in two different elements of the feedback loop. In Chapter 4, we studied estimator-based methods to diagnose faults that affect the plant itself. In the last chapter, we considered faults that affect the controller, and described schemes to develop fault-tolerant controllers. In this chapter, we will study what happens when the link *between* the plant and the controller is unreliable. Specifically, we will consider links that drop the information (or packets) that they carry with a certain probability, so that the plant's actuators receive no information at certain time-steps.

The issue of stability under such packet-dropping channels has been studied extensively

over the past decade in the context of *networked control* [20, 31, 37, 77, 46, 28, 68, 69, 73, 74]. Many of these works also consider the issue of *optimality* of the control scheme under packet drops. In this chapter, we will focus on the simpler issue of stability in order to understand the fundamental limitations of networked control.

6.1 Control of a Scalar System with Packet Drops

To start, we will consider a simple scalar plant of the form

$$x[k+1] = \alpha x[k] + \beta u[k],$$

where $x \in \mathbb{R}$ and $u \in \mathbb{R}$. The state of the plant is sent through a channel to a controller, which then applies an appropriate control input to the plant. When the channel is perfectly reliable, the controller can apply the input $u[k] = -\gamma x[k]$, where γ is chosen so that $\alpha - \beta\gamma$ has magnitude less than one. However, suppose that the packet is dropped with a certain probability p. In other words, at each time-step k, the controller applies the input $u[k] = -\gamma x[k]$ with probability $1-p$, and applies $u[k] = 0$ with probability p. This produces the closed loop system

$$x[k+1] = \begin{cases} (\alpha - \beta\gamma)x[k] & \text{with probability 1 - p} \\ \alpha x[k] & \text{with probability p} \end{cases}. \tag{6.1}$$

We will be interested in making the state of the system 'small', or at least prevent it from exploding to infinity. To make this more formal, suppose that $|\alpha| < 1$, in which case the system is already stable. In this case, even if we never apply an input to the system, the state will go to zero by itself, and thus this case is not too interesting. If $|\alpha| \geq 1$, on the other hand, the state will generally blow up (or stay constant) unless we apply an appropriate control signal. However, since inputs are applied *probabilistically* in the above setup, there is a chance (albeit small) that we will not apply inputs to the system for long stretches of time; specifically, the probability that we do not apply an input for a specific set of L contiguous time-steps is given by p^L. Thus, the usual notion of deterministic stability will have be to changed in order to properly analyze this scenario.

For example, we could ask what happens to the state *on average*. That is, we can study the *expected value* of the state $E[x[k]]$ and ask if that goes to zero, or is bounded. However, this solution is not satisfactory: the state could go to positive infinity half the time, and to negative infinity the other half of the time, and the average state would still be zero. To avoid this, we could consider instead the expected value of the *square* of the state; if this value is bounded, then the state *must* be bounded (on average). While we have been discussing a scalar system, we can generalize our discussion to vector states via the notion of a vector norm (see Section A.4.7 in Appendix A).

Definition 6.1. The system is said to be mean square stable if $E[\|\mathbf{x}[k]\|_2^2] < \infty$ for all k.

Based on (6.1), the mean squared state is given by

$$\begin{aligned}\sigma[k+1] &\triangleq E[x[k+1]^2] \\ &= E[((\alpha-\beta\gamma)x[k])^2](1-p) + E[(\alpha x[k])^2])p \\ &= (\alpha-\beta\gamma)^2 E[x[k]^2](1-p) + \alpha^2 E[x[k]^2]p \\ &= \left((\alpha^2 - 2\alpha\beta\gamma + \beta^2\gamma^2)(1-p) + \alpha^2 p\right) E[x[k]^2] \\ &= \left(\alpha^2 - 2\alpha\beta\gamma(1-p) + \beta^2\gamma^2(1-p)\right) E[x[k]^2] \\ &= \underbrace{\left(\alpha^2 - 2\alpha\beta\gamma(1-p) + \beta^2\gamma^2(1-p)\right)}_{f(\gamma)} \sigma[k].\end{aligned}$$

The above expression signifies that we can view $\sigma[k]$ as the state of a dynamical system, where the dynamics are given by $f(\gamma) = \alpha^2 - 2\alpha\beta\gamma(1-p) + \beta^2\gamma^2(1-p)$. If we can choose the parameter γ so that this quantity has magnitude less than 1, then we can guarantee that $\sigma[k] \to 0$, which proves mean square stability. To do this, let us find the minimum possible value of $f(\gamma)$ by differentiating with respect to γ and setting it equal to zero; this produces

$$\frac{df}{d\gamma} = -2\alpha\beta(1-p) + 2\beta^2(1-p)\gamma = 0 \Rightarrow \gamma_{opt} = \frac{\alpha}{\beta}.$$

Note that this is the same γ_{opt} that would be used to set $\alpha - \beta\gamma = 0$ even when the channels are perfectly reliable. With this γ, we have

$$f(\gamma_{opt}) = \alpha^2 - 2\alpha^2(1-p) + \alpha^2(1-p) = \alpha^2 p.$$

Thus we can never reduce $f(\gamma)$ to below $\alpha^2 p$, and so we will be able to obtain mean square stability if and only if $\alpha^2 p < 1$. This produces the following result.

Theorem 6.1. *Consider a scalar plant with dynamics given by α. Let the channel between the plant and the controller drop packets with a probability p. Then, it is possible to obtain mean square stability with a linear controller if and only if $\alpha^2 p < 1$.*

6.2 General Linear Systems with Packet Drops

We will now turn our attention to state-space models of the form

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k],\end{aligned} \tag{6.2}$$

with state $\mathbf{x} \in \mathbb{R}^n$, input $\mathbf{u} \in \mathbb{R}^m$ and output $\mathbf{y} \in \mathbb{R}^p$. Before we generalize the discussion in the previous section to study mean square stability, we will find it useful to review a different way of examining the stability of linear systems (without packet drops).

6.2.1 Lyapunov Stability of Linear Systems

In Section 2.5, we considered the stability of a linear system $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$, and showed that $\mathbf{x}[k] \to \mathbf{0}$ if and only if all eigenvalues of $\mathbf{A}$ are stable (i.e., have magnitude less than 1). We will now cover an alternative characterization of stability that will also be useful; we will be making extensive use of the material on positive definite matrices and vector norms from Section A.4.6 and A.4.7 in Appendix A.

First, the definition of stability (Definition 2.2) says that the state of the system should asymptotically go to zero. Since the state is a vector, this definition means that every component of the state vector should asymptotically go to zero. Rather than dealing with the vector directly, we could try to capture the 'size' of the state somehow, and show that this size goes to zero asymptotically. If we define the size of the vector appropriately (i.e., so that the size equals zero if and only if the vector itself is zero), and we show that the size of the vector goes to zero, then we can prove stability of the system.

To do this, we will use the notion of the *norm* of a vector. Recall from Section A.4.7 in Appendix A that for a given vector $\mathbf{a}$ and a norm $\rho(\cdot)$, $\rho(\mathbf{a}) \geq 0$, with equality if and only if $\mathbf{a} = \mathbf{0}$. Thus, for the system $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$, if we can show that $\rho(\mathbf{x}[k]) \to 0$ for some appropriate norm $\rho(\cdot)$, then we will have shown that the system is stable. Specifically, the goal will be to show that $\rho(\mathbf{x}[k+1]) < \rho(\mathbf{x}[k])$ for all k and all $\mathbf{x}[k]$, so that the norm is constantly decreasing. Since the norm of a vector is lower bounded by 0 (by definition), we will have shown that the norm of the state is converges to something. We will then argue that it must converge to 0.

An obvious first choice would be to consider the function $V(\mathbf{x}[k]) = \|x[k]\|_2^2$ (i.e., the square of the 2-norm of the state vector). This produces

$$\begin{aligned} V(\mathbf{x}[k+1]) - V(\mathbf{x}[k]) &= \|\mathbf{x}[k+1]\|_2^2 - \|\mathbf{x}[k]\|_2^2 \\ &= \|\mathbf{A}\mathbf{x}[k]\|_2^2 - \|\mathbf{x}[k]\|_2^2 \\ &= \mathbf{x}[k]'\mathbf{A}'\mathbf{A}\mathbf{x}[k] - \mathbf{x}[k]'\mathbf{x}[k] \\ &= \mathbf{x}[k]'\left(\mathbf{A}'\mathbf{A} - \mathbf{I}\right)\mathbf{x}[k]. \end{aligned}$$

In order to ensure that this is negative regardless of $\mathbf{x}[k]$, we would have to ensure that the matrix $\mathbf{A}'\mathbf{A} - \mathbf{I}$ is negative definite (see Section A.4.6). Certainly, if this condition is satisfied, then we have shown that $V(\mathbf{x}[k+1]) < V(\mathbf{x}[k])$ for all k (which means that $\rho(\mathbf{x}[k+1]) < \rho(\mathbf{x}[k])$). However, even if the system is stable, it is possible that $\mathbf{A}'\mathbf{A} - \mathbf{I}$ is not negative definite, in which case this test fails. This is illustrated by the following example.

Example 6.1. Consider the system

$$\mathbf{x}[k+1] = \begin{bmatrix} 0.5 & 1 \\ 0 & 0.5 \end{bmatrix} \mathbf{x}[k].$$

The eigenvalues of $\mathbf{A}$ are 0.5 and 0.5, and thus the system is stable. To see if the 2-norm of the system state decreases at each time-step, we check the matrix

$$\mathbf{A}'\mathbf{A} - \mathbf{I}_2 = \begin{bmatrix} 0.5 & 0 \\ 1 & 0.5 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \\ 0 & 0.5 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.75 & 0.5 \\ 0.5 & 0.25 \end{bmatrix}.$$

This matrix is *not* negative definite; it has eigenvalues at -0.9571 and 0.4571 (recall that a negative definite matrix must have all eigenvalues be negative). Thus, the 2-norm does not necessarily decrease at each time-step. For example, consider the initial state $\mathbf{x}[0] = \begin{bmatrix}0 & 1\end{bmatrix}'$, so that $\mathbf{x}[1] = \mathbf{A}\mathbf{x}[0] = \begin{bmatrix}1 & 0.5\end{bmatrix}'$. In this case we have $\|\mathbf{x}[0]\|_2^2 = 1$ and $\|\mathbf{x}[1]\|_2^2 = 1.25$, and thus the 2-norm has increased.

Since choosing the 2-norm does not seem to produce satisfactory results, we expand our focus to consider the square of the $\mathbf{P}$-norm of the state vector, for some positive definite matrix $\mathbf{P}$ that we will choose (i.e., $V(\mathbf{x}[k]) = \|\mathbf{x}[k]\|_{\mathbf{P}}^2$). The evolution of this quantity is given by

$$\begin{aligned} V(\mathbf{x}[k+1]) - V(\mathbf{x}[k]) &= \|\mathbf{x}[k+1]\|_{\mathbf{P}}^2 - \|\mathbf{x}[k]\|_{\mathbf{P}}^2 \\ &= \|\mathbf{A}\mathbf{x}[k]\|_{\mathbf{P}}^2 - \|\mathbf{x}[k]\|_{\mathbf{P}}^2 \\ &= \mathbf{x}[k]'\mathbf{A}'\mathbf{P}\mathbf{A}\mathbf{x}[k] - \mathbf{x}[k]'\mathbf{P}\mathbf{x}[k] \\ &= \mathbf{x}[k]'\left(\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P}\right)\mathbf{x}[k]. \end{aligned}$$

Now, if we can find a positive definite matrix $\mathbf{P}$ so that $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P}$ is negative definite, then we will have shown that the norm constantly decreases.

Example 6.2. Consider again the system from the previous example. We will try to find a positive definite matrix $\mathbf{P}$ so that

$$\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} \prec \mathbf{0};$$

if we choose $\mathbf{P} = \begin{bmatrix}9 & 6\\ 6 & 29\end{bmatrix}$, we find that

$$\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} = \begin{bmatrix} -6.75 & 0 \\ 0 & -6.75 \end{bmatrix},$$

which is negative definite. Thus, the $\mathbf{P}$-norm of the state constantly decreases.

The function $V(\mathbf{x}[k]) = \|\mathbf{x}[k]\|_{\mathbf{P}}^2$ is called a *Lyapunov Function* for the system, and $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} \prec 0$ is called the Lyapunov equation. If there exists a positive definite matrix $\mathbf{P}$ satisfying this inequality, then we know that $V(\mathbf{x}[k]) = \|\mathbf{x}[k]\|_{\mathbf{P}}^2$ converges to something; however, how do we know it converges to zero? For example, it could converge to some nonzero value, as shown in the following figure.

We will now argue that the Lyapunov function must converge to zero (i.e., the situation in the above figure cannot occur). First, note that if $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} \prec 0$, then all eigenvalues

of $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P}$ are strictly negative. Let $-\lambda$ be the eigenvalue that is closest to zero (λ is taken to be positive here), and let ϵ be any number smaller than λ. Next, note that the eigenvalues of $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} + \epsilon\mathbf{I}_n$ are simply the eigenvalues of $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P}$ shifted to the right by ϵ. Since $\epsilon < \lambda$, all eigenvalues of $\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} + \epsilon\mathbf{I}_n$ are still negative, i.e.,

$$\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} + \epsilon\mathbf{I}_n \prec \mathbf{0},$$

or equivalently,

$$\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} \prec -\epsilon\mathbf{I}_n. \tag{6.3}$$

Next, suppose that $V(\mathbf{x}[k])$ converges to some value $c > 0$, and note that $c < V(\mathbf{x}[0])$ (since the Lyapunov function decreased at each time-step. Define the set

$$\mathcal{X} = \{\mathbf{a} \in \mathbb{R}^n \mid c \leq V(\mathbf{a}) \leq V(\mathbf{x}[0])\}.$$

Note that the set $\mathcal{X}$ does not contain $\mathbf{a} = \mathbf{0}$ (since $c > 0$ and $V(\mathbf{0}) = 0$). From the set $\mathcal{X}$, choose any vector $\mathbf{a}$ such that $\mathbf{a}'\mathbf{a}$ is minimum over all vectors in $\mathcal{X}$, and denote this minimum value by δ. From (6.3), we have

$$\begin{aligned} V(\mathbf{x}[k+1]) - V(\mathbf{x}[k]) &= \mathbf{x}[k]'(\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P})\mathbf{x}[k] \\ &< -\epsilon\mathbf{x}[k]'\mathbf{x}[k]. \end{aligned}$$

Since $c \leq V(\mathbf{x}[k]) \leq V(\mathbf{x}[0])$, it must be the case that $\mathbf{x}[k] \in \mathcal{X}$, and thus $-\epsilon\mathbf{x}[k]'\mathbf{x}[k] \leq -\epsilon\delta < 0$. Next, note that

$$\begin{aligned} V(\mathbf{x}[k]) - V(\mathbf{x}[0]) = &\underbrace{V(\mathbf{x}[k]) - V(\mathbf{x}[k-1])}_{<-\epsilon\delta} + \underbrace{V(\mathbf{x}[k-1]) - V(\mathbf{x}[k-2])}_{<-\epsilon\delta} \\ &+ \underbrace{V(\mathbf{x}[k-2]) - V(\mathbf{x}[k-3])}_{<-\epsilon\delta} + \cdots + \underbrace{V(\mathbf{x}[1]) - V(\mathbf{x}[0])}_{<-\epsilon\delta} \\ &< -\epsilon\delta k, \end{aligned}$$

or equivalently, $V(\mathbf{x}[k]) < V(\mathbf{x}[0]) - \epsilon\delta k$. This indicates that for $k > \frac{V(\mathbf{x}[0])}{\epsilon\delta}$, we would have $V(\mathbf{x}[k]) < 0$ which is not possible (since $V(\cdot)$ is always nonnegative). Thus, it must be the case that $\delta = 0$, which indicates that the set $\mathcal{X}$ must contain the zero vector, which contradicts the assumption that $c > 0$. Thus, the Lyapunov function must converge to zero. While the above analysis shows that considering a norm of this form is *sufficient* to show stability, it turns out that the $\mathbf{P}$-norm is also necessary (we will not go over the proof of this here; see [36]). This leads to the following result.

Theorem 6.2. *The system* $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$ *is stable if and only if there exists a positive definite matrix* $\mathbf{P}$ *such that*

$$\mathbf{A}'\mathbf{P}\mathbf{A} - \mathbf{P} \prec 0.$$

Equivalently, there exists a positive definite matrix satisfying the above inequality if and only if all eigenvalues of $\mathbf{A}$ *are inside the unit circle.*

6.2.2 Mean Square Stability of Linear Systems with Packet Drops

With the tools from Lyapunov theory in hand, we will return to the system (6.2) and reconsider what happens when there are packet drops in the channel between the sensors and the controller. Specifically, we will consider a setup of the form shown below.

The output $\mathbf{y}[k]$ of the plant is sent via the channel to the controller C; the packet is dropped with probability p, and delivered correctly with probability $1-p$. For the controller, we will consider observer-feedback of the form

$$\hat{\mathbf{x}}[k+1] = \begin{cases} \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k]) & \text{with probability } 1-p \\ \mathbf{A}\hat{\mathbf{x}}[k] + \mathbf{B}\mathbf{u}[k] & \text{with probability } p \end{cases},$$
$$\mathbf{u}[k] = -\mathbf{K}\hat{\mathbf{x}}[k].$$

That is, when the state estimator receives a packet containing the output $\mathbf{y}[k]$, it incorporates it into its update via the corrective term $\mathbf{L}(\mathbf{y}[k] - \mathbf{C}\hat{\mathbf{x}}[k] - \mathbf{D}\mathbf{u}[k])$. However, when the state-estimator does not receive a packet at time-step k, it simply updates its estimate of the state according to the model of the system, without incorporating the corrective term. The evolution of the estimation error $\mathbf{e}[k] = \mathbf{x}[k] - \hat{\mathbf{x}}[k]$ is given by

$$\mathbf{e}[k+1] = \begin{cases} (\mathbf{A} - \mathbf{L}\mathbf{C})\mathbf{e}[k] & \text{with probability } 1-p \\ \mathbf{A}\mathbf{e}[k] & \text{with probability } p \end{cases}. \tag{6.4}$$

We would like to show that the estimation error is mean square stable (which means that the state estimator provides an accurate estimate of the system state, on average). To do this, we would like to show that the mean squared estimation error $E[\|\mathbf{e}[k]\|_2^2]$ is bounded (or goes to zero). From Section A.4.7 in Appendix A, recall that for any positive definite matrix $\mathbf{P}$, we have $\|\mathbf{e}[k]\|_2^2 \leq (\lambda_{\min}(\mathbf{P}))^{-1}\|\mathbf{e}[k]\|_{\mathbf{P}}^2$, where $\lambda_{\min}(\mathbf{P})$ is the smallest eigenvalue of $\mathbf{P}$. Thus, if we can show that $E[\|\mathbf{e}[k]\|_{\mathbf{P}}^2]$ goes to zero, we will have proved that the estimation error is mean square stable.

To this end, for system (6.4), we will examine

$$\begin{aligned} \sigma_{k+1} &\triangleq E[\|\mathbf{e}[k+1]\|_{\mathbf{P}}^2] \\ &= E\left[E[\|\mathbf{e}[k+1]\|_{\mathbf{P}}^2 \mid \mathbf{e}[k]]\right] \\ &= E[\|(\mathbf{A} - \mathbf{L}\mathbf{C})\mathbf{e}[k]\|_{\mathbf{P}}^2(1-p) + \|\mathbf{A}\mathbf{e}[k]\|_{\mathbf{P}}^2 p]. \end{aligned} \tag{6.5}$$

Now, examine the quantity

$$\|(\mathbf{A}-\mathbf{LC})\mathbf{e}[k]\|^2_{\mathbf{P}}(1-p)+\|\mathbf{Ae}[k]\|^2_{\mathbf{P}}p = \\ \mathbf{e}'[k]\left((\mathbf{A}-\mathbf{LC})'\mathbf{P}(\mathbf{A}-\mathbf{LC})(1-p)+\mathbf{A}'\mathbf{PA}p\right)\mathbf{e}[k].$$

If we can choose the positive definite matrix $\mathbf{P}$ and matrix $\mathbf{L}$ to satisfy

$$(\mathbf{A}-\mathbf{LC})'\mathbf{P}(\mathbf{A}-\mathbf{LC})(1-p)+\mathbf{A}'\mathbf{PA}p \prec \mathbf{P},$$

then we would obtain $E[\|(\mathbf{A}-\mathbf{LC})\mathbf{e}[k]\|^2_{\mathbf{P}}(1-p)+\|\mathbf{Ae}[k]\|^2_{\mathbf{P}}p] < E[\|\mathbf{e}[k]\|^2_{\mathbf{P}}]$, and (6.5) would become

$$\sigma_{k+1} < \sigma_k.$$

Using a similar argument to the one in the previous section, we have that $\sigma_k \to 0$, which proves mean square stability. This leads us to the following theorem.

Theorem 6.3. *Consider a system* $(\mathbf{A},\mathbf{B},\mathbf{C})$*, and let* p *be the probability that the state estimator receives the output* $\mathbf{y}[k]$ *over the channel at time-step* k*. If there exists a positive definite matrix* $\mathbf{P}$ *and a matrix* $\mathbf{L}$ *such that*

$$(\mathbf{A}-\mathbf{LC})'\mathbf{P}(\mathbf{A}-\mathbf{LC})(1-p)+\mathbf{A}'\mathbf{PA}p \prec \mathbf{P}, \qquad (6.6)$$

the estimation error is mean square stable.

When the matrix $\mathbf{C}$ is square and full rank, we can easily choose $\mathbf{L}=\mathbf{AC}^{-1}$, in which case (6.6) becomes

$$\mathbf{A}'\mathbf{PA}p-\mathbf{P} \prec \mathbf{0} \Leftrightarrow \sqrt{p}\mathbf{A}'\mathbf{PA}\sqrt{p}-\mathbf{P} \prec \mathbf{0}. \qquad (6.7)$$

From Theorem 6.2, we see that there is a matrix $\mathbf{P}$ satisfying this equation if and only if all eigenvalues of the matrix $\sqrt{p}\mathbf{A}$ have magnitude less than 1. This immediately leads to the following result.

Corollary 6.1. *Consider the system* (6.2) *with matrix* $\mathbf{C}$ *being square and full rank. Let* p *be the probability that the feedback loop drops a packet. Then, the state estimation error is mean square stable if and only if*

$$p|\lambda_{\max}(\mathbf{A})|^2 < 1 \ . \qquad (6.8)$$

where $\lambda_{\max}(\mathbf{A})$ *is the eigenvalue of* $\mathbf{A}$ *of largest magnitude.*

Note that this generalizes the condition found in Theorem 6.1 for scalar systems (in that case, we had $\mathbf{C} = 1$). However, for more general systems (i.e., when $\mathbf{C}$ is not invertible), it is harder to find an analytical characterization of the probability for which (6.6) holds. The main problem is that for any fixed p, both $\mathbf{L}$ and $\mathbf{P}$ are unknown in (6.6), and furthermore, these unknown matrices multiply each other. Thus, an inequality of the form (6.6) is known as a *bilinear matrix inequality*, since if we fix either $\mathbf{L}$ or $\mathbf{P}$ to some numerical value, the inequality is linear in the other unknown matrix. Bilinear matrix inequalities are very difficult to solve efficiently, and thus we cannot deal with (6.6) directly.

We can, however, transform (6.6) into a more convenient form by using the Schur Complement (Theorem A.12 in Section A.4.6). First, rearrange this inequality as

$$\begin{aligned}
&\mathbf{P} - (\mathbf{A} - \mathbf{LC})'\mathbf{P}(\mathbf{A} - \mathbf{LC})(1-p) + \mathbf{A}'\mathbf{PA}p \succ \mathbf{0} \\
\Leftrightarrow\ &\mathbf{P} - \begin{bmatrix}\sqrt{1-p}(\mathbf{A} - \mathbf{LC})' & \sqrt{p}\mathbf{A}'\end{bmatrix} \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix} \begin{bmatrix}\sqrt{1-p}(\mathbf{A} - \mathbf{LC}) \\ \sqrt{p}\mathbf{A}\end{bmatrix} \succ \mathbf{0} \\
\Leftrightarrow\ &\mathbf{P} - \begin{bmatrix}\sqrt{1-p}(\mathbf{A} - \mathbf{LC})' & \sqrt{p}\mathbf{A}'\end{bmatrix} \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix} \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix} \begin{bmatrix}\sqrt{1-p}(\mathbf{A} - \mathbf{LC}) \\ \sqrt{p}\mathbf{A}\end{bmatrix} \succ \mathbf{0}.
\end{aligned}$$

Applying the Schur Complement in Theorem A.12 with

$$\mathbf{Q} = \mathbf{P},\ \ \mathbf{S} = \begin{bmatrix}\sqrt{1-p}(\mathbf{A} - \mathbf{LC})' & \sqrt{p}\mathbf{A}'\end{bmatrix} \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix},\ \ \mathbf{R} = \begin{bmatrix}\mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\end{bmatrix},$$

we obtain

$$\mathbf{P} - (\mathbf{A} - \mathbf{LC})'\mathbf{P}(\mathbf{A} - \mathbf{LC})(1-p) + \mathbf{A}'\mathbf{PA}p \succ \mathbf{0} \Leftrightarrow \begin{bmatrix}\mathbf{P} & \sqrt{1-p}(\mathbf{A} - \mathbf{LC})'\mathbf{P} & \sqrt{p}\mathbf{A}'\mathbf{P} \\ \sqrt{1-p}\mathbf{P}(\mathbf{A} - \mathbf{LC}) & \mathbf{P} & \mathbf{0} \\ \sqrt{p}\mathbf{PA} & \mathbf{0} & \mathbf{P}\end{bmatrix} \succ 0.$$

While the above inequality is still bilinear, note that the bilinear terms are all of the form $\mathbf{PL}$. Since $\mathbf{P}$ is supposed to be positive definite, it will be invertible. Thus, we can define a new matrix $\mathbf{Y} = \mathbf{PL}$, in which case the above inequality becomes

$$\begin{bmatrix}\mathbf{P} & \sqrt{1-p}(\mathbf{A}'\mathbf{P} - \mathbf{C}'\mathbf{Y}') & \sqrt{p}\mathbf{A}'\mathbf{P} \\ \sqrt{1-p}(\mathbf{PA} - \mathbf{YC}) & \mathbf{P} & \mathbf{0} \\ \sqrt{p}\mathbf{PA} & \mathbf{0} & \mathbf{P}\end{bmatrix} \succ 0.$$

The above inequality is now *linear* in the unknown variables $\mathbf{P}$ and $\mathbf{Y}$ (assuming a fixed p), and naturally, is called a *Linear Matrix Inequality* (LMI). Such inequalities can be solved efficiently using convex optimization techniques, and software such as MATLAB provides built-in toolboxes for handling these problems [8]. Thus, to find the largest packet drop probability p for which mean square stability is possible, we use the following strategy.

1. Choose some value of p between 0 and 1.
2. Try to solve the LMI

$$\begin{bmatrix} \mathbf{P} & \sqrt{1-p}(\mathbf{A}'\mathbf{P} - \mathbf{C}'\mathbf{Y}') & \sqrt{p}\mathbf{A}'\mathbf{P} \\ \sqrt{1-p}(\mathbf{PA} - \mathbf{YC}) & \mathbf{P} & \mathbf{0} \\ \sqrt{p}\mathbf{PA} & \mathbf{0} & \mathbf{P} \end{bmatrix} \succ 0$$

   for $\mathbf{Y}$ and $\mathbf{P}$.
3. If a solution is found, mean square stability is possible for that value of p; increase p. If no solution exists, mean square stability is not possible for that value of p; decrease p. Return to step 2.

For any p that allows mean square stability, we can calculate the observer gain as $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$, once the LMI has been solved for that p.

Chapter 7

Information Dissemination in Distributed Systems and Networks

It has long been recognized that distributed and decentralized systems have certain desirable characteristics such as modularity, fault-tolerance, and simplicity of implementation. For these reasons, the distributed system paradigm is finding a foothold in an extraordinarily large number of applications (many of which are life- and mission-critical), ranging from transmitting patient diagnostic data in hospitals via multi-hop wireless networks [2], to coordinating teams of autonomous aircraft for search and rescue operations [70]. A key requirement in such systems and networks is for some or all of the nodes to calculate some function of certain parameters, or for some of the nodes to recover sequences of values sent by other nodes. For example, sink nodes in sensor networks may be tasked with calculating the average value of all available sensor measurements [24, 64, 10]. Another example is the case of multi-agent systems, where agents communicate with each other to coordinate their speed and direction [61, 67]. Yet another example is the case of transmitting streams of values in communication networks from a set of source nodes to a set of sink nodes [19, 49]. The problems of information dissemination and function calculation in networks have been studied by the computer science, communication, and control communities over the past few decades, leading to the development of various protocols and algorithms [44, 57, 4, 53, 25, 3, 16, 59, 17, 14]. A special case of the distributed function calculation problem is the *distributed consensus* problem, where all nodes in the network calculate the same function [57], and this topic has experienced a resurgence in the control community due to its applicability to tasks such as coordination of multi-agent systems, and its ability to model physical and biological behavior such as flocking (e.g., see [61, 67], and the references therein).

In this chapter, we will study the general problem of disseminating information and distributively calculating functions in networks. Specifically, we will analyze a particular

linear iterative strategy for information dissemination and use some of the linear system theory that we have covered in the previous chapters to show that such strategies are quite powerful: they can allow all nodes to calculate *arbitrary* and different functions of the initial values, even when there are some nodes in the network that behave in malicious and unpredictable ways. Before delving into the details of these strategies, we will start by considering the following simple example.

Example 7.1. Consider a simple sensor network with three sensors x_1, x_2 and x_3 arranged in a line as follows.

Suppose that each sensor is tasked with measuring the surrounding temperature, and then the three sensors are required to calculate the average temperature measurement across all sensors. In this example, x_1 measures the temperature to be 10, x_2 measures it to be 20, and x_3 measures it to be 30.

One way to calculate the average temperature would be for all sensors to send their values to a single sensor, who then calculates the average and sends it back to the other sensors. For example, sensors x_1 and x_3 would send their values to x_2, which calculates the average to be 20. This scheme is simple and efficient, but requires the sensors to decide who to select as the "leader", and then to arrange their transmissions in order to send their values to the leader.

As an alternative approach, suppose that each sensor simply *updates* its value to be an average of its own value and the values of its neighbor(s). For the values given above, sensor x_1 would update its value to be $0.5(10 + 20) = 15$, x_2 would update its value to be $\frac{1}{3}(10+20+30) = 20$ and x_3 would update its value to be $0.5(20+30) = 25$. Thus, we see that after doing this, all of the sensor values have come closer to the average value. If we have the sensors take the average again, their values would become $17.5, 20$ and 22.5, respectively. Continuing in this way, we see that all sensors asymptotically approach the average value in the network.

The benefit of the local averaging scheme is that none of the sensors need to know the network, or to elect a leader to average the values for them. However, there are various questions raised by this simple example. Is it true that the nodes will always converge to the same value if they simply take local averages? Will they converge to the average of the initial values? Can the nodes calculate the average (or other functions) in finite time via this scheme? What happens if some sensors are faulty or malicious? In the rest of this chapter, we will provide answers to these questions.

7.1 Distributed System Model

The distributed systems and networks that we will be studying in this chapter will be modeled as a directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of directed edges (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information from node x_j); see Appendix B for background and notation on graph theory. All nodes that can transmit information to node x_i are said to be neighbors of node x_i, and are represented by the set $\mathcal{N}_i$. The number of neighbors of node x_i is called the degree of node x_i, and is denoted as $\deg_i$. Similarly, the number of nodes that have node x_i as a neighbor is called the out-degree of node x_i, and is denoted as out-$\deg_i$.

We will deal with networks where information is disseminated via the *wireless broadcast* model, whereby each node sends the same information to all of its neighbors. This model, while obviously applicable to wireless networks, also holds when information is obtained by *direct sensing* (i.e., where each node measures or senses the values of its neighbor, as opposed to receiving that value through a transmission). We assume that every node in the network has an identifier (so that nodes can associate each piece of information that they sense or receive with the corresponding neighbor). Every node is also assumed to have some memory (so that they can store the information that they sense or receive from their neighbors), and sufficient computational capability to perform mathematical operations on this stored information (such as calculating the rank of a matrix, multiplying matrices, etc.). We will also generally assume that all communications between nodes are reliable (i.e., any transmission will be eventually received). However we do not assume any fixed time-limit on message delivery;[1] in other words, we assume that nodes in the network wait until they have received transmissions from all of their neighbors, and then execute their transmission or update strategies before waiting for the next transmissions from their neighbors. We will capture this behavior by referring to the behavior of a node at *time-step* k, by which we mean the k–th transmission or update step executed by that node. We assume that all messages are either delivered in the order they were transmitted, or have an associated time-stamp or sequence number to indicate the order of transmission.

7.2 Linear Iterative Strategies for Asymptotic Consensus

Suppose that each node x_i in the distributed system has some initial value, given by $x_i[0] \in \mathbb{R}$. The goal is for the nodes to calculate some function of $x_1[0], x_2[0], \ldots, x_n[0]$ by updating and/or exchanging their values at each time-step k, based on some distributed

[1] However, when we discuss networks with potentially faulty or malicious nodes, we will either assume that nodes always transmit a value (even if they are faulty), or we will assume that messages are delivered in a fixed amount of time. We will have to strengthen our assumptions in this way due to the fact it is impossible to perform fault-diagnosis without bounds on the time required to deliver messages – the receiving node will never be able to determine whether an expected message from another node is simply delayed, or if the transmitting node has failed [57].

strategy that adheres to the constraints imposed by the network topology. The scheme that we study in this chapter makes use of linear iterations; specifically, each node in the network repeatedly updates its own value to be a weighted linear combination of its previous value and those of its neighbors, which it then transmits. Mathematically, at each time-step, each node x_i updates and transmits its value as

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] \ , \tag{7.1}$$

where the w_{ij}'s are a set of weights from $\mathbb{R}$; for instance, these weights were chosen to be $\frac{1}{\deg_i + 1}$ in Example 7.1. Given the individual updates for each node, the values of all nodes at time-step k can be aggregated into the value vector $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_n[k] \end{bmatrix}'$, and the update strategy for the entire system can be represented as

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix}}_{\mathbf{W}} \mathbf{x}[k] \tag{7.2}$$

for $k \in \mathbb{N}$, where $w_{ij} = 0$ if $j \notin \mathcal{N}_i \cup \{x_i\}$. We will later generalize this model to include faulty or malicious updates by certain nodes.

Definition 7.1 (Calculable Function)**.** Let $g : \mathbb{R}^n \mapsto \mathbb{R}^r$ be a function of the initial values of the nodes (note that $g(\cdot)$ will be a vector-valued function if $r \geq 2$). We say $g(x_1[0], x_2[0], \ldots, x_n[0])$ is *calculable by node* x_i if it can be calculated by node x_i from some or all of the information that it receives from the linear iteration (7.2) over a sufficiently large number of time-steps. We call $g(\cdot)$ a *linear* function if $g(x_1[0], x_2[0], \ldots, x_n[0]) = \mathbf{Q}\mathbf{x}[0]$ for some $r \times n$ matrix $\mathbf{Q}$.

Definition 7.2 (Distributed Consensus)**.** The system is said to achieve distributed consensus if all nodes in the system calculate the value $g(x_1[0], \ldots, x_n[0])$ after running the linear iteration for a sufficiently large number of time-steps. When the field under consideration is $\mathbb{R}$ and

$$g(x_1[0], x_2[0], \ldots, x_n[0]) = \frac{1}{n}\mathbf{1}'\mathbf{x}[0] \ ,$$

where $\mathbf{1}$ is the vector of all 1's, the system is said to perform *distributed averaging* (i.e., the consensus value is the average of all initial node values).

Definition 7.3 (Asymptotic Consensus)**.** The system is said to reach asymptotic consensus if

$$\lim_{k \to \infty} x_i[k] = g(x_1[0], x_2[0], \ldots, x_n[0])$$

for every $i \in \{1, 2, \ldots, n\}$, where $g(x_1[0], x_2[0], \ldots, x_n[0]) \in \mathbb{R}$.

When $g(x_1[0], x_2[0], \ldots, x_n[0]) = \mathbf{c}'\mathbf{x}[0]$ for some vector $\mathbf{c}'$, the following result from [96] characterizes the conditions under which the iteration (7.2) achieves asymptotic consensus.

Theorem 7.1 ([96])**.** *The iteration given by* (7.2) *reaches asymptotic consensus on the linear function* $\mathbf{c}'\mathbf{x}[0]$ *(under the technical condition that* $\mathbf{c}$ *is normalized so that* $\mathbf{c}'\mathbf{1} = 1$*) if and only if the weight matrix* $\mathbf{W}$ *satisfies the following conditions:*

1. $\mathbf{W}$ *has a single eigenvalue at* 1*, with left eigenvector* $\mathbf{c}'$ *and right eigenvector* $\mathbf{1}$*.*
2. *All other eigenvalues of* $\mathbf{W}$ *have magnitude strictly less than* 1*.*

Proof. First, note that if $\mathbf{W}$ has an eigenvalue λ that has magnitude larger than 1 with corresponding eigenvector $\mathbf{v}$, then for any initial vector $\mathbf{x}[0] = \mathbf{v}$, the values of all nodes satisfy the property $\mathbf{x}[k] = \lambda^k \mathbf{v} \to \infty$. Thus, $\mathbf{W}$ cannot have eigenvalues with magnitude larger than 1 if asymptotic consensus is required.

Next, write $\mathbf{W}$ as

$$\mathbf{W} = \underbrace{\begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\end{bmatrix}}_{\mathbf{T}} \begin{bmatrix}\mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}\end{bmatrix} \underbrace{\begin{bmatrix}\bar{\mathbf{T}}_1 \\ \bar{\mathbf{T}}_2\end{bmatrix}}_{\mathbf{T}^{-1}},$$

where $\mathbf{Z}$ is a matrix that contains all of the eigenvalues at 1, and $\mathbf{S}$ is a matrix whose eigenvalues are all stable. The matrix $\mathbf{T}_1$ contains all eigenvectors corresponding to the eigenvalues at 1, and $\mathbf{T}_2$ contains the eigenvectors corresponding to the stable eigenvalues. Similarly, the matrix $\bar{\mathbf{T}}_1$ contains all left eigenvectors corresponding to the eigenvalues at 1, and $\bar{\mathbf{T}}_2$ contains all left eigenvectors corresponding to the stable eigenvalues. From this decomposition, note that

$$\mathbf{x}[k] = \begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\end{bmatrix} \begin{bmatrix}\mathbf{Z}^k & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^k\end{bmatrix} \begin{bmatrix}\bar{\mathbf{T}}_1 \\ \bar{\mathbf{T}}_2\end{bmatrix} \mathbf{x}[0] \to \begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\end{bmatrix} \begin{bmatrix}\mathbf{Z}^k & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix} \begin{bmatrix}\bar{\mathbf{T}}_1 \\ \bar{\mathbf{T}}_2\end{bmatrix} \mathbf{x}[0] = \mathbf{T}_1 \mathbf{Z}^k \bar{\mathbf{T}}_1 \mathbf{x}[0].$$

Now, if we want to guarantee that $\mathbf{x}[k]$ converges to $\mathbf{1}\mathbf{c}'\mathbf{x}[0]$, we should ensure that $\mathbf{T}_1 = \mathbf{1}$ and $\bar{\mathbf{T}}_1 = \mathbf{c}'$, which means that $\mathbf{Z} = 1$. Thus, asymptotic consensus on the function $\mathbf{c}'\mathbf{x}[0]$ is achieved if and only if all of the conditions in the theorem are satisfied. □

Note that the rate at which the nodes reach consensus is given by how quickly the term $\mathbf{S}^k$ drops to zero. If λ_2 denotes the eigenvalue of largest magnitude in $\mathbf{S}$ (which would make it the eigenvalue of second largest magnitude in $\mathbf{W}$), then $\mathbf{S}^k$ goes to zero approximately as λ_2^k. Thus, the smaller the second largest eigenvalue of the matrix, the faster consensus is reached (in terms of rate). The largest eigenvalue determines whether consensus is even possible, and is given the following name.

Definition 7.4. For any square matrix $\mathbf{A}$, the magnitude of the largest eigenvalue is denoted by $\rho(\mathbf{A})$, and is called the *spectral radius* of the matrix.

There is a large amount of literature dealing with choosing the weights in (7.2) so that the system reaches asymptotic consensus on a linear function [96, 61, 47, 15]. We will now describe one such method.

7.2.1 Choosing the Weight Matrix for Asymptotic Consensus

Mathematically, the first condition in Theorem 7.1 states that $\mathbf{W1} = \mathbf{1}$. In other words, the sum of the elements in each row of $\mathbf{W}$ must be equal to 1. One way to achieve this is to use the following concept.

Definition 7.5. Given a set of real numbers $x_1, x_2, \ldots, x_n$, the quantity

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$$

is said to be a *convex combination* of $x_1, x_2, \ldots, x_n$ if each of the scalars $\alpha_1, \alpha_2, \cdots, \alpha_n$ is nonnegative, and satisfy $\sum_{i=1}^{n} \alpha_i = 1$.

If we have each node update its value to be a convex combination of its own and neighbors' values, then we would have $w_{ij} \geq 0$ for all i, j, $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i \cup \{x_i\}$, and $\sum_{j=1}^{n} w_{ij} = 1$ for $i \in \{1, 2, \ldots, n\}$. This would ensure that $\mathbf{W}$ has an eigenvalue at 1 with eigenvector $\mathbf{1}$. Note that the scheme in Example 7.1 satisfied this condition, with $w_{ij} = \frac{1}{\deg_i + 1}$ if $x_j \in \mathcal{N}_i \cup \{x_i\}$.

To prove convergence, we have to show that all other eigenvalues of $\mathbf{W}$ are inside the unit circle. To do this, we will apply the Gersgorin Disc Theorem (see Theorem A.6 in Section A.4.4). Specifically, every diagonal element in $\mathbf{W}$ will be positive and less than 1. Furthermore, since each row sums to 1, we have

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} |w_{ij}| = 1 - w_{ii}.$$

Thus, all of the Gersgorin discs are contained within the unit circle, as illustrated by the following figure.

This indicates that $\mathbf{W}$ has no eigenvalues of magnitude larger than 1. Finally, we only need to show that $\mathbf{W}$ has a *single* eigenvalue at 1. To do this, we will use the following result.

Lemma 7.1. *If $\mathbf{A}$ is a square matrix with all entries positive, then $\mathbf{A}$ has only a single eigenvalue of largest magnitude.*

The proof of the above result requires a few other preliminary results, and so we will not cover it here. For details, see [39]. This lemma is promising, but our weight matrix $\mathbf{W}$ does not have all elements positive (some of them are zero, corresponding to the cases where there is no edge between two nodes). However, under a certain condition on the underlying network, we can indeed use this result.

Lemma 7.2. *Consider a network $\mathcal{G}$ with n nodes, and assume that $\mathcal{G}$ is strongly connected. If all edges and self-loops are assigned a positive weight so that each row sums to 1, then $\mathbf{W}$ has a single eigenvalue at 1.*

Proof. Choosing the weights as specified in the Lemma, the matrix $\mathbf{W}$ can be viewed as the *adjacency matrix* for the graph $\mathcal{G}$ (see Section B.2 in the Appendix). Specifically, Corollary B.1 indicates that there exists some $k \leq n-1$ such that all elements of $\mathbf{W}^k$ are positive. Next, note that the eigenvalues of $\mathbf{W}^k$ are the eigenvalues of $\mathbf{W}$ raised to the k–th power. However, Lemma 7.1 states that the largest eigenvalue of $\mathbf{W}^k$ only appears once; thus it must be the case that the eigenvalue 1 appears only once in $\mathbf{W}$. □

Using the above lemma and the Gersgorin Disc Theorem, we come to the following result.

Theorem 7.2. *Consider a network $\mathcal{G}$ with n nodes, and assume that $\mathcal{G}$ is strongly connected. If all edges and self-loops are assigned a positive weight so that each row sums to 1, then all nodes in the network will reach asymptotic consensus on the value $\mathbf{c}'\mathbf{x}[0]$, where $\mathbf{c}'\mathbf{W} = \mathbf{c}'$.*

Proof. If the conditions in the theorem are satisfied, then the Gersgorin Disc Theorem and Lemma 7.2 indicate that all conditions in Theorem 7.1 will be satisfied. Specifically, $\mathbf{W}$ will have a single eigenvalue at 1, with right eigenvector $\mathbf{1}$. Furthermore, all other eigenvalues will be inside the unit circle. Finally, defining $\mathbf{c}'$ to be the left eigenvector of $\mathbf{W}$ corresponding to the eigenvalue 1, we obtain the desired result. □

Example 7.2. Let us return to the three node sensor network from Example 7.1, where each node updates its value to be an average of its previous value and those of its neighbors. The weight matrix for this iteration is given by

$$\mathbf{W} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0.5 & 0.5 \end{bmatrix}.$$

The network is strongly connected, and all edges and self-loops have a positive weight so that each row sums to 1. Thus, the iteration is guaranteed to converge to consensus. To find what value each node converges to, we find the left eigenvector of $\mathbf{W}$ corresponding to eigenvalue 1 to be

$$\mathbf{c}' = \begin{bmatrix} \frac{2}{7} & \frac{3}{7} & \frac{2}{7} \end{bmatrix}.$$

Thus, each node will converge to $\frac{2}{7}x_1[0]+\frac{3}{7}x_2[0]+\frac{2}{7}x_3[0]$; note that this is *not* guaranteed to be the average of the initial values (which would have been $\frac{1}{3}(x_1[0] + x_2[0] + x_3[0])$. The reason for all nodes converging to the average in Example 7.1 is that the initial values were such that $\frac{2}{7}x_1[0] + \frac{3}{7}x_2[0] + \frac{2}{7}x_3[0] = \frac{1}{3}(x_1[0] + x_2[0] + x_3[0])$.

The above example shows that simple local averaging may not lead to consensus to the average of all initial values, because the left eigenvector $\mathbf{c}'$ of $\mathbf{W}$ corresponding to eigenvalue 1 may not be $\frac{1}{n}\mathbf{1}'$. One can guarantee convergence to the average if the sum of all the entries in each *column* is also equal to 1 (so that $\frac{1}{n}\mathbf{1}'$ is a left eigenvector of $\mathbf{W}$). To do this, suppose that the graph is undirected (i.e., the links between each node are bidirectional), and suppose that every node knows n. Then, if each node assigns $w_{ij} = \frac{1}{n}$ if $x_j \in \mathcal{N}_i$, $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i$, and $w_{ii} = 1 - \frac{\deg_i}{n}$, one can verify that every column does in fact sum to 1.

Example 7.3. In the three node network from Example 7.1, if every node applies the $\frac{1}{n}$ weights specified above, we obtain

$$\mathbf{W} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

For this matrix, we have $\mathbf{W}\mathbf{1} = \mathbf{1}$ and $\frac{1}{n}\mathbf{1}'\mathbf{W} = \frac{1}{n}\mathbf{1}'$, and thus all nodes will converge to the average of any initial values.

7.2.2 Notes on Terminology

The notion of convergence to a common equilibrium value is heavily related to the concept of a *Markov Chain* from probability theory. In that community, nonnegative matrices

where each row sums to 1 are called *stochastic* matrices. Furthermore, if the underlying graph is strongly connected, the matrix is called *irreducible*. If $\mathbf{W}$ is a nonnegative matrix and $\mathbf{W}^k$ is a positive matrix for some k, then $\mathbf{W}$ is called a *primitive* matrix. As shown in Corollary B.1, an irreducible matrix will be primitive if every diagonal element is nonzero; however irreducibility by itself does not imply primitivity (as shown in Example B.4). A stochastic matrix $\mathbf{W}$ that converges to $\mathbf{1c}'$ for some $\mathbf{c}'$ is called an *ergodic* matrix; as shown by the proof of Theorem 7.2, any primitive stochastic matrix is ergodic. However, not all ergodic matrices need to be primitive or irreducible. For example, consider the stochastic matrix

$$\mathbf{W} = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 1 \end{bmatrix}.$$

This matrix is upper triangular, and thus $\mathbf{W}^k$ will also be upper triangular for any k; therefore, the $(2,1)$ element will always be zero, and this matrix is not primitive. Furthermore, there is no path from the first node to the second node, so this matrix is not irreducible. However, this matrix has only one eigenvalue at 1 (and the other is at 0.5), and thus this matrix is guaranteed to converge to $\mathbf{1c}'$, where $\mathbf{c}'$ is the left eigenvector of $\mathbf{W}$ corresponding to eigenvalue 1 (according to Theorem 7.1). Specifically, $\mathbf{c}' = \begin{bmatrix} 0 & 1 \end{bmatrix}$, and one can verify that

$$\lim_{k\to\infty} \mathbf{W}^k = \mathbf{1c}' = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

7.3 Linear Iterative Strategies for Secure Data Aggregation

In the last section, we focused on using linear iterative strategies for asymptotic consensus in connected networks. In this section, we investigate the use of such strategies to treat the problem of data accumulation in networks (where some or all nodes have to gather *all* of the other values). Furthermore, we allow for the possibility that some nodes in the network update their values at each time-step in an arbitrary manner, instead of following the predefined strategy of using a specific weighted linear combination of their neighbors' (and own) values. Such arbitrary updates can occur, for example, if some nodes in the network are compromised by a malicious attacker whose objective is to disrupt the operation of the network [57, 38]. Alternatively, these arbitrary updates might be the result of hardware malfunctions at the nodes, which cause them to incorrectly calculate their update value [29]. Here, we will see that the graph connectivity is a determining factor for the ability of linear iterative strategies to tolerate malicious (or faulty) agents (see Section B.1 for a review of graph connectivity). In particular, if the connectivity of the graph is $2f$ or less, then regardless of the choice of weights in the linear iteration, it is possible to find a subset of f (or less) nodes that can conspire to prevent some nodes from calculating any function of all node values. On the other hand, if the network topology has connectivity at least $2f+1$, then any node can work around the misbehaving nodes to calculate any desired function of the initial node values, even if up to f nodes conspire and coordinate their actions in an attempt to disrupt the network. To prove these results,

we will exploit concepts from classical linear system theory (such as structured system theory, strong observability, and invariant zeros of linear systems).

In our development, we use $\mathbf{e}_i$ to denote the column vector with a 1 in its i'th position and 0's elsewhere. We will also denote the cardinality (number of elements) of a set $\mathcal{S}$ by $|\mathcal{S}|$, and for a pair of sets $\mathcal{S}$ and $\mathcal{T}$, we will use $\mathcal{S} \setminus \mathcal{T}$ to denote the set of elements of $\mathcal{S}$ that are not in $\mathcal{T}$.

7.3.1 Attacking the Network

To start, it is useful to understand some fundamental limitations in the ability of a network to tolerate malicious nodes. First, as an extreme, note that if *all* nodes are malicious, there is certainly nothing that we can do. More generally, one can imagine that if a "large" number of nodes are malicious, we would still be incapable of overcoming them. To quantify just how many malicious nodes are required to prevent data accumulation or function calculation in networks, we will study the following examples.

Example 7.4. Consider the network with three nodes from Example 7.1.

Suppose that we are going to run some algorithm A in this network so that node x_1 can calculate some function of all initial values; for instance, A could be the linear strategy that we have been studying. The algorithm is supposed to provide x_1 with enough information about x_2 and x_3's values to be able to calculate the function. However, all information that x_1 receives about x_3 comes through x_2. However, suppose x_2 is malicious and decides to pretend that x_3's value is something that it is not; x_2 will run the algorithm A with this false value of x_3. In this case, x_1 would never be able to tell that x_2 is lying, because it has no other way to receive information about x_3's value, and thus it has no choice but to trust x_2. Thus, x_2 can prevent x_1 from correctly calculating its desired function.

Example 7.5. Consider the following network with four nodes.

Once again, x_1 wants to calculate some function of all initial values via some algorithm A. However, suppose now that node x_3 is malicious, and will pretend that node x_4's value is something that it is not. However, node x_2 will run correctly, and use x_4's true value in the algorithm. In this case, the only way that information about x_4 gets to node x_1 is through x_2 and x_3. However, due to x_3's duplicitous behavior, the information that

x_1 receives about x_4 will be inconsistent. Furthermore, node x_1 has no *a priori* reason to believe x_2 over x_3, or vice versa. Thus, x_3 can prevent x_1 from ever finding the true value of x_4.

Both of the above examples demonstrate that networks can be disrupted by a small and carefully chosen set of malicious nodes. The key observation is that the malicious nodes acted as a gateway for information between two sets of non-malicious nodes. This can be formally captured by the notion of a *vertex cut* in the graph; recall that a vertex cut is a set of nodes that removes all paths from one group of nodes to another group of nodes (see Section B.1). Using the intuition from the above examples, we can state the following result.

Theorem 7.3. *For a given network $\mathcal{G}$, let κ_{ij} denote the size of the smallest (i,j)-cut between nodes x_i and x_j. If $\kappa_{ij} \leq 2f$, then there is a choice of f malicious nodes that can prevent node x_j from calculating any function of x_i's value, regardless of the algorithm that is used in the network.*

Proof. Given any nodes x_i and x_j, let S_{ij} denote the (i,j)-cut with size κ_{ij}. For the purposes of this proof, assume that x_i is not a neighbor of x_j (because otherwise, there is no vertex cut that removes all paths from x_i to x_j). All information about x_i must go through the nodes in S_{ij} to reach x_j. Suppose that half of the nodes in S_{ij} are malicious, and decide to run the algorithm pretending that x_i's value is something that it is not. Now the information that x_j receives from the algorithm is inconsistent, and x_j would not know which half of the vertex cut to trust. Since the cut was chosen to have size $\kappa_{ij} \leq 2f$, there are at most f malicious nodes that are required in order to disrupt the network in this way. □

7.3.2 Modeling Malicious Nodes in the Linear Iterative Strategy

The previous section showed that the connectivity of the network must be at least $2f+1$ in order to tolerate any f malicious nodes, regardless of the algorithm. We now look at the question: can the *linear iterative strategy* be used to overcome f malicious nodes in networks of connectivity $2f+1$? We will answer this question in the affirmative, but first, we need to mathematically model malicious behavior in the context of the linear strategy.

To this end, suppose that instead of applying the update equation (7.1), some node x_l updates its value at each time-step as

$$x_l[k+1] = w_{ll}x_l[k] + \sum_{j \in \mathcal{N}_l} w_{lj}x_j[k] + u_l[k] \ , \tag{7.3}$$

where $u_l[k]$ is an additive error at time-step k.

Definition 7.6. Suppose all nodes run the linear iteration for T time-steps in order to perform function calculation. Node x_l is said to be *malicious* (or *faulty*) if $u_l[k]$ is nonzero for at least one time-step k, $0 \leq k \leq T-1$.

Note that the model for malicious nodes considered here is quite general, and allows node x_l to update its value in a completely arbitrary manner (via appropriate choices of $u_l[k]$ at each time-step). As such, this model encapsulates a wide variety of malicious behavior (including a conspiracy by a set of malicious nodes). It is worth noting that we do not treat the case where nodes try to influence the result of the computation by modifying their initial values (because of the philosophically different nature of this problem). We will show that our protocol allows every node to obtain every initial value (maliciously modified or not), and so one can potentially handle corrupted initial values by having each node use some sort of statistical analysis to detect and discard suspicious initial values. This is also a non-issue when the desired function only involves initial values from some trusted nodes, but intermediate (routing) nodes can be malicious.

Let $\mathcal{F} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_f}\}$ denote the set of nodes that are malicious during a run of the linear iteration. Using (7.2) and (7.3), the linear iteration can then be modeled as

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] + \underbrace{\begin{bmatrix}\mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \cdots & \mathbf{e}_{i_f}\end{bmatrix}}_{\mathbf{B}_{\mathcal{F}}} \underbrace{\begin{bmatrix} u_{i_1}[k] \\ u_{i_2}[k] \\ \vdots \\ u_{i_f}[k] \end{bmatrix}}_{\mathbf{u}_{\mathcal{F}}[k]} \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k], \;\; 1 \leq i \leq n \;\; , \end{aligned} \tag{7.4}$$

where $\mathbf{y}_i[k]$ represents the outputs (node values) seen by node i during time-step k of the linear iteration. Specifically, $\mathbf{C}_i$ is a $(\deg_i + 1) \times n$ matrix with a single 1 in each row capturing the positions of the state-vector $\mathbf{x}[k]$ that are available to node x_i (these positions correspond to neighbors of node x_i, along with node x_i itself). Recall that $\mathbf{e}_l$ denotes a unit vector with a single nonzero entry with value 1 at its l'th position.

Example 7.6. Consider the three node network from Example 7.1. Since node x_1 receives its own value and x_2's value at each time-step, we can write

$$\mathbf{y}_1[k] = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{C}_1} \mathbf{x}[k].$$

Note that (7.4) is a *linear system* with unknown inputs (since x_i does not know the faults caused by the malicious nodes). However, there is one other aspect of this system that is unknown to x_i: the set of malicious nodes $\mathcal{F}$, and thus the exact matrix $\mathbf{B}_{\mathcal{F}}$. If this matrix were known, recovering all initial values via the linear iteration would reduce to the problem of strong observability of the above system (see Section 2.6.4). As we will see later, the fact that we do not know $\mathbf{B}_{\mathcal{F}}$ will require us to consider strong observability of a slightly different system in order to allow x_i to recover all initial values.

As in Section 2.6.2, the set of all values seen by node i during the first $L+1$ time-steps of the linear iteration is given by

$$\underbrace{\begin{bmatrix}\mathbf{y}_i[0]\\ \mathbf{y}_i[1]\\ \mathbf{y}_i[2]\\ \vdots\\ \mathbf{y}_i[L]\end{bmatrix}}_{\mathbf{y}_i[0:L]} = \underbrace{\begin{bmatrix}\mathbf{C}_i\\ \mathbf{C}_i\mathbf{W}\\ \mathbf{C}_i\mathbf{W}^2\\ \vdots\\ \mathbf{C}_i\mathbf{W}^L\end{bmatrix}}_{\mathcal{O}_{i,L}} \mathbf{x}[0] + \underbrace{\begin{bmatrix}0 & 0 & \cdots & 0\\ \mathbf{C}_i\mathbf{B}_{\mathcal{F}} & 0 & \cdots & 0\\ \mathbf{C}_i\mathbf{W}\mathbf{B}_{\mathcal{F}} & \mathbf{C}_i\mathbf{B}_{\mathcal{F}} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{C}_i\mathbf{W}^{L-1}\mathbf{B}_{\mathcal{F}} & \mathbf{C}_i\mathbf{W}^{L-2}\mathbf{B}_{\mathcal{F}} & \cdots & \mathbf{C}_i\mathbf{B}_{\mathcal{F}}\end{bmatrix}}_{\mathcal{J}_{i,L}^{\mathcal{F}}} \underbrace{\begin{bmatrix}\mathbf{u}_{\mathcal{F}}[0]\\ \mathbf{u}_{\mathcal{F}}[1]\\ \mathbf{u}_{\mathcal{F}}[2]\\ \vdots\\ \mathbf{u}_{\mathcal{F}}[L-1]\end{bmatrix}}_{\mathbf{u}_{\mathcal{F}}[0:L-1]}. \tag{7.5}$$

The matrices $\mathcal{O}_{i,L}$ and $\mathcal{J}_{i,L}^{\mathcal{F}}$ characterize the ability of node x_i to calculate the required function of the initial values. The matrix $\mathcal{O}_{i,L}$ is the *observability matrix* for the pair $(\mathbf{W}, \mathbf{C}_i)$, and $\mathcal{J}_{i,L}^{\mathcal{F}}$ is the *invertibility matrix* for the system $(\mathbf{W}, \mathbf{B}_{\mathcal{F}}, \mathbf{C}_i)$.

7.3.3 Calculating Functions in the Presence of Malicious Nodes when $\kappa \geq 2f+1$

First, suppose that there are at most f malicious nodes in the network, given by the set $\mathcal{F}$. From the perspective of any node x_i, the linear iteration has the form of the linear system in (7.4). As explained in Section 7.3.2, if $\mathcal{F}$ were known, this would simply be a problem of strong observability of the system. However, since $\mathcal{F}$ is unknown, a little more analysis is needed. Essentially, we would like to be able to uniquely identify the initial state of the system (7.4) from the outputs, regardless of the set $\mathcal{F}$ of f malicious nodes. In other words, we want to ensure that two different initial states and two different sets of f malicious nodes cannot produce the same outputs for all time. With this intuition, the following theorem presents the condition required for recovering the initial values despite any unknown set of f malicious nodes.

Theorem 7.4. *Suppose there exists a weight matrix* $\mathbf{W}$ *(with* $w_{ij} = 0$ *if* $x_j \notin \mathcal{N}_i \cup \{x_i\}$*) such that for any set* S *of* $2f$ *nodes, the system*

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] + \mathbf{B}_S\mathbf{u}_S[k]\\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k]\end{aligned}$$

is strongly observable. Then node x_i *can uniquely recover all initial values after at most* N *time-steps of the linear iteration, despite the actions of any unknown set of* f *malicious nodes.*

Note that the above theorem requires us to consider strong observability of a system with $2f$ nodes in order to overcome f malicious nodes – this is the price paid for not knowing the set of malicious nodes *a priori*.

Proof. We will show that if the condition in the theorem is satisfied, then the output of (7.4) can be generated by only one initial condition. We will prove by contradiction:

suppose that there is an initial condition $\bar{\mathbf{x}}[0] \neq \mathbf{x}[0]$ and some set $\mathcal{F}_c$ of f malicious nodes (perhaps different from $\mathcal{F}$) that causes the linear iteration to produce the same output values as the original linear iteration for all time. The linear iteration for this different configuration is given by

$$\begin{aligned}\bar{\mathbf{x}}[k+1] &= \mathbf{W}\bar{\mathbf{x}}[k] + \mathbf{B}_{\mathcal{F}_c}\mathbf{u}_{\mathcal{F}_c}[k] \\ \mathbf{y}_i[k] &= \mathbf{C}_i\bar{\mathbf{x}}[k].\end{aligned}$$

Since, by assumption, the output of this system is the same as the output of (7.4) for all k, we have

$$\mathbf{C}_i\bar{\mathbf{x}}[k] = \mathbf{C}_i\mathbf{x}[k] \Leftrightarrow \mathbf{C}_i(\mathbf{x}[k] - \bar{\mathbf{x}}[k]) = \mathbf{0} \ \ \forall k \in \mathbb{N}. \tag{7.6}$$

Define $\hat{\mathbf{x}}[k] = \mathbf{x}[k] - \bar{\mathbf{x}}[k]$, so that

$$\begin{aligned}\hat{\mathbf{x}}[k+1] &= \mathbf{W}\mathbf{x}[k] + \mathbf{B}_{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[k] - \mathbf{W}\bar{\mathbf{x}}[k] - \mathbf{B}_{\mathcal{F}_c}[k] \\ &= \mathbf{W}\hat{\mathbf{x}}[k] + \underbrace{\begin{bmatrix}\mathbf{B}_{\mathcal{F}} & \mathbf{B}_{\mathcal{F}_c}\end{bmatrix}}_{\mathbf{B}_{\mathcal{F}\cup\mathcal{F}_c}} \underbrace{\begin{bmatrix}\mathbf{u}_{\mathcal{F}}[k] \\ -\mathbf{u}_{\mathcal{F}_c}[k]\end{bmatrix}}_{\mathbf{u}_{\mathcal{F}\cup\mathcal{F}_c}[k]}.\end{aligned} \tag{7.7}$$

This defines a linear system with initial condition $\hat{\mathbf{x}}[0] \neq \mathbf{0}$ (since $\mathbf{x}[0] \neq \bar{\mathbf{x}}[0]$). However, (7.6) indicates that $\mathbf{C}_i\hat{\mathbf{x}}[k] = \mathbf{0}$ for all $k \in \mathbb{N}$, which indicates that the system $(\mathbf{W}, \mathbf{B}_{\mathcal{F}\cup\mathcal{F}_c}, \mathbf{C}_i, \mathbf{0})$ is not strongly observable. However, since each of $\mathcal{F}$ and $\mathcal{F}_c$ have at most f nodes each, $\mathcal{F} \cup \mathcal{F}_c$ has at most $2f$ nodes; thus, we reach a contradiction, since the assumption in the theorem is that the system with any set of $2f$ nodes is strongly observable.

Thus, if the system $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ is strongly observable for any set S of $2f$ nodes, the outputs of the linear iteration will uniquely specify the initial values despite the actions of any f malicious nodes. Furthermore, since strong observability implies that the outputs of the system over N time-steps are sufficient to recover the initial state, we see that node x_i can obtain all initial values after running the linear iteration for at most N time-steps. □

The above result says that if the system $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ is strongly observable for any set S of $2f$ nodes, node x_i can recover $\mathbf{x}[0]$ after viewing its neighbors' (and own) values over at most N time-steps, despite the actions of any set $\mathcal{F}$ of f malicious nodes. To see exactly how x_i can do this, note that the set of all values seen by x_i over N time-steps is given by

$$\mathbf{y}_i[0:N] = \mathcal{O}_{i,N}\mathbf{x}[0] + \mathcal{J}_{i,N}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:N-1].$$

Node x_i does not know $\mathbf{x}[0]$, $\mathbf{u}_{\mathcal{F}}[0:N-1]$ and the set $\mathcal{F}$; Suppose now that x_i simply tries to find a set $\mathcal{F}_c$ of f nodes such that $\mathbf{y}_i[0:L]$ is in the range of the matrix $\begin{bmatrix}\mathcal{O}_{i,L} & \mathcal{J}_{i,L}^{\mathcal{F}_c}\end{bmatrix}$, i.e.,

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\bar{\mathbf{x}}[0] + \mathcal{J}_{i,L}^{\mathcal{F}_c}\bar{\mathbf{u}}$$

for some vectors $\bar{\mathbf{x}}[0]$ and $\bar{\mathbf{u}}$. Since the output of the system can only have been generated by one particular state, the only way for x_i to write its values in this way is if $\bar{\mathbf{x}}[0] = \mathbf{x}[0]$

– thus, x_i has recovered all initial values. Note that checking for all $\binom{N}{f}$ possibilities when trying to determine the possible set $\mathcal{F}_c$ of malicious nodes is equivalent to the brute force method of determining up to f errors in an N-dimensional real number codeword with distance $2f+1$. In the coding theory literature, there exist efficient ways of performing this check for both structured and random real number codes (e.g., see [86] and the references therein), and one can potentially exploit those results to streamline the above decoding procedure. This is an open problem for research.

Next, we will show that when $\kappa \geq 2f+1$, one can find a weight matrix $\mathbf{W}$ so that $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ is strongly observable for any set S of up to $2f$ nodes.

Invariant Zeros

Recall from Section 2.9.3 that for any set $S \subset \mathcal{X}$ of $2f$ nodes, the matrix

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{W} - z\mathbf{I}_n & \mathbf{B}_S \\ \mathbf{C}_i & \mathbf{0} \end{bmatrix}$$

is called the *matrix pencil* for the system $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$. Furthermore, any $z \in \mathbb{C}$ for which

$$\text{rank}\left(\mathbf{P}(z)\right) < n + 2f$$

is called an *invariant zero* of the system, and the system $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ will be strongly observable if and only if it has no invariant zeros. We will now focus on choosing $\mathbf{W}$ (with the required sparsity constraints) so that the system has no invariant zeros for any set $S \subseteq \mathcal{X}$ of $2f$ nodes. Note that the assumption that the number of outputs is larger than the number of inputs will be trivially satisfied in networks that have $(2f+1)$-connectivity, because each node i will have at least $2f+1$ neighbors, and thus C_i will have at least $2f+2$ rows.

Let $S = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{2f}}\}$ denote any set of $2f$ nodes, and let $\bar{S} = \mathcal{X} \setminus S$. Define $\mathbf{B}_S = \begin{bmatrix} \mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \cdots & \mathbf{e}_{i_{2f}} \end{bmatrix}$. For any choice of weights for the linear iteration, let $\mathbf{W}_{\bar{S}}$ represent the weight matrix corresponding to interconnections within the set $\bar{S}$, $\mathbf{W}_{S,\bar{S}}$ represent the weight matrix corresponding to connections from the set S to the set $\bar{S}$, $\mathbf{W}_S$ represent the weight matrix corresponding to interconnections within the set S, and $\mathbf{W}_{\bar{S},S}$ represent the weight matrix corresponding to connections from the set $\bar{S}$ to the set S. Note that $\mathbf{W}_{\bar{S}}$ has dimension $(n-2f) \times (n-2f)$, and $\mathbf{W}_S$ has dimension $(2f) \times (2f)$. Furthermore, for any node x_i, let $\begin{bmatrix} \mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S} \end{bmatrix}$ denote a $(\deg_i + 1) \times n$ matrix with a single 1 in each row corresponding to the nodes in $\mathcal{X}$ that are neighbors of node x_i, along with node x_i itself (the first $n-2f$ columns correspond to nodes in $\bar{S}$ and the last $2f$ columns correspond to nodes in S).

Lemma 7.3. *For any set S of $2f$ nodes, the invariant zeros of $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$.*

Proof. Without loss of generality, we can assume that W is of the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{\bar{S}} & \mathbf{W}_{S,\bar{S}} \\ \mathbf{W}_{\bar{S},S} & \mathbf{W}_S \end{bmatrix},$$

since this is obtained simply by renumbering the nodes so that nodes in the set S have indices between $n-2f+1$ and n. This also means that $\mathbf{B}_S$ has the form $\mathbf{B}_S = \left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}_{2f}\end{smallmatrix}\right]$, and $\mathbf{P}(z)$ can be written as

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{W}_{\bar{S}} - z\mathbf{I}_{n-2f} & \mathbf{W}_{S,\bar{S}} & \mathbf{0} \\ \mathbf{W}_{\bar{S},S} & \mathbf{W}_S - z\mathbf{I}_{2f} & \mathbf{I}_{2f} \\ \mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S} & \mathbf{0} \end{bmatrix} .$$

From this expression, we see that

$$\operatorname{rank}(\mathbf{P}(z)) = 2f + \operatorname{rank}\left(\begin{bmatrix} \mathbf{W}_{\bar{S}} - z\mathbf{I}_{n-2f} & \mathbf{W}_{S,\bar{S}} \\ \mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S} \end{bmatrix} \right) ,$$

and so the invariant zeros of the set $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$. □

Lemma 7.3 reveals that in order to ensure that the system $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ is strongly observable for every set S of $2f$ nodes and every node x_i, we can focus on the problem of choosing the weight matrix $\mathbf{W}$ so that the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$ will have no invariant zeros, for any choice of i and for any set S of $2f$ nodes. The tricky part is that the weight matrix is structured, and we will somehow have to relate the invariant zeros of the system to the zero/nonzero constraints in $\mathbf{W}$. To accomplish this, we will use techniques from a branch of control theory pertaining to *linear structured systems* [66, 18]. See Appendix C for an overview of some of the main concepts from structured system theory.

Application to Secure Information Dissemination

To apply structured system theory to the problem of determining the number of invariant zeros of the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$, we note that all matrices in this set are essentially structured matrices, with the exception that the nonzero entries in $\mathbf{C}_{i,\bar{S}}$ and $\mathbf{C}_{i,S}$ are taken to be 1 rather than free parameters. However, this is easily remedied by defining the matrix $\Lambda_i = \operatorname{diag}\left(\lambda_{i,1}, \lambda_{i,2}, \cdots, \lambda_{i,\deg_i+1}\right)$, where the $\lambda_{i,j}$'s are a set of free (independent) parameters. Now consider the matrix $\begin{bmatrix}\bar{\mathbf{C}}_{i,\bar{S}} & \bar{\mathbf{C}}_{i,S}\end{bmatrix} = \Lambda_i \begin{bmatrix}\mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S}\end{bmatrix}$. The nonzero entries in the matrices $\bar{\mathbf{C}}_{i,\bar{S}}$ and $\bar{\mathbf{C}}_{i,S}$ are thus independent free parameters (at most one per row in each matrix), and these matrices therefore qualify as valid structured matrices. Now consider the matrix pencil

$$\bar{\mathbf{P}}(z) = \begin{bmatrix} \mathbf{W}_{\bar{S}} - z\mathbf{I}_{n-2f} & \mathbf{W}_{S,\bar{S}} \\ \bar{\mathbf{C}}_{i,\bar{S}} & \bar{\mathbf{C}}_{i,S} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n-2f} & 0 \\ 0 & \Lambda_i \end{bmatrix} \begin{bmatrix} \mathbf{W}_{\bar{S}} - z\mathbf{I}_{n-2f} & \mathbf{W}_{S,\bar{S}} \\ \mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S} \end{bmatrix} .$$

If the matrix Λ_i is invertible, the matrix pencil for the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \bar{\mathbf{C}}_{i,\bar{S}}, \bar{\mathbf{C}}_{i,S})$ will have the same rank as the matrix pencil for the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$ for all $z \in \mathbb{C}$. Thus, for the purposes of analyzing the number of invariant zeros of the system, we can assume without loss of generality that the nonzero entries in $\mathbf{C}_{i,S}$ and $\mathbf{C}_{i,\bar{S}}$ are all ones (i.e., we can take Λ_i to be the identity matrix).

Let us start by defining the graph $\mathcal{H}$ associated with the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$. For this set of matrices, note that the state vertices in the graph $\mathcal{H}$ are given by the set $\bar{S}$, and the input vertices are given by the set S; thus the graph has $n-2f$ state vertices and $2f$ input vertices. In particular, $\mathcal{H}$ can be obtained by first taking the graph of the network $\mathcal{G}$, and removing all incoming edges to nodes in S (since the nodes in S are treated as inputs in the above set of matrices). To this graph, add $\deg_i + 1$ output vertices (denoted by the set $\mathcal{Y}_i$), and place a single edge from the set $x_i \cup \mathcal{N}_i$ to vertices in $\mathcal{Y}_i$, corresponding to the single nonzero entry in each row of the matrices $\mathbf{C}_{i,\bar{S}}$ and $\mathbf{C}_{i,S}$. Furthermore, add a set of self loops to every state vertex to correspond to the nonzero entries on the diagonal of the weight matrix $\mathbf{W}_{\bar{S}}$.

We can now use the above results on structured systems to prove the following lemma for the linear iteration in (7.4). We will then use this lemma, in conjunction with Lemma 7.3 and Theorem 7.4, to show that all nodes will be able to uniquely determine the vector $\mathbf{x}[0]$ after running the linear iteration for at most n time-steps (with up to f malicious nodes).

Lemma 7.4. *Let the graph of the network $\mathcal{G}$ have connectivity κ. Let x_i be any node in the network, and let S be any set of $2f$ nodes. If $\kappa \geq 2f+1$, then for almost any choice of weights, the set $(\mathbf{W}_{\bar{S}}, \mathbf{W}_{S,\bar{S}}, \mathbf{C}_{i,\bar{S}}, \mathbf{C}_{i,S})$ will have no invariant zeros.*

Proof. We will show that the matrix pencil

$$\begin{bmatrix} \mathbf{W}_{\bar{S}} - z\mathbf{I}_{n-2f} & \mathbf{W}_{S,\bar{S}} \\ \mathbf{C}_{i,\bar{S}} & \mathbf{C}_{i,S} \end{bmatrix}$$

has full normal-rank (equal to n) even after the deletion of an arbitrary row. We will then use Theorem C.6 to prove the lemma.

Suppose we remove one of the rows of $\mathbf{P}(z)$ corresponding to a vertex $v \in \bar{S}$ (i.e., one of the top $n-2f$ rows of $\mathbf{P}(z)$), and denote the resulting matrix by $\bar{\mathbf{P}}(z)$. Note that removing this row is equivalent to removing all incoming edges to vertex v in $\mathcal{H}$; however, all outgoing edges from v are still left in the graph (since the column corresponding to vertex v is left in matrix $\bar{\mathbf{P}}(z)$). Thus, if we denote the resulting graph (after removing the incoming edges to v in $\mathcal{H}$) by $\bar{\mathcal{H}}$, we see that vertex v can be treated as an input vertex in $\bar{\mathcal{H}}$. Since the set $S \cup v$ has size $2f+1$, and since the graph $\mathcal{G}$ of the network has connectivity $\kappa \geq 2f+1$, Theorem B.2 says that there will be a linking of size $2f+1$ from the set $S \cup v$ to the set $x_i \cup \mathcal{N}_i$ in $\mathcal{G}$, and hence in $\bar{\mathcal{H}}$ (since such a linking would not use any incoming edges to vertex v). From Theorem C.2, the matrix pencil $\mathbf{P}(z)$ will thus have rank equal to $(n-2f-1)+2f+1 = n$ for almost any choice of $\mathbf{W}$, even after the deletion of the row of corresponding to any vertex $v \in \bar{S}$.

Now, instead of removing one of the first $n-2f$ rows, suppose we remove one of the rows of $\mathbf{P}(z)$ corresponding to one of the output vertices in $\mathcal{Y}_i$ (i.e., one of the bottom $\deg_i + 1$ rows of $\mathbf{P}(z)$). Since $\deg_i + 1 \geq \kappa + 1 \geq 2f+2$, removing one of the output vertices leaves at least $2f+1$ output vertices in the graph $\mathcal{H}$. Choose any $2f$ of the nodes in $x_i \cup \mathcal{N}_i$ such that none of the chosen nodes have an edge to the removed output vertex in $\mathcal{H}$ (this is possible since each output vertex only connects to a single vertex in

$x_i \cup \mathcal{N}_i$). Denote these $2f$ nodes by $\mathcal{X}_i$. Once again, since the graph is at least $2f+1$ connected, there exists a linking of size $2f$ from S to $\mathcal{X}_i$, and therefore to the remaining output vertices in the graph $\mathcal{H}$. From Theorem C.2, the matrix pencil will have rank equal to $(n-2f)+2f = n$ for almost any choice of $\mathbf{W}$, even after the deletion of one of the bottom $\deg_i + 1$ rows of $\mathbf{P}(z)$).

The above arguments show that $\mathbf{P}(z)$ will generically have full column rank after the deletion of an arbitrary row. From Theorem C.6, the number of invariant zeros of $\mathbf{P}(z)$ is equal to $n-2f$ minus the maximal number of vertices in $\bar{S}$ contained in the disjoint union of a size $2f$ linking from S to $\mathcal{Y}_i$, a cycle family in $\bar{S}$, and a $\mathcal{Y}_i$-topped path family. If we simply take all the self-loops in $\mathcal{H}$ (corresponding to the nonzero weights on the diagonal of the weight matrix $\mathbf{W}_{\bar{S}}$), we will have a set of disjoint cycles that covers all $n-2f$ vertices in $\bar{S}$. Thus, the matrix pencil $\mathbf{P}(z)$ will generically have no invariant zeros, thereby proving the lemma. □

With the above lemma in hand, one can now prove the following result.

Theorem 7.5. *Let f denote the maximum number of malicious nodes that are to be tolerated in a given network $\mathcal{G}$, and let κ denote the connectivity of the network. If $\kappa \geq 2f+1$, then for almost any choice of weight matrix, every node in the network can calculate any function of the initial values after running the linear iteration for at most n time-steps, even when there are up to f malicious nodes that update their values arbitrarily (and possibly in a coordinated manner) at each time-step.*

Proof. From Lemmas 7.3 and 7.4, we see that for almost any choice of weight matrix $\mathbf{W}$ (satisfying the required sparsity constraints), the set $(\mathbf{W}, \mathbf{B}_S, \mathbf{C}_i, \mathbf{0})$ will have no invariant zeros, for any x_i and any set S of $2f$ nodes. Furthermore, since the set of weights for which this property does not hold lies on an algebraic variety (i.e., it holds for almost any choice of weights), it will hold generically (and therefore simultaneously) for all x_i and for all possible sets S of $2f$ nodes. Thus, Theorem 7.4 is satisfied for all x_i, and every node can calculate any arbitrary function of the initial values after running the linear iteration for at most n time-steps, despite the actions of up to f malicious nodes. □

Remark 7.1. The above results were derived in [83, 84, 85]. Extensions of these results to the case where some nodes are faulty but not malicious (i.e., nodes that do not inject the worst-case inputs into the system), and where the objective is to reach consensus in the network, were performed in [62], using ideas similar to the fault-diagnosis schemes covered in Chapter 4.

Appendix A

Linear Algebra

A.1 Fields

A *field* $\mathbb{F}$ is a set of elements, together with two operations written as addition ($+$) and multiplication ($\times$),[1] satisfying the following properties [22]:

1. *Closure:* $a + b \in \mathbb{F}$ and $ab \in \mathbb{F}$ for all $a, b \in \mathbb{F}$.
2. *Commutativity:* $a + b = b + a$ and $ab = ba$ for all $a, b \in \mathbb{F}$.
3. *Associativity:* $(a + b) + c = a + (b + c)$ and $(ab)c = a(bc)$ for all $a, b, c \in \mathbb{F}$.
4. *Distributivity:* $a(b + c) = ab + ac$ for all $a, b, c \in \mathbb{F}$.
5. *Additive identity:* There exists an element $0 \in \mathbb{F}$ such that for all $a \in \mathbb{F}$, $a + 0 = a$.
6. *Multiplicative identity:* There exists an element $1 \in \mathbb{F}$ such that for all $a \in \mathbb{F}$, $1a = a$.
7. *Additive inverse:* For each element $a \in \mathbb{F}$, there exists an element $d \in \mathbb{F}$ such that $a + d = 0$.
8. *Multiplicative inverse:* For each nonzero element $a \in \mathbb{F}$, there exists an element $e \in \mathbb{F}$ such that $ae = 1$.

The number of elements in a field can be infinite (such as in the field of real or complex numbers), or finite. We will be dealing only with fields of infinite size in this course.

[1] The multiplicative operator is also frequently denoted by simply concatenating the operands (i.e., $a \times b$ is written as ab). We will adopt this convention throughout the discussion.

A.2 Vector Spaces and Subspaces

A *vector space* $\mathbb{V}$ over a field $\mathbb{F}$ (denoted as $(\mathbb{V}, \mathbb{F})$) is a set of elements $\mathbb{V}$, together with two operations written as addition ($+$) and multiplication ($\times$), satisfying the following properties:

1. *Closure under addition:* $\mathbf{x} + \mathbf{y} \in \mathbb{V}$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{V}$.
2. *Closure under scalar multiplication:* $a\mathbf{x} \in \mathbb{V}$ for all $\mathbf{x} \in \mathbb{V}$ and $a \in \mathbb{F}$.
3. *Commutativity of addition:* $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{V}$.
4. *Associativity of addition:* $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{V}$.
5. *Additive identity:* There exists an element $\mathbf{0} \in \mathbb{V}$ such that for all $\mathbf{x} \in \mathbb{V}$, $\mathbf{x}+\mathbf{0} = \mathbf{x}$.
6. *Additive inverse:* For each $\mathbf{x} \in \mathbb{V}$, there exists an element $\mathbf{y} \in \mathbb{V}$ such that $\mathbf{x} + \mathbf{y} = \mathbf{0}$.
7. *Multiplication by scalar identity:* For all $\mathbf{x} \in \mathbb{V}$, the multiplicative identity 1 in the field $\mathbb{F}$ satisfies $1\mathbf{x} = \mathbf{x}$.
8. *Associativity under scalar multiplication:* For all $a, b \in \mathbb{F}$ and for all $\mathbf{x} \in \mathbb{V}$, $(ab)\mathbf{x} = a(b\mathbf{x})$.
9. *Distributivity of scalars:* For all $a, b \in \mathbb{F}$ and for all $\mathbf{x}, \mathbf{y} \in \mathbb{V}$, $a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$ and $(a + b)\mathbf{x} = a\mathbf{x} + b\mathbf{x}$.

The elements of $\mathbb{V}$ are called *vectors* and the elements of $\mathbb{F}$ are called *scalars*. As done above, we will denote vectors with a boldface character $\mathbf{x} \in \mathbb{V}$, and scalars in the usual typeface $a \in \mathbb{F}$.

A *subspace* of a vector space $(\mathbb{V}, \mathbb{F})$ is a subset $\mathbb{W} \subseteq \mathbb{V}$ such that $(\mathbb{W}, \mathbb{F})$ is a vector space with the operations of addition and multiplication defined above.

Although the term *vector* refers to any element of a set $\mathbb{V}$ satisfying the above vector space properties, we will most commonly encounter vectors in the following sense. For any elements $a_1, a_2, \ldots, a_n \in \mathbb{F}$, define the *column vector*

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}.$$

The set of all column vectors of the above form is denoted by $\mathbb{F}^n$, and is a vector space over the field $\mathbb{F}$. The operations of addition and multiplication by scalars in this vector space

are done on a component by component basis, i.e., for any $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n, b \in \mathbb{F}$,

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_n + b_n \end{bmatrix}, \quad b \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} ba_1 \\ ba_2 \\ \vdots \\ ba_n \end{bmatrix}.$$

Alternatively, one can define a *row vector*

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix},$$

with the operations of addition and multiplication defined in an analogous manner.

A.3 Linear Independence and Bases

Given a vector space $(\mathbb{V}, \mathbb{F})$, a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r \in \mathbb{V}$ are said to be *linearly independent* if the only set of scalars $a_1, a_2, \ldots, a_r \in \mathbb{F}$ satisfying the equation

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_r\mathbf{v}_r = \mathbf{0}$$

is $a_1 = a_2 = \cdots = a_r = 0$. Otherwise, the vectors are said to be linearly dependent. Note that if a set of vectors are not linearly independent, then one of the vectors can be written as a linear combination of the other vectors.

Let $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\}$ be a set of linearly independent vectors of maximal size in the vector space $(\mathbb{V}, \mathbb{F})$. The set $\mathcal{B}$ is called a *basis* for the vector space. The following result holds.

Theorem A.1. *Any vector in the vector space $(\mathbb{V}, \mathbb{F})$ can be written as a linear combination of the vectors in $\mathcal{B}$.*

Proof. Consider any vector $\mathbf{v} \in \mathbb{V}$. Since $\mathcal{B}$ is a linearly independent set of maximal size, it must be the case that there is a nontrivial solution $a, a_1, a_2, \ldots, a_r$ to the equation

$$a\mathbf{v} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_r\mathbf{v}_r = \mathbf{0};$$

otherwise, the set $\{\mathbf{v}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\}$ is a set of linearly independent vectors that is larger than $\mathcal{B}$. Next, note that it must be the case that $a \neq 0$, since otherwise, there would be a nontrivial linear combination of the vectors in $\mathcal{B}$ that sum to zero, contradicting the assumption of a linearly independent set. Thus, we can write

$$\mathbf{v} = -\frac{a_1}{a}\mathbf{v}_1 - \frac{a_2}{a}\mathbf{v}_2 - \cdots - \frac{a_r}{a}\mathbf{v}_r,$$

proving the above statement. □

Theorem A.2. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two different bases for a vector space $(\mathbb{V}, \mathbb{F})$. Then $|\mathcal{B}_1| = |\mathcal{B}_2|$.*

The proof of the above result can be found in [22]. In words, the above result says that all bases for a vector space are comprised of the same number of vectors. The size of any basis for a vector space $(\mathbb{V}, \mathbb{F})$ is called the *dimension* of the vector space, and is denoted by $\dim(\mathbb{V})$.

A.4 Matrices

An $m \times n$ *matrix* $\mathbf{A}$ over a field $\mathbb{F}$ is a rectangular array of elements from $\mathbb{F}$, depicted as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

When $m = n$, the matrix is said to be *square*. Note that a column vector is a matrix with $n = 1$ and a row-vector is a matrix with $m = 1$. The set of all $m \times n$ matrices over a field $\mathbb{F}$ is denoted by $\mathcal{M}_{m,n}(\mathbb{F})$. Each column of an $m \times n$ matrix can be viewed as an element of the vector space $(\mathbb{F}^m, \mathbb{F})$, and each row can be viewed as an element of the vector space $(\mathbb{F}^n, \mathbb{F})$. The element in row i and column j of a matrix $\mathbf{A}$ is denoted by a_{ij}. For a vector $\mathbf{a} \in \mathcal{M}_{n,1}(\mathbb{F})$, the i–th component is simply denoted as a_i. The *transpose* of an $m \times n$ matrix $\mathbf{A}$ is the $n \times m$ matrix obtained by converting each of the columns into rows, and is denoted by

$$\mathbf{A}' = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}.$$

With this notation, row vectors will usually be denoted as the transpose of a column vector (i.e., $\mathbf{a}$ is a column vector and $\mathbf{a}'$ is a row vector).

The *inner product* of two column vectors $\mathbf{a}, \mathbf{b} \in \mathcal{M}_{n,1}$ is defined as

$$\mathbf{a}'\mathbf{b} \triangleq \sum_{k=1}^{n} a_k b_k.$$

The product of two matrices $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ and $\mathbf{B} \in \mathcal{M}_{p,q}(\mathbb{F})$ is defined if and only if $n = p$, and produces a matrix $\mathbf{C} \in \mathcal{M}_{m,q}(\mathbb{F})$. Entry (i, j) in matrix $\mathbf{C} = \mathbf{AB}$ is obtained

as

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

A square $n \times n$ matrix $\mathbf{A}$ is said to be *diagonal* if all entries except $a_{11}, a_{22}, \ldots, a_{nn}$ are zero. An $n \times n$ diagonal matrix with $a_{11} = a_{22} = \cdots = a_{nn} = 1$ is called the *identity matrix*, and is denoted by $\mathbf{I}_n$. Note that for any matrix $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{F})$, we have $\mathbf{I}_m\mathbf{A} = \mathbf{A}\mathbf{I}_n = \mathbf{A}$. A matrix is said to be *upper-triangular* if all entries below the main diagonal are zero, and *lower-triangular* if all entries above the main diagonal are zero.

The *rank* of a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ is defined as the maximum number of linearly independent columns in that matrix, and is denoted by rank($\mathbf{A}$). The following results are fundamental (see [22] for proofs).

Theorem A.3. *Given a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$:*

1. *rank*($\mathbf{A}$) = *rank*($\mathbf{A}'$).
2. *rank*($\mathbf{A}$) $\leq \min(m, n)$.
3. *For any matrices $\mathbf{B} \in \mathcal{M}_{p,m}(\mathbb{F})$ and $\mathbf{C} \in \mathcal{M}_{n,q}(\mathbb{F})$,*

$$\textit{rank}(\mathbf{BAC}) \leq \min(\textit{rank}(\mathbf{A}), \textit{rank}(\mathbf{B}), \textit{rank}(\mathbf{C})).$$

4. *For any matrices $\mathbf{B} \in \mathcal{M}_{p,m}(\mathbb{F})$ and $\mathbf{C} \in \mathcal{M}_{n,q}(\mathbb{F})$ satisfying rank($\mathbf{B}$) = m and rank($\mathbf{C}$) = n, rank($\mathbf{BAC}$) = rank($\mathbf{A}$).*

The first identity in the above list is of particular importance. Since the columns of $\mathbf{A}'$ are simply the rows of $\mathbf{A}$, this identity says that the maximum number of linearly independent columns in $\mathbf{A}$ is *equal* to the maximum number of linearly independent rows in $\mathbf{A}$.

A matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ is said to be *full-column-rank* if rank($\mathbf{A}$) = n, and *full-row-rank* if rank($\mathbf{A}$) = m. A full-column-rank matrix is also called *nonsingular*. A square matrix that is full-column-rank (or equivalently full-row-rank) is said to be *nonsingular* or *invertible*. For every invertible matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$, there is another invertible matrix, denoted $\mathbf{A}^{-1} \in \mathcal{M}_{n,n}(\mathbb{F})$ such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}.$$

A.4.1 Linear transformations, Range space and Null Space

Given a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ and a vector $\mathbf{x} \in \mathbb{F}^n$, the product $\mathbf{A}\mathbf{x}$ is a vector in $\mathbb{F}^m$. Thus the matrix $\mathbf{A}$ can be viewed as a *mapping* or *transformation* from the vector space

$(\mathbb{F}^n, \mathbb{F})$ to the vector space $(\mathbb{F}^m, \mathbb{F})$. More specifically, it is a *linear mapping* because it satisfies the *Principle of Superposition*:

$$\mathbf{A}(\alpha_1 \mathbf{x}_1 + \beta \mathbf{x}_2) = \alpha \mathbf{A}\mathbf{x}_1 + \beta \mathbf{A}\mathbf{x}_2, \ \ \forall \alpha, \beta \in \mathbb{F}, \ \ \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}^n.$$

The *range space* (also known as the *column space* or *image*) of the linear transformation $\mathbf{A}$ is defined as

$$\mathcal{R}(\mathbf{A}) \triangleq \{\mathbf{y} \in \mathbb{F}^m \ \mid \ \mathbf{y} = \mathbf{A}\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{F}^n\},$$

and the *null space* (or *kernel*) of $\mathbf{A}$ is defined as

$$\mathcal{N}(\mathbf{A}) \triangleq \{\mathbf{x} \in \mathbb{F}^n \ \mid \ \mathbf{A}\mathbf{x} = \mathbf{0}\}.$$

One can readily verify that $\mathcal{R}(\mathbf{A})$ and $\mathcal{N}(\mathbf{A})$ are, in fact, subspaces of $\mathbb{F}^m$ and $\mathbb{F}^n$, respectively. Since every vector in the space $\mathcal{R}(\mathbf{A})$ is a linear combination of the columns of $\mathbf{A}$, we see that a maximal set of linearly independent columns from $\mathbf{A}$ forms a basis for $\mathcal{R}(\mathbf{A})$. Furthermore, $\dim(\mathcal{R}(\mathbf{A})) = \text{rank}(\mathbf{A})$.

The dimension of the null space of $\mathbf{A}$ is called the *nullity* of $\mathbf{A}$. The following fundamental result relates the rank and nullity of a linear transformation $\mathbf{A}$.

Theorem A.4. *For any matrix* $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$, $\dim(\mathcal{R}(\mathbf{A})) + \dim(\mathcal{N}(\mathbf{A})) = n$.

Proof. Suppose that the dimension of $\mathcal{N}(\mathbf{A})$ is k, and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ form a basis for this space. Let $\mathbf{v}_{k+1}, \mathbf{v}_{k+2}, \ldots, \mathbf{v}_n$ be any set of $k - n$ vectors such that $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is a linearly independent set; this set spans the entire space $\mathbb{F}^n$. Now consider any vector $\mathbf{y}$ in the range space of $\mathbf{A}$, and note that by the definition of the range space,

$$\begin{aligned} \mathbf{y} = \mathbf{A}\mathbf{x} &= \mathbf{A}\left(\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_n \mathbf{v}_n\right) \\ &= \mathbf{A}\left(\alpha_{k+1} \mathbf{v}_{k+1} + \alpha_{k+2} \mathbf{v}_{k+2} + \cdots + \alpha_n \mathbf{v}_n\right) \end{aligned}$$

for some scalars $\alpha_1, \ldots, \alpha_n$ (this is because any vector $\mathbf{x} \in \mathbb{F}^n$ can be written as a linear combination of the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$). The last line in the above expression follows since $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are in the null space of $\mathbf{A}$. Thus, the vectors $\mathbf{A}\mathbf{v}_{k+1}, \mathbf{A}\mathbf{v}_{k+2}, \ldots, \mathbf{A}\mathbf{v}_n$ span the range space of $\mathbf{A}$ (i.e., any vector in $\mathcal{R}(\mathbf{A})$ can be written as a linear combination of those vectors). It only remains to show that these vectors are linearly independent, and thus are indeed a basis. To do this, suppose that there is some nontrivial linear combination of $\mathbf{A}\mathbf{v}_{k+1}, \mathbf{A}\mathbf{v}_{k+2}, \ldots, \mathbf{A}\mathbf{v}_n$ that is equal to zero, i.e.,

$$\beta_{k+1} \mathbf{A}\mathbf{v}_{k+1} + \beta_{k+2} \mathbf{A}\mathbf{v}_{k+2} + \cdots + \beta_n \mathbf{A}\mathbf{v}_n = \mathbf{0}$$

for some scalars $\beta_{k+1}, \ldots, \beta_n$. By pulling out $\mathbf{A}$ on the left, this is equivalent to saying that the vector $\beta_{k+1}\mathbf{v}_{k+1} + \cdots \beta_n \mathbf{v}_n$ is in the null space of $\mathbf{A}$, or equivalently that

$$\beta_{k+1}\mathbf{v}_{k+1} + \cdots \beta_n \mathbf{v}_n = \gamma_1 \mathbf{v}_1 + \gamma_2 \mathbf{v}_2 + \cdots + \gamma_k \mathbf{v}_k$$

for some scalars $\gamma_1, \ldots, \gamma_k$ (again, because $\mathbf{v}_1, \ldots, \mathbf{v}_k$ form a basis for the null space of $\mathbf{A}$). But this would mean that there is a nontrivial linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ that is equal to zero, and this is not possible by the fact that these vectors were chosen to be linearly independent. Thus, it must be the case that $\mathbf{A}\mathbf{v}_{k+1}, \mathbf{A}\mathbf{v}_{k+2}, \ldots, \mathbf{A}\mathbf{v}_n$ are linearly independent, and do indeed form a basis for the range space of $\mathbf{A}$. This shows that $\dim(\mathcal{R}(\mathbf{A})) = n - k$, and so $\dim(\mathcal{R}(\mathbf{A})) + \dim(\mathcal{N}(\mathbf{A})) = n$. □

One can also speak of the *row space* of a given matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$, which is the space spanned by the rows of $\mathbf{A}$. Similarly, the *left null space* of $\mathbf{A}$ is the set of all row vectors $\mathbf{x} \in \mathbb{F}^m$ such that $\mathbf{x}\mathbf{A} = \mathbf{0}$.

A.4.2 Systems of Linear Equations

Given a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F})$ and a vector $\mathbf{y} \in \mathbb{F}^m$, we will often be interested in solving a linear equation of the form

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

for the vector $\mathbf{x} \in \mathbb{F}^n$. First, note that there exists such a vector if and only if $\mathbf{y} \in \mathcal{R}(\mathbf{A})$. However, this does not mean that there will be a *unique* solution for $\mathbf{x}$. Specifically, consider any solution $\mathbf{x}$, and consider any vector $\mathbf{n} \in \mathcal{N}(\mathbf{A})$. Then $\mathbf{A}(\mathbf{x}+\mathbf{n}) = \mathbf{A}\mathbf{x}+\mathbf{A}\mathbf{n} = \mathbf{y} + \mathbf{0} = \mathbf{y}$, and so $\mathbf{x} + \mathbf{n}$ is also a feasible solution. Thus, if the null space of $\mathbf{A}$ is non-empty (i.e., $\mathbf{A}$ is not full-column-rank), then there will be an infinite number of solutions to the equation. The following lemma shows the converse result, and also demonstrates an important proof method.

> **Lemma A.1.** *Let* $\mathbf{A}$ *be a nonsingular matrix (i.e., all of its columns are linearly independent). Then there is a unique solution* $\mathbf{x}$ *to the equation* $\mathbf{A}\mathbf{x} = \mathbf{y}$*, for any vector* $\mathbf{y}$*.*

Proof. Suppose that there were, in fact, two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$. This would mean that $\mathbf{y} = \mathbf{A}\mathbf{x}_1 = \mathbf{A}\mathbf{x}_2$, which in turn means that $\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{0}$. Thus, $\mathbf{x}_1 - \mathbf{x}_2$ is in the null space of $\mathbf{A}$, which is empty due to the fact that $\mathbf{A}$ is nonsingular. Thus $\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}$, which produces the desired contradiction. □

The above argument is in the form of a *proof by contradiction*. This is an extremely effective proof technique, and very commonly used:

> *Reductio ad absurdum*, which Euclid loved so much, is one of a mathematician's finest weapons. It is a far finer gambit than any chess gambit: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game. – G. H. Hardy [34]

A.4.3 Determinants

The *determinant* is a function that maps a square matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$ to a scalar value in the field $\mathbb{F}$. For $n = 2$, the determinant is given by

$$\det\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = a_{11}a_{22} - a_{21}a_{22}.$$

Determinants of higher order matrices can be defined recursively using the following *Laplace expansion*. First, for $n = 1$, the determinant of the matrix is defined to be the single scalar element comprising that matrix. For $n \geq 2$, let $\tilde{\mathbf{A}}_{ij}$ denote the $(n-1)\times(n-1)$ matrix that is obtained by removing row i and column j from matrix $\mathbf{A}$. Then, for any $1 \leq i \leq n$,

$$\det(\mathbf{A}) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(\tilde{\mathbf{A}}_{ij}).$$

This expansion will yield the same result for $\det(\mathbf{A})$ no matter which row i is used. Alternatively, the expansion can be performed around any column, yielding the same result. For example, for $n = 3$ and with $i = 1$, we have

$$\begin{aligned}
\det\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right) &= a_{11} \det\left(\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}\right) - a_{12} \det\left(\begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix}\right) \\
&\quad + a_{13} \det\left(\begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}\right) \\
&= a_{11}a_{22}a_{33} - a_{11}a_{23}a_{32} + a_{12}a_{23}a_{31} - a_{12}a_{21}a_{33} \\
&\quad + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31}.
\end{aligned}$$

Taking a closer look at the above determinants, we see the following.

> The determinant of a matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$ is a polynomial in the elements a_{ij}. Each term in the polynomial is a product of n elements, no two of which are from the same row or column of $\mathbf{A}$. Furthermore, each of these products is multiplied by either $+1$ or -1.

An immediate consequence of the above definitions is that the determinant of an upper-triangular or a lower-triangular matrix is simply the product of the elements on the diagonal. Some other important properties of determinants are summarized below (see [39] for proofs).

Theorem A.5. *Given matrices* $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{n,n}(\mathbb{F})$*:*

1. $\det(\mathbf{A}) = 0$ *if and only if rank*$(\mathbf{A}) < n$.
2. $\det(\mathbf{A}) = \det(\mathbf{A}')$.
3. $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$.
4. *If* $\mathbf{A}$ *is invertible, then* $\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})}$.

A.4.4 Eigenvalues and Eigenvectors

An element $\lambda \in \mathbb{F}$ is called an *eigenvalue* of a square matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$ if there exists a vector $\mathbf{v} \in \mathbb{F}^n$ such that

$$\mathbf{Av} = \lambda\mathbf{v} \ .$$

The vector $\mathbf{v}$ is called the *eigenvector* corresponding to the eigenvalue λ. To further characterize the eigenvalues of a given matrix $\mathbf{A}$, one can rearrange the above equation to obtain $(\lambda\mathbf{I}_n - \mathbf{A})\,\mathbf{v} = \mathbf{0}$. This indicates that any eigenvalue-eigenvector pair λ and $\mathbf{v}$ must be such that $\mathbf{v}$ lies in the null space of the matrix $\lambda\mathbf{I}_n - \mathbf{A}$. Recall from Theorem A.5 in Section A.4.3 that the matrix $\lambda\mathbf{I}_n - \mathbf{A}$ will have a nontrivial null space if and only if $\det(\lambda\mathbf{I}_n - \mathbf{A}) = 0$. Next, since the determinant of a matrix is simply a polynomial in the elements of that matrix, we see that

$$\det(\lambda\mathbf{I}_n - \mathbf{A}) = \lambda^n + p_{n-1}\lambda^{n-1} + p_{n-2}\lambda^{n-2} + \cdots + p_1\lambda + p_0 \ , \tag{A.1}$$

where $p_0, p_1, \ldots, p_{n-1}$ are each polynomials in the elements of $\mathbf{A}$. This is called the *characteristic polynomial* of matrix $\mathbf{A}$ and the roots are the eigenvalues of matrix $\mathbf{A}$. When the field under consideration is the field of complex numbers $\mathbb{C}$, the *Fundamental Theorem of Algebra*[2] says that the above polynomial will have n solutions for λ. This brings us to the following result.

Any matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{C})$ has n eigenvalues.

It is not necessarily the case that all of the eigenvalues will be distinct (i.e., if the polynomial (A.1) has repeated roots). The number of times that λ appears as a root of (A.1) is called the *algebraic multiplicity* of that eigenvalue; this is usually shortened to *multiplicity*. For a given eigenvalue λ, the null space of the matrix $\lambda\mathbf{I}_n - \mathbf{A}$ is called the *eigenspace* of $\mathbf{A}$ associated with λ (because any vector in this space is an eigenvector of

[2]This theorem states that any polynomial of degree n over the field of complex numbers has n roots in that field. This result does not hold in arbitrary fields; in certain fields, it is possible for a given polynomial not to have *any* roots in that field.

$\mathbf{A}$ associated with λ). The dimension of the eigenspace associated with λ indicates the maximum number of linearly independent eigenvectors associated with λ, and is called the *geometric multiplicity* of λ. Note that the geometric multiplicity of λ might be less than its algebraic multiplicity. For example, consider the matrix

$$\mathbf{A} = \begin{bmatrix} a & 1 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \Rightarrow \det(\lambda \mathbf{I}_3 - \mathbf{A}) = (\lambda - a)^3.$$

This polynomial has three roots at a, and thus the matrix has an eigenvalue of a, with algebraic multiplicity 3. Now consider the matrix

$$a\mathbf{I}_3 - \mathbf{A} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

This matrix has a rank of 1, and a nullity of $3-1 = 2$. Thus the eigenspace corresponding to eigenvalue a has dimension 2, which means that there are only two linearly independent eigenvectors for that eigenvalue. Thus the geometric multiplicity of eigenvalue a is two.

For a given matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$, let $\lambda_1, \lambda_2, \ldots, \lambda_d$ denote the distinct eigenvalues. For each eigenvalue $\lambda_i, 1 \leq i \leq d$, let $\mathbf{V}_i$ denote a matrix whose columns form a basis for the eigenspace of $\mathbf{A}$ associated with λ_i (i.e., $(\lambda_i \mathbf{I}_n - \mathbf{A})\mathbf{V}_i = \mathbf{0}$). Note that the columns of $\mathbf{V}_i$ form a set of linearly independent eigenvectors of $\mathbf{A}$ associated with λ_i. The following result holds.

Lemma A.2. *For a given matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$, let $\lambda_1, \lambda_2, \ldots, \lambda_d$ be the distinct eigenvalues, and let $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_d$ be the matrices whose columns are the eigenvectors corresponding to the eigenvalues. Then all of the columns of the matrix $\begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_d \end{bmatrix}$ are linearly independent.*

The above result is equivalent to saying that all eigenvectors of a given matrix are linearly independent. We will provide a proof here for the case where the matrix has n different eigenvalues. For a proof of the above result in the general case, see [39].

Proof. (For n different eigenvalues) Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the n different eigenvalues of the matrix, and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be corresponding eigenvectors. Suppose that they *are not* linearly independent.[3] Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ be the largest possible subset of the eigenvectors that are linearly independent (this can be done by reordering the eigenvalues and eigenvectors, as necessary). Thus, $\mathbf{v}_{r+1}$ must be linearly dependent on this subset, and so we can write

$$\mathbf{v}_{r+1} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_r \mathbf{v}_r \tag{A.2}$$

[3] Again, this is setting up a proof by contradiction.

for some nonzero scalars $\alpha_1, \alpha_2, \ldots, \alpha_r$. Now, multiply both sides of the above equation on the left by $\mathbf{A}$ and note that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ to get

$$\lambda_{r+1}\mathbf{v}_{r+1} = \alpha_1\lambda_1\mathbf{v}_1 + \alpha_2\lambda_2\mathbf{v}_2 + \cdots + \alpha_r\lambda_r\mathbf{v}_r.$$

Substituting (A.2) into the left hand side of the above equation and rearranging, we get

$$\alpha_1(\lambda_1 - \lambda_{r+1})\mathbf{v}_1 + \alpha_2(\lambda_2 - \lambda_{r+1})\mathbf{v}_2 + \cdots + \alpha_r(\lambda_r - \lambda_{r+1})\mathbf{v}_r = 0.$$

Now, since all of the λ_i's are assumed to be distinct, none of the terms $\lambda_i - \lambda_{r+1}$ are equal to zero. Furthermore, there exists at least one α_i that is nonzero (as described above). Thus, this equation states that there is some nonzero linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ that sum to zero. But this contradicts the fact that these vectors were chosen as the largest linearly independent set from the set of all eigenvectors. Thus, it cannot be the case that we can write $\mathbf{v}_{r+1}$ as a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_r$, and thus, all eigenvectors in the original set must be linearly independent. □

Recall from Theorem A.3 that the number of linearly independent columns in a matrix is the same as the number of linearly independent rows. This means that if $\lambda\mathbf{I}_n - \mathbf{A}$ has rank less than n (i.e., λ is an eigenvalue of $\mathbf{A}$), then the left null space of $\lambda\mathbf{I}_n - \mathbf{A}$ has the same dimension as the null space. Thus, there exists a row vector $\mathbf{w}'$ such that $\mathbf{w}'(\lambda\mathbf{I}_n - \mathbf{A}) = \mathbf{0}$, or equivalently, $\mathbf{w}'\mathbf{A} = \lambda\mathbf{w}'$. Thus, the vector $\mathbf{w}'$ is called the *left-eigenvector* of $\mathbf{A}$ corresponding to the eigenvalue λ.

The Gersgorin Disc Theorem

It will often be very useful to get a general idea of where the eigenvalues of a matrix are located without actually finding all of them. The following result provides a method to accomplish this.

Theorem A.6 (Gersgorin Disc Theorem)**.** *Consider a matrix* $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{C})$*. For each row* i *of* $\mathbf{A}$*, define the region*

$$R_i = \left\{ z \in \mathbb{C} \;\middle|\; |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\}.$$

Then all eigenvalues of $\mathbf{A}$ *are contained in the union of these regions, i.e.,* $\cup_{i=1}^{n} R_i$*.*

Each of the regions R_i is a disc centered around a diagonal element of $\mathbf{A}$; the radius of the disc is the sum of the absolute values of the remaining elements in each row. Before proving the theorem, it will be useful to consider a simple example to illustrate these regions more tangibly.

Example A.1. Consider the matrix

$$\mathbf{A} = \begin{bmatrix} -2 & -1 & 2 \\ 0.5 & 0 & -0.5 \\ 0 & 1 & 5 \end{bmatrix}.$$

The disc R_1 is centered at -2, and contains all points within a radius $|-1| + |2| = 3$. The region R_2 is centered at 0 and contains all points within a radius of 1. The disc R_3 is centered at 5, and contains all points within a radius of 1. These three discs are illustrated in the following figure. All eigenvalues of $\mathbf{A}$ are contained in the union of these discs.

One can verify that the eigenvalues of $\mathbf{A}$ are -1.6148, -0.3141 and 4.9289, which are indeed within the given regions.

We now present the following proof from [39] for the Gersgorin Disc Theorem.

Proof. Let λ be an eigenvalue of $\mathbf{A}$, with eigenvector $\mathbf{v}$. In the proof, we will use the notation $[\mathbf{v}]_i$ to denote the i–th element of the vector $\mathbf{v}$. Let p denote the location in $\mathbf{v}$ that has largest magnitude (i.e., $|[\mathbf{v}]_p| \geq |[\mathbf{v}]_i|$ for all $i \neq p$). Then, we have

$$\lambda[\mathbf{v}]_p = [\lambda\mathbf{v}]_p = [\mathbf{A}\mathbf{v}]_p = \sum_{j=1}^{n} a_{pj}[\mathbf{v}]_j = \sum_{\substack{j=1 \\ j\neq p}}^{n} a_{pj}[\mathbf{v}]_j + a_{pp}[\mathbf{v}]_p.$$

Moving $a_{pp}[\mathbf{v}]_p$ to the left-hand side, we obtain

$$(\lambda - a_{pp})[\mathbf{v}]_p = \sum_{\substack{j=1 \\ j\neq p}}^{n} a_{pj}[\mathbf{v}]_j.$$

Taking the magnitude of both sides, we have

$$|(\lambda - a_{pp})[\mathbf{v}]_p| = |\sum_{\substack{j=1 \\ j\neq p}}^{n} a_{pj}[\mathbf{v}]_j| \leq \sum_{\substack{j=1 \\ j\neq p}}^{n} |a_{pj}[\mathbf{v}]_j| = \sum_{\substack{j=1 \\ j\neq p}}^{n} |a_{pj}||[\mathbf{v}]_j| \leq \sum_{\substack{j=1 \\ j\neq p}}^{n} |a_{pj}||[\mathbf{v}]_p|.$$

The first inequality follows from the triangle inequality, and the last inequality follows from the fact that $|[\mathbf{v}]_p|$ is the maximum value in $\mathbf{v}$. Since $|(\lambda - a_{pp})[\mathbf{v}]_p| = |(\lambda - a_{pp})||[\mathbf{v}]_p|$, we can divide both sides by $|[\mathbf{v}]_p|$ to obtain

$$|(\lambda - a_{pp})| \leq \sum_{\substack{j=1 \\ j \neq p}}^{n} |a_{pj}|.$$

In other words, the eigenvalue is located inside a disc centered at a_{pp}, with radius equal to the sum of the magnitudes of the other elements in the p–th row. Since we do not know p a priori (i.e., we do not know the eigenvector $\mathbf{v}$ and the location of its largest element), we can only say that the eigenvalue must lie inside one of the discs centered at each diagonal element in $\mathbf{A}$; this is given by the union of the regions R_i defined in the theorem. □

A.4.5 Similarity Transformations, Diagonal and Jordan Forms

Given a matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{F})$ and an invertible matrix $\mathbf{T} \in \mathcal{M}_{n,n}(\mathbb{F})$, a *similarity transformation* on $\mathbf{A}$ is defined as the matrix

$$\hat{\mathbf{A}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}.$$

One important feature of the similarity transformation is that it does not change the eigenvalues of the matrix. To see this, note that

$$\begin{aligned}
\det\left(\lambda\mathbf{I}_n - \hat{\mathbf{A}}\right) = \det\left(\lambda\mathbf{I}_n - \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\right) &= \det\left(\lambda\mathbf{T}^{-1}\mathbf{T} - \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\right) \\
&= \det\left(\mathbf{T}^{-1}\left(\lambda\mathbf{I}_n - \mathbf{A}\right)\mathbf{T}\right) \\
&= \det\left(\mathbf{T}^{-1}\right)\det\left(\lambda\mathbf{I}_n - \mathbf{A}\right)\det\left(\mathbf{T}\right) \\
&= \det\left(\lambda\mathbf{I}_n - \mathbf{A}\right).
\end{aligned}$$

In the last two steps above, we have used items 3 and 4 from Theorem A.5. Thus the values of λ for which $\det\left(\lambda\mathbf{I}_n - \mathbf{A}\right) = 0$ are exactly the same values of λ for which $\det\left(\lambda\mathbf{I}_n - \hat{\mathbf{A}}\right) = 0$, meaning that both matrices have exactly the same eigenvalues.

Via an appropriate choice of $\mathbf{T}$, one can transform the matrix $\mathbf{A}$ into very useful forms. For example, suppose that $\mathbf{A}$ has n linearly independent eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. This means that the matrix $\mathbf{T} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n\end{bmatrix}$ is invertible. Now note that

$$\begin{aligned}
\mathbf{A}\mathbf{T} = \mathbf{A}\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n\end{bmatrix} &= \begin{bmatrix}\mathbf{A}\mathbf{v}_1 & \mathbf{A}\mathbf{v}_2 & \cdots & \mathbf{A}\mathbf{v}_n\end{bmatrix} \\
&= \begin{bmatrix}\lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \cdots & \lambda_n\mathbf{v}_n\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n\end{bmatrix}.
\end{aligned}$$

Multiplying both sides of the above equation on the left by $\mathbf{T}^{-1}$, we obtain

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}.$$

This is called the *diagonal* form of matrix $\mathbf{A}$. If a given matrix $\mathbf{A}$ can be put into this form, it is said to be *diagonalizable*.

Recall from the discussion on eigenvalues and eigenvectors that a given matrix $\mathbf{A}$ does not necessarily have n linearly independent eigenvectors (i.e., if the geometric multiplicity of some eigenvalues is less than their algebraic multiplicity). In this case, the matrix $\mathbf{A}$ is not diagonalizable. One can, however, find a form that works for *all* matrices over the complex field. Specifically, for any eigenvalue λ, define the square matrix

$$\mathbf{J}_\lambda = \begin{bmatrix} \lambda & 1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & 0 & \cdots & 0 \\ 0 & 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & \lambda \end{bmatrix}.$$

In words, $\mathbf{J}_\lambda$ is a matrix with λ on the diagonal and 1 on each entry directly above the diagonal, but zeros everywhere else. This matrix is called a *Jordan block* corresponding to λ.

We will now state the following result; the proof of the result can be found in [39].

Theorem A.7. *Given any matrix $\mathbf{A} \in \mathcal{M}_{n,n}(\mathbb{C})$, there exists an invertible matrix $\mathbf{T} \in \mathcal{M}_{n,n}(\mathbb{C})$ such that*

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \begin{bmatrix} \mathbf{J}_{\lambda_1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\lambda_2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_{\lambda_3} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{J}_{\lambda_r} \end{bmatrix},$$

where each $\mathbf{J}_{\lambda_i}$ is a Jordan block corresponding to an eigenvalue λ_i of $\mathbf{A}$. The Jordan blocks in the above decomposition can be of different sizes. Furthermore, the number of Jordan blocks corresponding to any given eigenvalue λ is equal to the geometric multiplicity of that eigenvalue.

The above decomposition is known as the *Jordan form* of matrix $\mathbf{A}$. Note that if every Jordan block has size 1, then the Jordan form is simply a diagonal matrix, which corresponds to the case of diagonalizability studied earlier.

A.4.6 Symmetric and Definite Matrices

A matrix $\mathbf{P} \in \mathcal{M}_{n,n}(\mathbb{R})$ is said to be *symmetric* if $\mathbf{P} = \mathbf{P}'$.

> **Theorem A.8.** *Every symmetric $n \times n$ matrix has n real eigenvalues, with n linearly independent eigenvectors.*

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of $\mathbf{P}$, with eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Suppose that $\mathbf{v}_i'\mathbf{v}_i = \gamma_i$ for some positive γ_i. Then, the vector $\frac{1}{\sqrt{\gamma_i}}\mathbf{v}_i$ is also an eigenvector, and $\frac{1}{\sqrt{\gamma_i}}\mathbf{v}_i'\frac{1}{\sqrt{\gamma_i}}\mathbf{v}_i = 1$. Thus, we can assume without loss of generality that $\mathbf{v}_i'\mathbf{v}_i = 1$ for all $i \in \{1, 2, \ldots, n\}$ (otherwise, we just scale the vectors as described).

Next, note that

$$\mathbf{v}_i'\mathbf{P} = (\mathbf{P}'\mathbf{v}_i)' = (\mathbf{P}\mathbf{v}_i)' = (\lambda\mathbf{v}_i)' = \lambda\mathbf{v}_i',$$

and thus $\mathbf{v}_i'$ is a left-eigenvector of $\mathbf{P}$. Let $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n\end{bmatrix}$, and recall from the last section that

$$\mathbf{P}\mathbf{V} = \mathbf{V}\Lambda,$$

where Λ is a diagonal matrix with the eigenvalues on the diagonal. This is equivalent to writing

$$\mathbf{P} = \mathbf{V}\Lambda\mathbf{V}^{-1} \Leftrightarrow \mathbf{V}^{-1}\mathbf{P} = \Lambda\mathbf{V}^{-1}.$$

Denote the rows of $\mathbf{V}^{-1}$ by $\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n$; the above equation can then be written as

$$\begin{bmatrix}\bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_2 \\ \vdots \\ \bar{\mathbf{v}}_n\end{bmatrix}\mathbf{P} = \begin{bmatrix}\lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n\end{bmatrix}\begin{bmatrix}\bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_2 \\ \vdots \\ \bar{\mathbf{v}}_n\end{bmatrix},$$

which yields $\bar{\mathbf{v}}_i\mathbf{P} = \lambda_i\bar{\mathbf{v}}_i$ for $i \in \{1, 2, \ldots, n\}$. Thus, $\bar{\mathbf{v}}_i$ is a left eigenvector of $\mathbf{P}$ corresponding to eigenvalue λ_i. Above, we saw that $\mathbf{v}_i'$ is a left eigenvector corresponding to eigenvalue λ_i. Thus, we have

$$\mathbf{V}^{-1} = \begin{bmatrix}\mathbf{v}_1' \\ \mathbf{v}_2' \\ \vdots \\ \mathbf{v}_n'\end{bmatrix} = \mathbf{V}'.$$

The above analysis is encapsulated in the following result.

Theorem A.9. *Let $\mathbf{P} \in \mathcal{M}_{n,n}(\mathbb{R})$ be a symmetric matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, and eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, scaled so that $\mathbf{v}_i'\mathbf{v}_i = 1$ for $i \in \{1, 2, \ldots, n\}$. Let $\mathbf{V}$ be an $n \times n$ matrix whose columns are the eigenvectors. Then*

$$\mathbf{P} = \mathbf{V}\Lambda\mathbf{V}'$$

where Λ is a diagonal matrix with the eigenvalues on the diagonal.

Note that the difference between the diagonalization result from the previous section for general $n \times n$ matrices and this theorem is that the matrix of eigenvectors can be inverted simply by transposing it in the latter case.

A special kind of symmetric matrix is given by the following definition.

Definition A.1. A symmetric $n \times n$ matrix $\mathbf{P}$ is said to be *positive definite* if, for any vector $\mathbf{a} \in \mathbb{R}^n$,

$$\mathbf{a}'\mathbf{P}\mathbf{a} \geq 0,$$

with equality holding if and only if $\mathbf{a} = \mathbf{0}$. Similarly, a symmetric matrix $\mathbf{P}$ is said to be *negative definite* if, for any vector $\mathbf{a} \in \mathbb{R}^n$,

$$\mathbf{a}'\mathbf{P}\mathbf{a} \leq 0,$$

with equality holding if and only if $\mathbf{a} = \mathbf{0}$. The notation $\mathbf{P} \succ 0$ is used to indicate a positive definite matrix, and $\mathbf{P} \prec 0$ is used to denote a negative definite matrix.

Similarly, for two symmetric matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, the notation $\mathbf{P}_1 \succ \mathbf{P}_2$ is used to indicate that $\mathbf{a}'\mathbf{P}_1\mathbf{a} \geq \mathbf{a}'\mathbf{P}_2\mathbf{a}$ for all $\mathbf{a}$, or equivalently, that the matrix $\mathbf{P}_1 - \mathbf{P}_2$ is positive definite. Similarly, the notation $\mathbf{P}_1 \prec \mathbf{P}_2$ is used to indicate that $\mathbf{P}_1 - \mathbf{P}_2$ is negative definite.

Theorem A.10. *All eigenvalues of a positive definite matrix are positive. All eigenvalues of a negative definite matrix are negative.*

Proof. Let $\mathbf{P}$ be a positive definite matrix, and suppose that it has a negative eigenvalue λ, with corresponding eigenvector $\mathbf{v}$. Then we have

$$\mathbf{v}'\mathbf{P}\mathbf{v} = \mathbf{v}'\lambda\mathbf{v} = \lambda\mathbf{v}'\mathbf{v},$$

and since $\mathbf{v}'\mathbf{v}$ is positive (it is just the sum of squares of the elements of $\mathbf{v}$), we see that $\mathbf{v}'\mathbf{P}\mathbf{v} < 0$, which contradicts the assumption that $\mathbf{P}$ is positive definite. Similarly, if

$\lambda = 0$, then $\mathbf{v}'\mathbf{P}\mathbf{v} = 0$ for a nonzero vector $\mathbf{v}$, again contradicting the assumption of a positive definite matrix. Thus, $\mathbf{P}$ can only have positive eigenvalues. The proof for negative definite matrices is completely analogous. □

Theorem A.11. *Let $\mathbf{T} \in \mathcal{M}_{m,n}(\mathbb{R})$ be a matrix that has rank m, and let $\mathbf{P} \in \mathcal{M}_{n,n}(\mathbb{R})$ be a positive definite matrix. Then*

$$\mathbf{T}\mathbf{P}\mathbf{T}' \succ 0.$$

Proof. For any vector $\mathbf{a} \in \mathbb{R}^m$, we have

$$\mathbf{a}'\mathbf{T}\mathbf{P}\mathbf{T}'\mathbf{a} = (\mathbf{a}'\mathbf{T})\mathbf{P}(\mathbf{a}'T)' \geq 0$$

with equality if and only if $\mathbf{a}'\mathbf{T} = \mathbf{0}$ (since $\mathbf{P}$ is positive definite). Next, note that since $\mathbf{T}$ has full row rank, $\mathbf{a}'\mathbf{T} = \mathbf{0}$ if and only if $\mathbf{a}' = \mathbf{0}$. Thus, we have that $\mathbf{a}'\mathbf{T}\mathbf{P}\mathbf{T}'\mathbf{a} \geq 0$ with equality if and only if $\mathbf{a}' = \mathbf{0}$. □

It is often useful to convert a given positive definite matrix into a positive definite matrix of a different form. The following result provides one mechanism by which to do this.

Theorem A.12. *For any symmetric matrix $\mathbf{Q} \in \mathcal{M}_{m,m}(\mathbb{R})$, positive definite matrix $\mathbf{R} \in \mathcal{M}_{n,n}(\mathbb{R})$, and matrix $\mathbf{S} \in \mathcal{M}_{m,n}(\mathbb{R})$,*

$$\begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}' & \mathbf{R} \end{bmatrix} \succ 0$$

if and only if

$$\mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}' \succ 0.$$

This operation is known as the Schur Complement.

Proof. First, note that we can write [39]

$$\begin{bmatrix} \mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}' & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I}_m & -\mathbf{S}\mathbf{R}^{-1} \\ \mathbf{0} & \mathbf{I}_n \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}' & \mathbf{R} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\mathbf{R}^{-1}\mathbf{S}' & \mathbf{I}_n \end{bmatrix}}_{\mathbf{T}'}.$$

Since the matrix $\mathbf{T}$ is invertible, we use Theorem A.11 to note that the matrix on the left hand side of the above equation is positive definite. Furthermore, since this matrix is block diagonal, each of the individual blocks must be positive definite as well. Thus, we have $\mathbf{R} \succ 0$ (as assumed), and $\mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}' \succ 0$. □

Both Theorem A.11 and A.12 can be applied to negative definite matrices as well; the proofs are completely analogous.

A.4.7 Vector Norms

It is often useful to measure the 'size' of a given vector in a vector space $(\mathbb{V}, \mathbb{R})$. However, the metric that is used to accomplish this must be carefully defined in order to be useful. The notion of a vector *norm* captures this.

Definition A.2. Consider a vector space $(\mathbb{V}, \mathbb{R})$, and let ρ be a function that maps vectors $\mathbf{v} \in \mathbb{V}$ to the nonnegative real numbers. The function $\rho(\cdot)$ is said to be a *vector norm* if it satisfies the following properties.

1. $\rho(c\mathbf{a}) = |c|\rho(\mathbf{a})$ for all $c \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{V}$.
2. $\rho(\mathbf{a} + \mathbf{b}) \leq \rho(\mathbf{a}) + \rho(\mathbf{b})$ for all $\mathbf{a}, \mathbf{b} \in \mathbb{V}$.
3. $\rho(\mathbf{a}) = 0$ if and only if $\mathbf{a} = \mathbf{0}$.

A commonly used norm is given by the following definition.

Definition A.3. For any positive integer r, the r-norm of a given vector $\mathbf{a} \in \mathbb{R}^n$ is defined as

$$\|\mathbf{a}\|_r = (|a_1|^r + |a_2|^r + \cdots + |a_n|^r)^{\frac{1}{r}}.$$

When $r = 1$, this is just the sum of the absolute values of the elements of the vector, and when $r = 2$, this is the square root of the sum of squares of the elements. Note that we can equivalently write

$$\|\mathbf{a}\|_2^2 = \mathbf{a}'\mathbf{a} = a_1^2 + a_2^2 + \cdots + a_n^2.$$

When $r = \infty$, the norm just produces the largest (in magnitude) element of the vector. We will also find it useful to consider the following generalization of the 2-norm.

Definition A.4. For any $n \times n$ positive definite matrix $\mathbf{P}$, the $\mathbf{P}$-norm of a given vector $\mathbf{a} \in \mathbb{R}^n$ is

$$\|\mathbf{a}\|_{\mathbf{P}} = (\mathbf{a}'\mathbf{P}\mathbf{a})^{\frac{1}{2}}.$$

Note that the $\mathbf{P}$-norm is guaranteed to be nonnegative, and equal to zero only if $\mathbf{a} = \mathbf{0}$ (by the definition of a positive definite matrix). We can provide upper and lower bounds on the $\mathbf{P}$-norm of a given vector as follows.

Theorem A.13. *Let* $\mathbf{P}$ *be a positive definite matrix, and let* $\lambda_{\max}(\mathbf{P})$ *and* $\lambda_{\min}(\mathbf{P})$ *be the largest and smallest eigenvalues of* $\mathbf{P}$*, respectively. Then for any vector* $\mathbf{a} \in \mathbb{R}^n$*,*

$$\lambda_{\min}(\mathbf{P})\|\mathbf{a}\|_2^2 \leq \|\mathbf{a}\|_{\mathbf{P}}^2 \leq \lambda_{\max}(\mathbf{P})\|\mathbf{a}\|_2^2.$$

Proof. From Theorem A.9, we can write $\mathbf{P} = \mathbf{V}\Lambda\mathbf{V}'$, where Λ is a diagonal matrix containing the eigenvalues of $\mathbf{P}$. Next, note that

$$\|\mathbf{a}\|_{\mathbf{P}}^2 = \mathbf{a}'\mathbf{P}\mathbf{a} = \mathbf{a}'\mathbf{V}\Lambda\mathbf{V}'\mathbf{a}.$$

Define $\mathbf{z} = \mathbf{V}'\mathbf{a}$, so that

$$\|\mathbf{a}\|_{\mathbf{P}}^2 = \mathbf{z}'\Lambda\mathbf{z} = \begin{bmatrix} z_1 & z_2 & \cdots & z_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \sum_{i=1}^n \lambda_i z_i^2.$$

Since $\lambda_{\max}(\mathbf{P})$ and $\lambda_{\min}(\mathbf{P})$ are the largest and smallest eigenvalues, respectively, we have

$$\sum_{i=1}^n \lambda_{\min}(\mathbf{P}) z_i^2 \leq \sum_{i=1}^n \lambda_i z_i^2 \leq \sum_{i=1}^n \lambda_{\max}(\mathbf{P}) z_i^2,$$

or equivalently

$$\lambda_{\min}(\mathbf{P})\|\mathbf{z}\|_2^2 \leq \|\mathbf{a}\|_{\mathbf{P}}^2 \leq \lambda_{\max}(\mathbf{P})\|\mathbf{z}\|_2^2.$$

Finally, note that

$$\|\mathbf{z}\|_2^2 = \|\mathbf{V}'\mathbf{a}\|_2^2 = \mathbf{a}'\mathbf{V}\mathbf{V}'\mathbf{a} = \mathbf{a}'\mathbf{a} = \|\mathbf{a}\|_2^2,$$

which proves the result. □

Appendix B

Graph Theory

A graph $\mathcal{G}$ is defined by a set of vertices (or nodes) $\mathcal{X} = \{x_1, \ldots, x_n\}$, and a set of edges between the vertices $\mathcal{E}$. Thus, a graph is given by the pair $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$. Graphs are commonly depicted visually by using circles for the nodes, and arrows for the edges.

Example B.1. Consider the vertex set $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, and the edge set

$$\mathcal{E} = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_1\}, \{x_4, x_1\}, \{x_4, x_4\}\}.$$

The graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$ is drawn as follows.

If, $\forall x_i, x_j \in \mathcal{X}$, $(x_i, x_j) \in \mathcal{E} \Leftrightarrow (x_j, x_i) \in \mathcal{E}$, the graph is said to be *undirected*. In words, this means that if x_i connects to x_j, then x_j should connect to x_i. Undirected graphs are frequently drawn by putting a single line between two connecting nodes, and dropping the arrows.

The nodes in the set $\mathcal{N}_i = \{x_j | (x_j, x_i) \in \mathcal{E}, x_j \neq x_i\}$ are said to be neighbors of node x_i. The number of neighbors of node x_i is called the *degree* of x_i, and is denoted by $\deg_i = |\mathcal{N}_i|$. If $(x_i, x_i) \in \mathcal{E}$, then x_i is said to have a *self-loop* (but this is not counted in the degree of x_i). In the example above, $\deg_1 = 2$, $\deg_2 = 1$, $\deg_3 = 2$ and $\deg_4 = 1$.

A *subgraph* of $\mathcal{G}$ is a graph $\mathcal{H} = \{\bar{\mathcal{X}}, \bar{\mathcal{E}}\}$, with $\bar{\mathcal{X}} \subseteq \mathcal{X}$ and $\bar{\mathcal{E}} \subseteq \mathcal{E}$ (where all edges in $\bar{\mathcal{E}}$ are between vertices in $\bar{\mathcal{X}}$). The subgraph $\mathcal{H}$ is said to be *induced* if it is obtained from $\mathcal{G}$ by deleting a set of vertices (and all edges coming into and out of those vertices),

but leaving all other edges intact. The subgraph $\mathcal{H}$ is called *spanning* if it contains all vertices of $\mathcal{G}$ (i.e., $\bar{\mathcal{X}} = \mathcal{X}$).

Example B.2. Consider the following four graphs $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ and $\mathcal{H}_4$.

The first graph $\mathcal{H}_1$ is a subgraph of $\mathcal{G}$, since it contains only vertices and edges that appear in $\mathcal{G}$ (e.g., either by looking at vertices x_1, x_2, x_3 or x_1, x_2, x_4 in the original graph). However, it is not spanning (it only contains three vertices), and it is not induced – if we delete any single vertex in $\mathcal{G}$, we do not get the graph $\mathcal{H}_1$.

The graph $\mathcal{H}_2$ is a spanning subgraph of $\mathcal{G}$, but it is not induced (it is missing several edges).

The graph $\mathcal{H}_3$ is an induced subgraph of $\mathcal{G}$ (obtained by deleting vertex x_4), but it is not spanning.

The graph $\mathcal{H}_4$ is not a subgraph of $\mathcal{G}$. The easiest way to see this is to note that starting from any vertex in $\mathcal{H}_4$, we can follow the edges through all of the other vertices and get back to the starting vertex. However, this is not possible in $\mathcal{G}$.

B.1 Paths and Connectivity

A *path* P from vertex x_{i_0} to vertex x_{i_t} is a sequence of vertices $x_{i_0} x_{i_1} \cdots x_{i_t}$ such that $(x_{i_j}, x_{i_{j+1}}) \in \mathcal{E}$ for $0 \leq j \leq t-1$. The nonnegative integer t is called the length of the path. A path is called a *cycle* if its start vertex and end vertex are the same, and no other vertex appears more than once in the path. Paths P_1 and P_2 are *vertex disjoint* if they have no vertices in common. Similarly, paths P_1 and P_2 are *internally vertex disjoint* if they have no vertices in common, with the possible exception of the end points. A set of paths $P_1, P_2, \ldots, P_r$ are (internally) vertex disjoint if the paths are pairwise (internally) vertex disjoint. Given two subsets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$, a set of r vertex disjoint paths, each with start vertex in $\mathcal{X}_1$ and end vertex in $\mathcal{X}_2$, is called an *r-linking* from $\mathcal{X}_1$ to $\mathcal{X}_2$.[1] Note that if $\mathcal{X}_1$ and $\mathcal{X}_2$ are not disjoint, we will take each of their common vertices to be a vertex disjoint path between $\mathcal{X}_1$ and $\mathcal{X}_2$ of length zero.

A graph is said to be *strongly connected* if there is a path between vertices x_i and x_j for every $x_i, x_j \in \mathcal{X}$. We will call a graph *disconnected* if there exists at least one pair

[1] There are various algorithms to find linkings, such as the Ford-Fulkerson algorithm, which has run-time polynomial in the number of vertices [93, 23].

of vertices $x_i, x_j \in \mathcal{X}$ such that there is no path from x_i to x_j. A *vertex cut* in a graph is a subset $\mathcal{S} \subset \mathcal{X}$ such that removing the vertices in $\mathcal{S}$ (and the associated edges) from the graph causes the graph to be disconnected. More specifically, an (i,j)-cut is a set $\mathcal{S}_{ij} \subset \mathcal{X}$ whose removal causes all paths from node x_i to x_j to be eliminated. The size of the smallest (i,j)-cut is denoted by κ_{ij}. A graph is said to be κ-connected if every vertex cut has cardinality at least κ. The *connectivity* of a graph is the smallest size of a vertex cut, i.e.,

$$\kappa = \min_{i,j} \kappa_{ij}.$$

Note that if a graph is κ-connected, the in-degree of every node must be at least κ (since otherwise, removing all the neighbors of the offending node would disconnect the graph, thereby producing a vertex cut of size less than κ).

The following classical result will play an important role in our derivations (e.g., see [93]).

> **Theorem B.1** (Menger)**.** *A graph $\mathcal{G}$ is κ-connected if and only if there are at least κ internally vertex disjoint paths between any two non-adjacent vertices in the graph.*

Note that the above theorem specifies that connectivity implies disjoint paths between *non-adjacent* vertices; if vertex v_i is a neighbor of v_j, then removing any number of other vertices will not remove the direct link between v_i and v_j, and thus the notion of an (i,j)-cut is not defined.

Menger's theorem can be used to prove the following useful lemma.

> **Lemma B.1** (Expansion Lemma)**.** *Suppose that graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is κ-connected. Consider the graph $\mathcal{G}'$ obtained by adding a new vertex v to $\mathcal{G}$, and adding κ incoming and outgoing edges between v and κ different vertices in $\mathcal{V}$. Then $\mathcal{G}'$ is κ-connected.*

In turn, the Expansion Lemma produces the following result.

> **Theorem B.2.** *Suppose $\mathcal{G}$ is κ-connected. Given any two subsets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$, with $|\mathcal{X}_1| \geq \kappa$, $|\mathcal{X}_2| \geq \kappa$, there exists a κ-linking from $\mathcal{X}_1$ to $\mathcal{X}_2$.*

Note that some of the paths in this linking might have zero length (i.e., if $\mathcal{X}_1 \cap \mathcal{X}_2$ is nonempty). Further details can be found in standard texts such as [93].

B.2 Matrix Representations of Graphs

Any graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$ with n nodes can be represented by an $n \times n$ *adjacency* matrix $\mathbf{A}$, where $a_{ij} = 0$ if $(x_j, x_i) \notin \mathcal{E}$, and $a_{ij} > 0$ otherwise ($a_{ij} = 1$ is frequently used to indicate the presence of an edge).

Example B.3. Consider the graph of Example B.1. The adjacency matrix for this graph is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

The adjacency matrix captures all of the edges in the graph; more generally, one can say that $a_{ij} > 0$ if and only if there is a path of length 1 from x_j to x_i. The adjacency matrix can also be used to infer paths of longer length in the graph, as shown by the following result.

Lemma B.2. *Given an adjacency matrix $\mathbf{A}$ for a graph $\mathcal{G}$, entry (i, j) of the matrix $\mathbf{A}^k$ is positive if and only if there is a path of length k from x_j to x_i in $\mathcal{G}$.*

Proof. The proof is obvious for $k = 1$. For $k = 2$, entry (i, j) of matrix $\mathbf{A}^2$ is given by

$$\sum_{l=1}^{n} a_{il} a_{lj}.$$

Since all elements of $\mathbf{A}$ are nonnegative, we see that entry (i, j) is positive if and only if there is at least one $l \in \{1, 2, \ldots, n\}$ such that $a_{il} a_{lj} \neq 0$. In the graph, this means that there is an edge from node x_j to x_l, and edge from node x_l to x_i. Thus, entry (i, j) is positive if and only if there is a path of length two from x_j to x_i.

To prove for $k > 2$, we will use induction. Specifically, assume that entry (i, j) of the matrix $\mathbf{A}^{k-1}$ is positive if and only if there is a path of length $k-1$ from x_j to x_i. Denote $\mathbf{A}^{k-1} = \bar{\mathbf{A}}$, and note that $\mathbf{A}^k = \mathbf{A}\mathbf{A}^{k-1} = \mathbf{A}\bar{\mathbf{A}}$. Then, entry (i, j) of matrix $\mathbf{A}^k$ is given by

$$\sum_{l=1}^{n} a_{il} \bar{a}_{lj},$$

where $\bar{a}_{lj}$ is entry (l, j) of matrix $\bar{\mathbf{A}}$. Once again, entry (i, j) of $\mathbf{A}^k$ is positive if and only if $a_{il}\bar{a}_{lj} > 0$ for at least one $l \in \{1, 2, \ldots, n\}$. Since $\bar{a}_{lj} > 0$ if and only if there is path of length $k-1$ from x_j to x_l, we see that $a_{il}\bar{a}_{lj} > 0$ if and only if there is a path of length k from x_j to x_i. □

Example B.4. Consider the following graph.

The adjacency matrix of this graph is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

For $k = 2, 3, 4, \ldots$, we have

$$\mathbf{A}^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \mathbf{A}^3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathbf{A}, \ \mathbf{A}^4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{A}^2, \ \cdots$$

The above matrices indicate that $\mathcal{G}$ has no paths from x_2 to x_1 (or from x_1 to x_2) of even length. Similarly, there are no paths from each node to itself of odd length.

The above example illustrates that if element (i, j) of matrix $\mathbf{A}^k$ is positive for some k, it is not necessarily the case that the element is positive for all powers of $\mathbf{A}$ larger than k. However, in some graphs, we can in fact make this claim.

Lemma B.3. *Let $\mathcal{G}$ be a graph with n nodes, where every node has a self-loop. Let $\mathbf{A}$ be the adjacency matrix for the graph. Then if entry (i, j) of $\mathbf{A}^k$ is positive for some k, then entry (i, j) of $\mathbf{A}^{k+r}$ will also be positive for all integers $r \geq 0$.*

Proof. If entry (i, j) of $\mathbf{A}^k$ is positive, then there is a path of length k from x_j to x_i (from Lemma B.2). Now, for any positive integer r, we can form a path of length $k + r$ from x_j to x_i simply by starting at x_j, taking the self-loop to get back to x_j r times, and then taking the path of length k to x_i (this is not unique). Thus, entry (i, j) of $\mathbf{A}^{k+r}$ will also be positive, for any $r > 0$. □

The above results lead to the following useful corollary.

Corollary B.1. *Let $\mathcal{G}$ be a strongly connected graph with n nodes, where every node has a self-loop. Let $\mathbf{A}$ be the adjacency matrix for the graph. Then there exists some positive integer $k \leq n-1$ such that every element of $\mathbf{A}^{k+r}$ is positive, for all integers $r \geq 0$.*

Proof. Since the graph is strongly connected, there is a path between any two nodes x_j and x_i; let d_{ji} denote the length of the shortest path between the two nodes. Since there are only n nodes in the graph, it must be the case that $d_{ji} < n-1$. Furthermore, since each node has a self-loop, there is a path of length $d_{ij} + r$ between x_j and x_i, for any integer $r \geq 0$. Let

$$k = \max_{1 \leq i,j \leq n} d_{ji}$$

be the largest of the shortest paths between any two nodes. Then, there is a path of length k between any specific nodes x_j and x_i, and so every element of $\mathbf{A}^k$ will be positive. By Lemma B.3, we see that all entries of $\mathbf{A}^{k+r}$ will be positive for any integer $r \geq 0$. □

Appendix C

Structured System Theory

Linear systems are often studied from an *algebraic* perspective, based on the *rank* of certain matrices. While such tests are easy to derive from the mathematical model, they may have certain drawbacks. First, for large systems, checking the rank of the observability, controllability, or invertibility matrix may become infeasible, or computationally unreliable (due to the entries becoming large, or due to round-off errors in computations). Second, it is difficult to relate rank-based tests to underlying physical properties of the system itself; for example, if a system is *not* controllable, how should one modify the system in order to make it controllable? To deal with these questions, researchers started developing alternative tests in the 1970's by using the underlying *structure* of the system, and applying *graph-theory* to identify whether certain properties hold [55]. This perspective provides a great deal of intuition, and will be very useful for identifying the key aspects of linear systems that are required for certain properties to hold.

C.1 Structured Systems

Consider a linear system of the form

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] \ , \end{aligned} \tag{C.1}$$

with $\mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$. Rather than worrying about the exact numerical values of the system matrices, we will instead consider only the *structure* of the system, that is, which entries in the matrices are zero, and which entries are not zero. More specifically, by replacing all of the nonzero entries with independent free parameters, we obtain a *structured system* [55, 40, 66, 18]. We can now ask how the zero/non-zero structure of the system relates to system properties. We will use the following definition.

Definition C.1. A structured system is said to have a certain property (such as controllability, observability, invertibility, etc.) if that property holds for at least one numerical choice of free parameters in the system.

Example C.1. Consider the structured linear system

$$\begin{aligned} \mathbf{x}[k+1] &= \begin{bmatrix} \lambda_1 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \\ 0 & 0 & \lambda_4 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 0 \\ 0 \\ \lambda_5 \end{bmatrix} u[k] \\ \mathbf{y}[k] &= \begin{bmatrix} 0 & 0 & \lambda_6 \end{bmatrix} \mathbf{x}[k] + \lambda_7 u[k], \end{aligned} \tag{C.2}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_7$ are independent free parameters. The system is *structurally controllable* if we can find some values for these parameters so that the resulting numerical system is controllable. This is indeed possible; for example, use

$$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 0, \lambda_5 = 1.$$

However, one cannot find any assignment of free parameters for which the system will be observable, because the observability matrix is given by

$$\mathcal{O}_L = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^L \end{bmatrix} = \begin{bmatrix} 0 & 0 & \lambda_6 \\ 0 & 0 & \lambda_6\lambda_4 \\ 0 & 0 & \lambda_6\lambda_4^2 \\ \vdots & \vdots & \vdots \\ 0 & 0 & \lambda_6\lambda_4^L \end{bmatrix}.$$

This matrix has rank 1 for any L, and thus the system is not structurally observable. □

We will later see that structural properties are actually stronger than implied by the above definition – if a structural property holds for one choice of free parameters, it will hold for *almost any* choice of free parameters. In other words, it turns out that structural properties are *generic*. For example, these free parameters could be chosen 'randomly' from some continuous probability distribution, and the property will hold (as long as it holds for some choice of free parameters). We will see this more formally later.

Since only the zero/nonzero patterns in the system matrices are of interest in structured system theory, structured systems can be completely characterized by an associated graph $\mathcal{H}$ (see Appendix B for background on graph theory). The vertex set of $\mathcal{H}$ is given by $\mathcal{X} \cup \mathcal{U} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is a set of *state vertices*, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ is a set of *input vertices*, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_p\}$ is a set of *output vertices*. The edge set of $\mathcal{H}$ is given by $\mathcal{E}_{xx} \cup \mathcal{E}_{ux} \cup \mathcal{E}_{xy} \cup \mathcal{E}_{uy}$, where

- $\mathcal{E}_{xx} = \{(x_j, x_i) \mid \mathbf{A}_{ij} \neq 0\}$ is the set of edges corresponding to interconnections between the state vertices,
- $\mathcal{E}_{ux} = \{(u_j, x_i) \mid \mathbf{B}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the state vertices,
- $\mathcal{E}_{xy} = \{(x_j, y_i) \mid \mathbf{C}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the state vertices and the output vertices, and
- $\mathcal{E}_{uy} = \{(u_j, y_i) \mid \mathbf{D}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the output vertices.

Note that the number of edges in the graph is equal to the number of nonzero parameters in the system matrices.

Example C.2. Consider the structured system from Equation C.2. This system has $n = 3$ states, $m = 1$ input and $p = 1$ output. The associated graph $\mathcal{H}$ thus has $n + m + p = 5$ vertices, and 7 edges (since there are 7 free parameters in the system), and is shown in Fig. C.1. To see how this graph is generated, consider the state x_1 in (C.2). The value of this state at time-step $k+1$ depends on the values of states $x_1[k]$ and $x_2[k]$, but not on $x_3[k]$ or $u[k]$ (since the entries of the $\mathbf{A}$ and $\mathbf{B}$ matrices corresponding to those elements are zero). Thus, the vertex x_1 in $\mathcal{H}$ has incoming edges from state vertices x_1 and x_2. Similarly, the output of the system $y[k]$ depends on the third state and the input, and so there is an outgoing edge from the state vertex x_3 to the output vertex y, and also from the input vertex to the output vertex.

Figure C.1: The graph $\mathcal{H}$ associated with system in equation (C.2).

□

Note that the graph captures all of the information about the structured system; this indicates that one can identify if structured systems have certain properties by analyzing the graph $\mathcal{H}$. Before we delve into the details, we will warm up by seeing how one can use the graph of a square matrix to analyze the rank of that matrix.

C.2 Graphical Test for Nonsingularity

Just as we associated a graph with the linear system (C.1), we can also associate a graph $\mathcal{H}$ with a given $n \times n$ square matrix $\mathbf{A}$. In this case, the graph contains only the n state vertices (and the associated edges) described in the previous section.

Example C.3. Consider the structured matrix

$$\mathbf{A} = \begin{bmatrix} 0 & \lambda_1 & 0 & \lambda_2 \\ \lambda_3 & \lambda_4 & \lambda_5 & 0 \\ 0 & 0 & \lambda_6 & \lambda_7 \\ \lambda_8 & 0 & 0 & 0 \end{bmatrix}. \tag{C.3}$$

The graph associated with $\mathbf{A}$ is shown in Fig. C.2.

Figure C.2: The graph $\mathcal{H}$ associated with the matrix in equation (C.3).

□

Given a structured square matrix $\mathbf{A}$, is there a way to assign values to the free parameters so that the matrix is invertible? To answer this question, recall from Theorem A.5 in Section A.4.3 that a square matrix is invertible if and only if its determinant is nonzero. Furthermore, the determinant of a matrix is a polynomial in the elements of that matrix, and each term of the polynomial is a product of n elements, no two of which come from the same row or column. Since the elements of a structured matrix are either free parameters λ_i, or fixed zeros, the determinant will be a polynomial $f(\lambda_1, \lambda_2, \ldots, \lambda_r)$ (where r is the number of nonzero parameters). If the determinant is zero for all choices of the free parameters (i.e., there is no choice of free parameters that makes the determinant nonzero), then this polynomial will be identically zero. On the other hand, if the polynomial is a nonzero function of the free parameters, then any numerical choice of parameters that causes the polynomial to be a nonzero number will cause the matrix to be invertible. To more explicitly characterize the values of the parameters for which the determinant will be zero, we will require the notion of an *algebraic variety*, defined below.

Definition C.2. [91] Let $\mathcal{F} = \{f_i(\lambda_1, \lambda_2, \cdots, \lambda_r)\}$ be a finite set of polynomials in the parameters $\lambda_1, \lambda_2, \ldots, \lambda_r \in \mathbb{R}$. The set

$$\mathcal{S} = \{(\lambda_1^*, \lambda_2^*, \ldots, \lambda_r^*) \mid f_i(\lambda_1^*, \lambda_2^*, \ldots, \lambda_r^*) = 0 \ \forall f_i(\cdot) \in \mathcal{F}\}.$$

is called a *variety*. It is called a proper variety if $\mathcal{S}$ is not the entire space $\mathbb{R}^r$.

More formally, if the system has a structural property, then the set of free parameters for which the property does *not* hold lies on a proper variety (see [66, 18] for more details). Such varieties are said to have *measure zero* in the space of parameters; in other words, *almost every* choice of parameters will not be in the variety, and thus the property will hold for almost any choice of parameters.

Example C.4. The determinant of matrix $\mathbf{A}$ in (C.3) is equal to

$$\det(\mathbf{A}) = -\lambda_1\lambda_5\lambda_7\lambda_8 - \lambda_2\lambda_4\lambda_6\lambda_8 = f(\lambda_1, \lambda_2, \ldots, \lambda_8).$$

Thus, the matrix $\mathbf{A}$ will be invertible for all values of the free parameters $\lambda_1, \ldots, \lambda_r$, except for those that lie on an algebraic variety. □

Now consider how each term in the determinant is represented in the graph $\mathcal{H}$. Specifically, since each free parameter corresponds to an edge in the graph, each term in the determinant corresponds to a set of n edges. Consider any term (and associated set of n edges). Each row in the matrix has a single free parameter that appears in that term; this means that no two edges in the set enter the same state vertex. Furthermore, every column in the matrix has a single free parameter that appears in the term; this means that no two edges in the set exit the same state vertex. Thus, this set of n edges is such that each state vertex has exactly one incoming edge and one outgoing edge. It is easy to see that this corresponds to a set of *disjoint cycles* in the graph, going through all of the state vertices. Thus, each term in the determinant corresponds to a different set of cycles that cover all of the state vertices.

Example C.5. Consider the term $\lambda_1\lambda_5\lambda_7\lambda_8$ in $\det(\mathbf{A})$ for the matrix (C.3). The set of edges corresponding to the free parameters in this term is shown in Fig. C.3*a*. The set of edges corresponding to the term $\lambda_2\lambda_4\lambda_6\lambda_8$ is shown in Fig. C.3*b*. Both of these terms correspond to a set of disjoint cycles that cover all state vertices.

□

This brings us to the following result.

Figure C.3: Disjoint cycles in the graph corresponding to each term in the determinant. (a) The cycles corresponding to the term $\lambda_1\lambda_5\lambda_7\lambda_8$. ($b$) The cycles corresponding to the term $\lambda_2\lambda_4\lambda_6\lambda_8$.

Theorem C.1. *The determinant of a structured $n \times n$ matrix $\mathbf{A}$ is a nonzero polynomial in the free parameters if and only if the graph of $\mathbf{A}$ has a set of disjoint cycles that covers all state vertices.*

A corollary of the above result is that an $n \times n$ structured matrix will be invertible for almost any numerical choice of the free parameters if the graph of the matrix has a set of disjoint cycles that covers all state vertices, and it will not be invertible for any choice of free parameters otherwise.

Example C.6. Consider the matrix

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & \lambda_2 \\ 0 & 0 \end{bmatrix}. \tag{C.4}$$

The graph of this matrix is shown in Fig. C.4. Clearly, the above matrix has rank at most 1, and thus it cannot be invertible for any choice of free parameters. Indeed, its determinant is $\det(\mathbf{A}) = 0$ for all choices of λ_1 and λ_2. As we can see from the graph, there is no way to choose a cycle that involves the second state vertex (it does not even have an outgoing edge). □

Figure C.4: Graph of matrix $\mathbf{A}$ in equation (C.4).

C.3 Structural Properties of Linear Systems

We now return to the task of analyzing the graph $\mathcal{H}$ associated with a given structured linear system to determine whether the system possesses certain properties (such as observability, controllability or invertibility). We will start by considering graph-theoretic conditions to evaluate the generic rank of the matrix pencil for a structured linear system. This test will end up forming the basis of all other tests.

C.3.1 Generic Rank of the Matrix Pencil

Recall from Section 2.9.2 that the matrix pencil for system (C.1) is given by

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{A} - z\mathbf{I}_n & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}.$$

Definition C.3. The *generic normal rank* of the matrix pencil $\mathbf{P}(z)$ is the maximum rank of the matrix over all choices of free parameters in $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ and $z \in \mathbb{C}$.

The following result from [91, 90] characterizes the generic normal rank of $\mathbf{P}(z)$ in terms of the graph associated with the structured system.

Theorem C.2. *[91, 90] Let the graph of a structured linear system be given by $\mathcal{H}$. The generic normal rank of the matrix pencil $\mathbf{P}(z)$ is equal to $n + r$, where r is the size of the largest linking from the input vertices to the output vertices in $\mathcal{H}$.*

Note that the generic normal rank of the matrix pencil is at least n, since the matrix $\mathbf{A} - z\mathbf{I}$ will have generic rank n for any choice of parameters in $\mathbf{A}$ and any z that is not an eigenvalue of $\mathbf{A}$. Before we prove the above theorem, we will prove the following intermediate result.

Lemma C.1. *Suppose that the generic normal rank of $\mathbf{P}(z)$ is $n + r$. Then there is an $(n + r) \times (n + r)$ submatrix of $\mathbf{P}(z)$ that is generically invertible, and furthermore, this submatrix contains the matrix $\mathbf{A} - z\mathbf{I}$.*

Proof. Let $(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}, \bar{\mathbf{D}})$ and $\bar{z}$ be the numerically specified matrices and scalar, respectively, that causes the matrix pencil to achieve rank $n + r$, and denote the resulting

pencil by $\bar{\mathbf{P}}(\bar{z})$. Thus, there must be an $(n+r)\times(n+r)$ square and invertible submatrix in $\bar{\mathbf{P}}(\bar{z})$; denote this submatrix by $\bar{\mathbf{M}}(\bar{z})$. If $\bar{\mathbf{M}}(\bar{z})$ contains the matrix $\bar{\mathbf{A}} - \bar{z}\mathbf{I}$, we are done. If not, revert $\bar{z}$ to be a free variable z, and note that the determinant of $\bar{\mathbf{M}}(z)$ is a nonzero polynomial in z (it may be a constant). Let $\mathcal{S}$ denote the set of values of z that cause this polynomial to be zero (note that the cardinality of $\mathcal{S}$ is equal to the degree of the polynomial). Next, note that the first n columns and rows of $\bar{\mathbf{P}}(z)$ will be linearly independent for any z that is not an eigenvalue of $\bar{\mathbf{A}}$; let the set $\mathcal{T}$ denote the set of eigenvalues of $\bar{\mathbf{A}}$. Thus, for any $z \in \mathbb{C} \setminus \{\mathcal{S} \cup \mathcal{T}\}$, the first n rows and columns of $\bar{\mathbf{P}}(z)$ will be linearly independent, and simultaneously $\bar{\mathbf{M}}(z)$ will be invertible (indicating that $\bar{\mathbf{P}}(z)$ will have rank $n+r$). Thus, one can find a square submatrix of $\bar{\mathbf{P}}(z)$ that corresponds to the intersection of the first n rows and columns of that matrix, along with r other rows and columns. Since this square matrix has nonzero determinant for this particular numerical choice of free parameters, it will have nonzero determinant for almost any choice of free parameters (i.e., all but those that lie on an algebraic variety). □

We are now in place to prove Theorem C.2; for clarity, we will prove each of the directions of the theorem as a separate lemma.

Lemma C.2. *Let the graph of a structured linear system be given by $\mathcal{H}$. If the generic normal rank of the matrix pencil $\mathbf{P}(z)$ is equal to $n+r$, then there is a linking of size r from the input vertices to the output vertices in $\mathcal{H}$.*

Proof. By Lemma C.1, there is an $(n+r)\times(n+r)$ submatrix of $\mathbf{P}(z)$ that is generically invertible and contains $\mathbf{A}-z\mathbf{I}$; denote this submatrix by $\mathbf{M}(z)$. The determinant of $\mathbf{M}(z)$ is a nonzero polynomial, and by definition, each term in the determinant is a product of $n+r$ different elements, no two of which are in the same row or column of $\mathbf{M}(z)$. Consider any one of these terms, and consider only the elements in that term in the matrix $\mathbf{P}(z)$. Each of the first n rows will have exactly one of these elements, and each of the first n columns will have exactly one of these elements. In the graph $\mathcal{H}$, this corresponds to a single incoming and outgoing edge from each state vertex. Furthermore r columns in $\left[\begin{smallmatrix}\mathbf{B}\\ \mathbf{D}\end{smallmatrix}\right]$ will have exactly one of these elements each, and r rows in $\begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}$ will also have exactly one of these elements each. In $\mathcal{H}$, this corresponds to a single outgoing edge from each of r different input vertices, and a single incoming edge to each of r different output vertices. All together, the edges and vertices corresponding to these elements represent a set of r vertex-disjoint paths in $\mathcal{H}$ from r inputs to r outputs, together with a set of cycles composed of state vertices. Thus, $\mathcal{H}$ contains a linking of size r from the input vertices to the output vertices. □

Lemma C.3. *Let the graph of a structured linear system be given by $\mathcal{H}$. If there is a linking of size r from the input vertices to the output vertices in $\mathcal{H}$, then the generic normal rank of the matrix pencil is equal to $n+r$.*

Proof. We will start by considering the case where $\mathcal{H}$ is a path-graph, where the input vertex connects to a single state vertex, which leads into a path that covers all state vertices, and the last state vertex connects to the single output vertex. The system

matrices corresponding to this graph are of the form

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ * & 0 & 0 & \cdots & 0 & 0 \\ 0 & * & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & * & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} * \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & * \end{bmatrix}, \quad \mathbf{D} = 0,$$

where $*$ represents independent free variables. The matrix pencil for this system is

$$\begin{bmatrix} \mathbf{A} - z\mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \left[\begin{array}{cccccc|c} -z & 0 & 0 & \cdots & 0 & 0 & * \\ * & -z & 0 & \cdots & 0 & 0 & 0 \\ 0 & * & -z & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & * & -z & 0 \\ \hline 0 & 0 & 0 & \cdots & 0 & * & 0 \end{array}\right]. \tag{C.5}$$

If we set $z = 0$ and set all of the free parameters to be any nonzero values, the above matrix has full rank (equal to $n + 1$); thus, the determinant of the above matrix will be a nonzero polynomial in z and the free parameters. If the system has no states, but only a direct edge from the input to the output, we have $\mathbf{D} = *$ and all other system matrices are empty, and thus $\mathbf{P}(z) = \mathbf{D}$, which is again full rank (equal to 1) for any nonzero value of the free parameter

Now, we will prove the lemma. Let $P_1, P_2, \ldots, P_r$ be the paths in the r-linking from the inputs to the outputs in $\mathcal{H}$. For any parameter in the system matrices that does not correspond to an edge in this linking, set that parameter to 0. Now, after a simple reordering of the rows and columns of the matrix pencil (which does not change the rank of the matrix), we obtain

$$\operatorname{rank}(\mathbf{P}(z)) = \operatorname{rank} \begin{bmatrix} \mathbf{P}_1(z) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2(z) & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_3(z) & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_r(z) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -z\mathbf{I}_q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where q is equal to the number of state vertices not included in any of the paths of the linking. The zero columns and rows at the end of the matrix pencil represent those inputs and outputs that are not used in the linking. Each of the matrices $\mathbf{P}_i(z)$ is a matrix pencil of the form (C.5), corresponding to the path P_i, and thus has generically full normal rank. Thus the generic rank of the above matrix is equal to the sum of the ranks of each of the $\mathbf{P}_i(z)$'s plus q, which is equal to $n+r$. This concludes the proof. □

Lemmas C.2 and C.3 together prove Theorem C.2. Based on this characterization, we can obtain graph-theoretic conditions for various structural properties.

C.3.2 Structural Controllability

To obtain a graph theoretic condition for structural controllability, it is instructive to first consider an example.

Example C.7. Consider the structured system

$$\mathbf{x}[k+1] = \begin{bmatrix} 0 & 0 & 0 \\ \lambda_1 & \lambda_2 & 0 \\ \lambda_3 & 0 & \lambda_4 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} \lambda_5 \\ 0 \\ 0 \end{bmatrix} u[k]. \tag{C.6}$$

The controllability matrix for this system is given by

$$\mathcal{C}_{n-1} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} \end{bmatrix} = \begin{bmatrix} \lambda_5 & 0 & 0 \\ 0 & \lambda_1\lambda_5 & \lambda_1\lambda_2\lambda_5 \\ 0 & \lambda_3\lambda_5 & \lambda_3\lambda_4\lambda_5 \end{bmatrix}.$$

The determinant of this matrix is given by

$$f(\lambda_1, \lambda_2, \cdots, \lambda_5) = \lambda_1\lambda_3\lambda_5^3(\lambda_4 - \lambda_2).$$

This determinant is zero if and only if any of λ_1, λ_3 or λ_5 are zero, or if $\lambda_4 = \lambda_2$. Let us examine the graphical interpretation of these conditions. The graph $\mathcal{H}$ associated with the above system is shown in Fig. C.5.

Figure C.5: Graph $\mathcal{H}$ of system (C.6).

Setting any of λ_1, λ_3 or λ_5 to zero corresponds to removing the corresponding edge from the graph. Note that this would remove all paths from the input vertex to some (or all) of the vertices. This is an intuitive result: if some state vertex cannot be reached by a path from an input, there is no way that the system can be controllable (since the input will not affect that state in any way).

One might imagine that this reachability condition is also sufficient for controllability. However, consider the case where $\lambda_2 = \lambda_4 = 0$ (i.e., the graph $\mathcal{H}$ does not have self-loops on state vertices x_2 and x_3). Even though this graph still has a path from the input vertex to every state vertex, we saw above that the determinant of the controllability matrix will be zero, causing the system to be uncontrollable. The intuition behind this is that when x_2 and x_3 do not have self-loops, they have no 'dynamics', meaning that

they do not maintain their previous state, and instead only take the value provided by x_1, scaled by λ_1 or λ_3, respectively. Thus x_2 and x_3 cannot be driven independently to different points in the state-space. Similarly, even if we put self-loops on the two vertices but set them to have the same numerical value (i.e., $\lambda_2 = \lambda_4$), the dynamics of the two states are exactly the same, and thus they cannot be driven independently. However, as long as λ_2 and λ_4 are different (which will be the case generically), and none of λ_1, λ_3 or λ_5 are zero (which, again, will be the case generically) the system will be controllable. □

The above example illustrates that simply having paths from the input vertices to the state vertices is not enough for controllability; one needs to ensure that there are sufficient dynamics in the system to allow the states to be driven independently. The theorem characterizes the conditions on $\mathcal{H}$ for the system to be structurally controllable. The terminology *$\mathcal{U}$-rooted path* is used to denote a path that starts at an input vertex (i.e., a vertex in $\mathcal{U}$).

Theorem C.3 ([18])**.** *The structured pair $(\mathbf{A}, \mathbf{B})$ is structurally controllable if and only if the graph $\mathcal{H}$ satisfies both of the following properties:*

1. *Every state vertex $x_i \in \mathcal{X}$ can be reached by a path from some input vertex.*
2. *$\mathcal{H}$ contains a subgraph that is a disjoint union of cycles and $\mathcal{U}$-rooted paths, covering all of the state vertices.*

To prove the above theorem, we will make use of the PBH test for controllability (see Section 2.9.1). We will prove the theorem with the aid of two lemmas.[1]

Lemma C.4. *Let $\mathcal{H}$ denote the graph associated with the structured pair $(\mathbf{A}, \mathbf{B})$. If every state vertex can be reached by a path from some input vertex, then for almost any choice of free parameters in the system matrices, rank $\begin{bmatrix}\mathbf{A} - z\mathbf{I} & \mathbf{B}\end{bmatrix} = n$ for every $z \in \mathbb{C} \setminus \{0\}$. Furthermore, if there is at least one state vertex that cannot be reached by a path from some input, then for any choice of free parameters in the system matrices, there is at least one $z \in \mathbb{C}$ such that rank $\begin{bmatrix}\mathbf{A} - z\mathbf{I} & \mathbf{B}\end{bmatrix} \leq n - 1$.*

Proof. First, denote $f(z) = \det(\mathbf{A} - z\mathbf{I})$, and note that it is a nonzero polynomial in the free parameters in $\mathbf{A}$ and z, but it does not include any of the free parameters from $\mathbf{B}$. Next, consider the matrix $\bar{\mathbf{P}}_1(z)$ formed by removing the first column of $\begin{bmatrix}\mathbf{A} - z\mathbf{I} & \mathbf{B}\end{bmatrix}$. In the graph $\mathcal{H}$, this corresponds to removing all outgoing edges from the first state

[1] The following proofs are different from the standard proofs of the above theorem, and are inspired by the approaches used in [91, 90].

vertex, but leaving all of the incoming edges. Thus, this state vertex can effectively be viewed as an 'output' vertex in the modified graph, leaving $n-1$ state vertices. From Theorem C.2, we see that the matrix pencil $\bar{\mathbf{P}}_1(z)$ will have generic normal rank equal to $(n-1)+1=n$ if there is a linking of size 1 from the input vertices to the 'output' (state) vertex. This condition is satisfied since each state vertex can be reached by a path from some input, and thus there is an $n \times n$ submatrix of $\bar{\mathbf{P}}_1(z)$ that has a nonzero determinant $f_1(z)$; this determinant is a polynomial in z and some of the free parameters in $\mathbf{A}$ and $\mathbf{B}$, with the exception of all free parameters in the first row of $\mathbf{A}$ (since that column was removed).

Continuing in this way, we obtain $n+1$ nonzero polynomials $f(z), f_1(z), f_2(z), \ldots, f_n(z)$, where $f_i(z)$ is a polynomial that does not depend on any free parameters from the i-th column of $\mathbf{A}$, and $f(z)$ does not contain any free parameters from $\mathbf{B}$. Thus, each free parameter in the system matrices does not appear in at least one of these polynomials.

Now, the matrix $\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\end{bmatrix}$ will have rank less than n only if all of these polynomials are zero simultaneously. In other words, the value of z that causes this matrix to lose rank must be a common zero of all $n+1$ polynomials. However, since each free parameter does not appear in at least one polynomial, any common root of all polynomials cannot be a function of any of the free parameters. Thus, $z=0$ is the only possible common root of all polynomials (generically). This implies that, generically, rank $\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\end{bmatrix}$ for all $z \in \mathbb{C} \setminus \{0\}$.

On the other hand, suppose that some state vertices cannot be reached by a path from any inputs; let $\mathcal{X}_1$ denote the set of all state vertices that can be reached by a path from some input, and let $\mathcal{X}_2 = \mathcal{X} \setminus \mathcal{X}_1$ denote all state vertices that cannot be reached by a path from any input. Then, renumbering the vertices so that those in $\mathcal{X}_1$ come first, the $\mathbf{A}$ and $\mathbf{B}$ matrices have the form

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}\mathbf{B}_1 \\ \mathbf{0}\end{bmatrix}.$$

The zero matrix in $\mathbf{A}$ is due to the fact that no state vertex in $\mathcal{X}_2$ has an edge from a state vertex in $\mathcal{X}_1$ (otherwise there would be a path from an input vertex to that vertex in $\mathcal{X}_2$). Similarly, the zero matrix in $\mathbf{B}$ exists because no input vertex directly connects to any vertex in $\mathcal{X}_2$. The matrices $\mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{22}$ and $\mathbf{B}_1$ represent the edges that exist between the state and input vertices. Thus, we have

$$\text{rank}\begin{bmatrix}\mathbf{A}-z\mathbf{I} & \mathbf{B}\end{bmatrix} = \text{rank}\begin{bmatrix}\mathbf{A}_{11}-z\mathbf{I} & \mathbf{A}_{12} & \mathbf{B}_1 \\ \mathbf{0} & \mathbf{A}_{22}-z\mathbf{I} & \mathbf{0}\end{bmatrix}.$$

For any choice of free parameters in $\mathbf{A}$ and $\mathbf{B}$, the matrix $\mathbf{A}_{22}-z\mathbf{I}$ will lose rank for any z that is an eigenvalue of $\mathbf{A}_{22}$, and thus for those values of z, the above matrix pencil will have rank at most $n-1$. □

Lemma C.5. *If $\mathcal{H}$ contains a subgraph that is a disjoint union of cycles and $\mathcal{U}$-rooted paths, covering all the state vertices, then for almost any choice of free parameters in $\mathbf{A}$ and $\mathbf{B}$, rank $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} = n$. If $\mathcal{H}$ does not contain such a subgraph, then for any choice of free parameters, rank $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} < n$*

Proof. Suppose that $\mathcal{H}$ contains a subgraph that is a disjoint union of a set of cycles and $\mathcal{U}$-rooted paths, covering all state vertices. Set the values of all parameters corresponding to edges not in this subgraph to be zero. Then, the resulting matrix $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}$ has zero elements everywhere except for n locations, each of which is in a different row and column. Thus, the matrix will have rank n for any nonzero choice of these remaining parameters. More generally, the determinant of the n corresponding columns of $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}$ will be a nonzero polynomial in all free parameters in those columns, and thus the matrix will have rank n for almost any choice of free parameters.

To prove the second statement, we will prove the contrapositive. Suppose rank $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} = n$ for some numerical choice of free parameters. Then, there is an $n \times n$ submatrix that has nonzero determinant, and this determinant is a polynomial in the free parameters. Each term in the determinant consists of n free parameters, each from a different row and column of $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}$. The set of edges corresponding to these parameters represents a single incoming edge to each state vertex, and a single outgoing edge from the vertices that correspond to the columns containing those parameters. Thus, each state vertex has exactly a single incoming edge, and at most a single outgoing edge, and each input vertex has at most a single outgoing edge. This set of edges corresponds exactly to a disjoint union of a set of cycles and $\mathcal{U}$-rooted paths that cover all state vertices. □

The above two lemmas together prove Theorem C.3.

C.3.3 Structural Observability

The test for structural observability is completely analogous to the proof for structural controllability, and the appropriate conditions are given by the following theorem. The terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem C.4 ([18])**.** *The structured pair $(\mathbf{A}, \mathbf{C})$ is structurally observable if and only if the graph $\mathcal{H}$ satisfies both of the following properties:*

1. *Every state vertex $x_i \in \mathcal{X}$ has a path to some output vertex.*
2. *$\mathcal{H}$ contains a subgraph that covers all state vertices, and that is a disjoint union of cycles and $\mathcal{Y}$-topped paths.*

The proof is a straightforward translation of the proof of Theorem C.3.

C.3.4 Structural Invertibility

Recall from Theorem 2.7 that a system is invertible if and only if the matrix pencil for the system has full column rank for at least one choice of $z \in \mathbb{C}$. Theorem C.2 provides

the conditions under which this will hold, leading to the following result.

Theorem C.5 ([18, 91])**.** *Let the graph of a structured linear system be given by $\mathcal{H}$. Then the system is structurally invertible if and only if there is a linking of size m from the input vertices to the output vertices, where m is the number of inputs.*

The above result says that if the graph of the structured system has m vertex disjoint paths from the inputs to the outputs, then for almost any (real-valued) choice of free parameters in $\mathbf{A}$, $\mathbf{B}, \mathbf{C}$ and $\mathbf{D}$, the transfer function matrix $\mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$ will have rank m. Based on Theorem 2.4, this means that the first m columns of the matrix $\mathcal{M}_{N-\text{nullity}(\mathbf{D})+1}$ will be linearly independent of all other columns in $\mathcal{M}_{N-\text{nullity}(\mathbf{D})+1}$.

Example C.8. Consider the structured system

$$\begin{aligned} \mathbf{x}[k+1] &= \begin{bmatrix} \lambda_1 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \\ 0 & 0 & \lambda_4 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 0 & 0 \\ 0 & \lambda_5 \\ \lambda_6 & 0 \end{bmatrix} \mathbf{u}[k] \\ \mathbf{y}[k] &= \begin{bmatrix} \lambda_7 & 0 & 0 \\ 0 & 0 & \lambda_8 \end{bmatrix} \mathbf{x}[k]. \end{aligned} \tag{C.7}$$

The graph $\mathcal{H}$ associated with this system is given by

This graph has a linking of size $m = 2$ from the inputs to the outputs, and thus this system is structurally invertible.

C.3.5 Structural Invariant Zeros

It turns out that the number of invariant zeros of a linear system is also a structured property (i.e., a linear system with a given zero/nonzero structure will have the same number of invariant zeros for almost any choice of the free parameters) [90]. The following theorem from [18] characterizes the generic number of invariant zeros of a structured system in terms of the associated graph $\mathcal{H}$. Once again, the terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem C.6 ([90, 18])**.** *Let* $\mathbf{P}(z)$ *be the matrix pencil of the structured set* $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$*, and suppose the generic normal rank of* $\mathbf{P}(z)$ *is* $n+m$ *even after the deletion of an arbitrary row from* $\mathbf{P}(z)$*. Then the generic number of invariant zeros of system* (C.1) *is equal to* n *minus the maximal number of vertices in* $\mathcal{X}$ *contained in the disjoint union of a size* m *linking from* $\mathcal{U}$ *to* $\mathcal{Y}$*, a set of cycles in* $\mathcal{X}$*, and a set of* $\mathcal{Y}$*-topped paths.*

From Theorem 2.5, note that one can characterize strong observability of a system by examining the number of invariant zeros of that system. The proof of the above theorem follows the same lines as the proof of Theorem C.3. First, one shows that the matrix pencil will generically have rank $n + m$ for all $z \in \mathbb{C} \setminus \{0\}$ by constructing a set of nonzero polynomials, each of which does not involve at least one free parameter from the system matrices. This implies that any common root of these polynomials must be $z = 0$. Then, one shows that the matrix pencil at $z = 0$ will have rank $n+m$ if the graph contains a disjoint set of cycles, m-linking and $\mathcal{Y}$-topped paths. A connection between the connectivity of the graph and strong observability is developed in Section 7.3.3.

Bibliography

[1] P. Antsaklis and A. N. Michel. *Linear Systems*. Birkhauser Boston, 2005.

[2] S. D. Baker and D. H. Hoglund. Medical-grade, mission-critical wireless networks [Designing an enterprise mobility solution in the healthcare environment]. *IEEE Engineering in Medicine and Biology Magazine*, 27(2):86–95, Mar. 2008.

[3] P. Barooah and J. Hespanha. Estimation on graphs from relative measurements. *IEEE Control Systems Magazine*, 27(4):57–74, Aug. 2007.

[4] D. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[5] R. E. Blahut. *Algebraic Codes for Data Transmission*. Cambridge, UK: Cambridge University Press, 2003.

[6] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2nd edition, 2006.

[7] National Transportation Safety Board. Pipeline rupture and subsequent fire in Bellingham, Washington. Technical report, 1999.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[9] A. Cardenas, S. Amin, and S. S. Sastry. Research challenges for security of control systems. In *Proceedings of the 3rd USENIX Workshop on Hot Topics in Security*, 2008.

[10] C. G. Cassandras and W. Li. Sensor networks and cooperative control. *European Journal of Control*, 11(4–5):436–463, 2005.

[11] R. N. Charette. This car runs on code. *IEEE Spectrum*, Feb. 2009.

[12] E. Y. Chow and A. S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):603–614, 1984.

[13] M. Darouach, M. Zasadzinski, and S. J. Xu. Full-order observers for linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 39(3):606–609, Mar. 1994.

[14] S. Deb, M. Médard, and C. Choute. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, June 2006.

[15] M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, Mar. 1974.

[16] M. Dietzfelbinger. Gossiping and broadcasting versus computing functions in networks. *Discrete Applied Mathematics*, 137(2):127–153, Mar. 2004.

[17] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *Proceedings of the 5th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 69–76, 2006.

[18] J.-M. Dion, C. Commault, and J. van der Woude. Generic properties and control of linear structured systems: a survey. *Automatica*, 39(7):1125–1144, July 2003.

[19] M. Effros, R. Koetter, and M. Médard. Breaking network logjams. *Scientific American*, pages 78–85, June 2007.

[20] N. Elia. Remote stabilization over fading channels. *Systems & Control Letters*, 54:237–249, 2005.

[21] P. M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. *Automatica*, 26(3):459–474, 1990.

[22] S. H. Friedberg, A. J. Insel, and L. E. Spence. *Linear Algebra*. Upper Saddle River, NJ: Prentice Hall, 4th edition, 2003.

[23] A. Gibbons. *Algorithmic Graph Theory*. Cambridge, UK: Cambridge University Press, 1985.

[24] A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):755–764, Apr. 2005.

[25] A. Giridhar and P. R. Kumar. Toward a theory of in-network computation in wireless sensor networks. *IEEE Communications Magazine*, 44(4):98–107, Apr. 2006.

[26] W. S. Gray, O. R. Gonzalez, and M. Dogan. Stability analysis of digital linear flight controllers subject to electromagnetic disturbances. *IEEE Transactions on Aerospace and Electronic Systems*, 36(4):1204–1218, Oct. 2000.

[27] G. Gross. Expert: Hackers penetrating industrial control systems. *Computerworld*, Nov. 2009.

[28] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *IEEE Transactions on Automatic Control*, 54(8):1807–1819, Aug. 2009.

[29] C. N. Hadjicostis. *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*. Boston, MA: Kluwer Academic Publishers, 2002.

[30] C. N. Hadjicostis. Non-concurrent error detection and correction in fault-tolerant discrete-time LTI dynamic systems. *IEEE Transactions on Circuits and Systems*, 50(1):45–55, Jan. 2003.

[31] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 1205–1210, 2002.

[32] C. N. Hadjicostis and G. Verghese. Structured redundancy for fault tolerance in LTI state-space models and Petri nets. *Kybernetika*, 35:39–55, Jan. 1999.

[33] T. Hamilton. Smart grid saves power, but can it thwart hackers? *The Toronto Star*, Aug. 2009.

[34] G. H. Hardy. *A Mathematician's Apology*. Cambridge University Press, 1940.

[35] M. L. J. Hautus. Strong detectability and observers. *Linear Algebra and its Applications*, 50:353–368, Apr. 1983.

[36] J. P. Hespanha. *Linear Systems Theory*. Princeton University Press, 2009.

[37] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, Jan. 2007.

[38] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger. Byzantine modification detection in multicast networks using random network coding. *IEEE Transactions on Information Theory*, 54(6):2798–2803, 2008.

[39] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[40] S. Hosoe. Determination of generic dimensions of controllable subspaces and its application. *IEEE Transactions on Automatic Control*, 25(6):1192–1196, Dec. 1980.

[41] M. Hou and P. C. Muller. Disturbance decoupled observer design: A unified viewpoint. *IEEE Transactions on Automatic Control*, 39(6):1338–1341, June 1994.

[42] M. Hou and R. J. Patton. Input observability and input reconstruction. *Automatica*, 34(6):789–794, June 1998.

[43] J. M. House and G. E. Kelly. An overview of building diagnostics. In *National Conference on Building Commissioning*, pages 1–9, 2000.

[44] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks*. Berlin, Germany: Springer-Verlag, 2005.

[45] J. Hyarinen and Satu Karki. *Building Optimization and Fault Diagnosis Source Book*. International Energy Agency, 1996.

[46] O. C. Imer, S. Yuksel, and T. Basar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 42(9):1429–1439, Sep. 2006.

[47] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[48] J. Jin, M.-J. Tahk, and C. Park. Time-delayed state and unknown input observation. *International Journal of Control*, 66(5):733–745, Mar. 1997.

[49] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, Oct. 2003.

[50] K. Kowalenko. The cyberhacker's next victim: Industrial infrastructures. *The Institute*, Apr. 2010.

[51] W. Kratz. Characterization of strong observability and construction of an observer. *Linear Algebra and its Applications*, 221:31–40, May 1995.

[52] B. Krebs. 'Smart Grid' raises security concerns. *The Washington Post*, July 2009.

[53] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge, UK: Cambridge University Press, 2006.

[54] K. Lee, E. Feron, and A. Pritchett. Air traffic complexity: An input-output approach. In *Proceedings of the American Control Conference*, pages 474–479, 2007.

[55] C.-T. Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19(3):201–208, June 1974.

[56] C.-Y. Liu. A study of flight-critical computer system recovery from space radiation-induced error. *IEEE Aerospace and Electronic Systems Magazine*, 17(7):19–25, July 2002.

[57] N. A. Lynch. *Distributed Algorithms*. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1996.

[58] V. Madani and D. Novosel. Taming the power grid. *IEEE Spectrum*, Jan. 2005.

[59] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–122, 2006.

[60] P. J. Moylan. Stable inversion of linear systems. *IEEE Transactions on Automatic Control*, 22(1):74–78, Feb. 1977.

[61] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[62] F. Pasqualetti, A. Bicchi, and F. Bullo. Consensus computation in unreliable networks: A system theoretic approach. *IEEE Transactions on Automatic Control*. To appear.

[63] R. Patton, P. Frank, and R. Clark, editors. *Fault Diagnosis in Dynamic Systems: Theory and Application*. Prentice Hall International, 1989.

[64] Michael Rabbat and Robert D. Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 20–27, 2004.

[65] D. Rappaport and L. M. Silverman. Structure and stability of discrete-time optimal systems. *IEEE Transactions on Automatic Control*, 16(3):227–233, June 1971.

[66] K. J. Reinschke. *Multivariable Control A Graph-Theoretic Approach*. Berlin, Germany: Springer-Verlag, 1988.

[67] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the American Control Conference*, pages 1859–1864, 2005.

[68] C. L. Robinson and P. R. Kumar. Sending the most recent observation is not optimal in networked control. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 334–339, 2007.

[69] C. L. Robinson and P. R. Kumar. Optimizing controller location in networked control systems with packet drops. *IEEE Journal on Selected Areas in Communications*, 26(4):661–671, May 2008.

[70] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick. An overview of emerging results in cooperative UAV control. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 602–607, 2004.

[71] M. Saif and Y Guan. A new approach to robust fault detection and identification. *IEEE Transactions on Aerospace and Electronic Systems*, 29(3):685–695, July 1993.

[72] M. K. Sain and J. L. Massey. Invertibility of linear time-invariant dynamical systems. *IEEE Transactions on Automatic Control*, AC-14(2):141–149, Apr. 1969.

[73] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, Jan. 2007.

[74] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proceedings of the 2001 American Control Conference*, pages 1491–1496, 2001.

[75] L. M. Silverman. Inversion of multivariable linear systems. *IEEE Transactions on Automatic Control*, AC-14(3):270–276, June 1969.

[76] L. M. Silverman. Discrete Riccati equations: Alternative algorithms, asymptotic properties and system theory interpretations. In C. T. Leondes, editor, *Control and Dynamic Systems*, volume 12, pages 313–386. New York, NY: Academic Press, 1976.

[77] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, Sep. 2004.

[78] J. R. Sklaroff. Redundancy management technique for space shuttle computers. *IBM Journal of Research and Development*, 20(1):20 –28, Jan. 1976.

[79] S. Skogestad. *Chemical and Energy Process Engineering*. CRC Press, 2009.

[80] T. Smith. Hacker jailed for revenge sewage attacks. *The Register*, Oct. 2001.

[81] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems security. Technical report, National Institute of Standards and Technology, 2008.

[82] S. Sundaram and C. N. Hadjicostis. Delayed observers for linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 52(2):334–339, Feb. 2007.

[83] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents – part I: Attacking the network. In *Proceedings of the American Control Conference*, 2008.

[84] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents – part II: Overcoming malicious behavior. In *Proceedings of the American Control Conference*, 2008.

[85] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Transactions on Automatic Control*, 56(7):1495–1508, July 2011.

[86] G. Takos and C. N. Hadjicostis. Error correction in DFT codes subject to low-level quantization noise. *IEEE Transactions on Signal Processing*, 56(3):1043–1054, Mar. 2008.

[87] D. Teneketzis and N. R. Sandell, Jr. Linear regulator design for stochastic systems by a multiple time-scales method. *IEEE Transactions on Automatic Control*, AC-22(4):615–621, Aug. 1977.

[88] F. Turner, editor. *High Performance Buildings*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 2010.

[89] M. E. Valcher. State observers for discrete-time linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 44(2):397–401, Feb. 1999.

[90] J. van der Woude. The generic number of invariant zeros of a structured linear system. *SIAM Journal on Control and Optimization*, 38(1):1–21, Nov. 1999.

[91] J. W. van der Woude. A graph-theoretic characterization for the rank of the transfer matrix of a structured system. *Mathematics of Control, Signals and Systems*, 4(1):33–40, Mar. 1991.

[92] J. von Neumann. *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components*. Princeton, NJ: Princeton Univ. Press, 1956.

[93] D. B. West. *Introduction to Graph Theory*. Upper Saddle River, NJ: Prentice-Hall Inc., 2001.

[94] A. S. Willsky. On the invertibility of linear systems. *IEEE Transactions on Automatic Control*, 19(2):272–274, June 1974.

[95] W. M. Wonham. On pole assignment in multi-input controllable linear systems. *IEEE Transactions on Automatic Control*, AC-12(6):660–665, Dec. 1967.

[96] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, Sep. 2004.

[97] Y. C. Yeh. Triple-triple redundant 777 primary flight computer. In *Proceedings of the IEEE Aerospace Applications Conference*, pages 293–307, 1996.