30.114 Advanced Feedback and Control

SUTD, Spring 2019

Reuben R.W. Wang

Instructor: *Professor Foong Shao Hui*.
Instructor office hours: *Wednesday and Friday*.
Instructor email: `foongshaohui@sutd.edu.sg`.

Contents

Chapter 1

Introduction and Recap

§1.1 Introduction

The analysis and understanding of multi-state dynamic systems with possibly unknown or unobserveable states with the goal of stabilizing and/or improving performance using modern digital controllers and components. In mechanical control systems, we generally will be looking at analog systems, but this is not always true for digital systems. An overview of the course progression is given in figure 1.1 below.

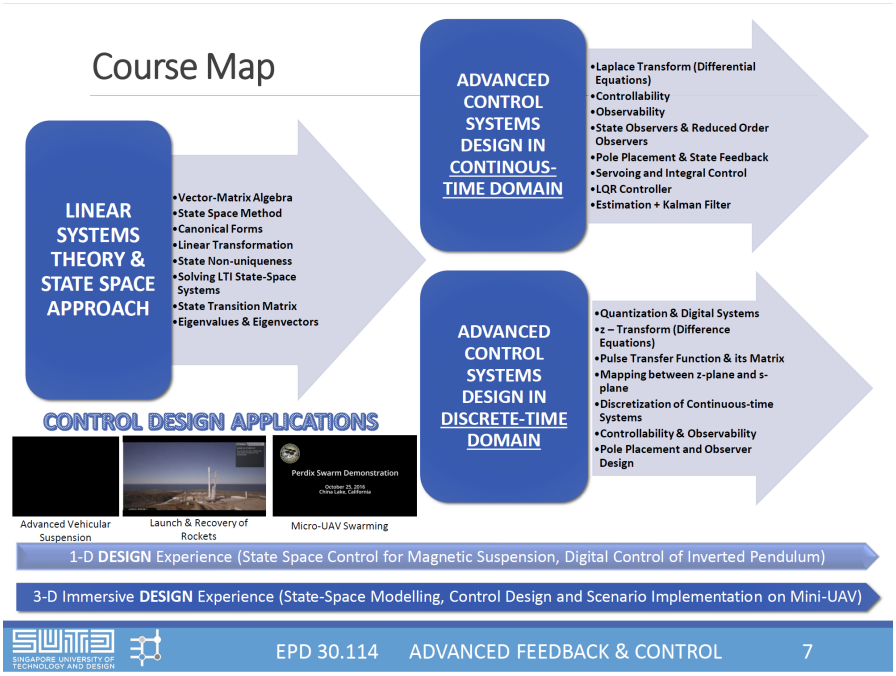

Figure 1.1: Course Map

As seen above, we will be building upon the knowledge from the previous systems and control class in term 5 via the following key areas.

1. Not all states of the system can be observed,
2. We will be looking at multi-input multi-output (MIMO) systems.
3. We will be studying optimal controllers (LQR).
4. We will be looking at digital and discrete time systems (as opposed to analog/continuous time systems)
5. We will also be looking at difference equations (as opposed to differential equations).

Some of the more specific concepts we will be learning to extend our knowledge of control theory and applications include periodic signals and discrete-time systems. Mathematical modelling and analysis of discrete time systems in various disciplines using state space, pulse transfer function and and z-transform. Relating controllability and observability and their canonical forms to synthesize and design advanced continuous and discrete-time controllers. An introduction of pole-placement based controller design and formulation of state observers.

This class will be focusing on linear systems since this is still an undergraduate course. The programming languages/software we will utilize for this course will be MATLAB, LabVIEW and C. Before we delve into the new material, let's do a quick recap of the materials from traditional control theory in *30.104 Systems and Control*.

§1.2 Complex Variables

Any complex variable can be represented as $s = \sigma + j\omega$. Complex functions can be expressed as $G(s) = G_x + jG_y$ where G_x and G_y are real functions. For control systems, these will generally take the form:

$$G(s) = \frac{N(s)}{D(s)} \tag{1.1}$$

where $N(s)$ and $D(s)$ are polynomial functions in s. As such, $N(s)$ and $D(s)$ tell us about the *poles* and *zeros* of the system via polynomial factorization over a complex field by the fundamental theorem of algebra. The number of poles and zeros depends on the *degree* of the respective polynomial functions.

§1.3 Differential Equations

For differential equations, we have *linear* and *non-linear* differential equations. These depend on the form of the differential equation. For instance, consider the following 2 differential equations:

$$\ddot{x} + \cos(2t)\dot{x} + 10x = 0, \quad (x^2 - 1)\ddot{x} + \dot{x} + 3x = 0 \tag{1.2}$$

The first (upper) is linear in x whereas the second (lower) is non-linear. The general form of a linear differential equation can be written as:

$$\sum_n f_n(t)\frac{d^n x(t)}{dt^n} = 0 \tag{1.3}$$

If the coefficients of the time-derivative terms ($\frac{d^n x(t)}{dt^n}$) are time-**dependent**, we call these *time-varying*. The differential equation is *time-invariant* otherwise. We largely studied **linear and time-invariant (LTI)** systems in 30.104.

§1.4 Laplace Transform

The Laplace transform is defined as:

$$\boxed{\mathcal{L}\left[f(t)\right] = F(s) = \int_0^\infty f(t)e^{-st}dt} \tag{1.4}$$

which satisfy the properties of *linearity*, *frequency shifts*, *time shifts*, *scaling*, *differentiation* and *integration*. 2 useful theorems that we have used to apply in the S-domain is the *initial value theorem* and *final value theorem*.

Theorem 1.4.1. Initial Value Theorem: *For the one-sided Laplace transformation of $f(t)$ for which the time-domain function is bounded on $t \in (0, \infty)$, we have that*

$$\lim_{t\to 0} f(t) = \lim_{s\to\infty} sF(s) \tag{1.5}$$

if $\lim_{t\to 0^+} f(t)$ exists and is well defined.

Theorem 1.4.2. Final Value Theorem: *For the one-sided Laplace transformation of $f(t)$ for which the time-domain function is bounded on $t \in (0, \infty)$, we have that*

$$\lim_{t\to infty} f(t) = \lim_{s\to 0} sF(s) \tag{1.6}$$

if $\lim_{t\to\infty} f(t)$ exists tends to a finite value.

We can also utilize The inverse Laplace transform to go from the s-domain to the time-domain $F(s) \to f(t)$. The inverse Laplace transform can simply be read off the table of Laplace transforms (utilizing the LT properties). The table is given in figure 1.2

§1.5 System Modelling

§1.5.1 Mechanical Systems

For mechanical systems, we utilized Newton's second law to derive the equations of motion. In general, we saw that we could categorize our modelling into translational and rotational systems, which with their associated "*versions*" of Newton's second law given below:

$$\sum F = m\ddot{x}, \quad \sum \tau = J\ddot{\theta} \tag{1.7}$$

Table of Laplace Transforms

	$f(t)=\mathcal{L}^{-1}\{F(s)\}$	$F(s)=\mathcal{L}\{f(t)\}$		$f(t)=\mathcal{L}^{-1}\{F(s)\}$	$F(s)=\mathcal{L}\{f(t)\}$
1.	1	$\frac{1}{s}$	2.	$\mathbf{e}^{at}$	$\frac{1}{s-a}$
3.	$t^n,\quad n=1,2,3,\ldots$	$\frac{n!}{s^{n+1}}$	4.	$t^p, p>-1$	$\frac{\Gamma(p+1)}{s^{p+1}}$
5.	$\sqrt{t}$	$\frac{\sqrt{\pi}}{2s^{\frac{3}{2}}}$	6.	$t^{n-\frac{1}{2}},\quad n=1,2,3,\ldots$	$\frac{1\cdot3\cdot5\cdots(2n-1)\sqrt{\pi}}{2^n s^{n+\frac{1}{2}}}$
7.	$\sin(at)$	$\frac{a}{s^2+a^2}$	8.	$\cos(at)$	$\frac{s}{s^2+a^2}$
9.	$t\sin(at)$	$\frac{2as}{\left(s^2+a^2\right)^2}$	10.	$t\cos(at)$	$\frac{s^2-a^2}{\left(s^2+a^2\right)^2}$
11.	$\sin(at)-at\cos(at)$	$\frac{2a^3}{\left(s^2+a^2\right)^2}$	12.	$\sin(at)+at\cos(at)$	$\frac{2as^2}{\left(s^2+a^2\right)^2}$
13.	$\cos(at)-at\sin(at)$	$\frac{s\left(s^2-a^2\right)}{\left(s^2+a^2\right)^2}$	14.	$\cos(at)+at\sin(at)$	$\frac{s\left(s^2+3a^2\right)}{\left(s^2+a^2\right)^2}$
15.	$\sin(at+b)$	$\frac{s\sin(b)+a\cos(b)}{s^2+a^2}$	16.	$\cos(at+b)$	$\frac{s\cos(b)-a\sin(b)}{s^2+a^2}$
17.	$\sinh(at)$	$\frac{a}{s^2-a^2}$	18.	$\cosh(at)$	$\frac{s}{s^2-a^2}$
19.	$\mathbf{e}^{at}\sin(bt)$	$\frac{b}{(s-a)^2+b^2}$	20.	$\mathbf{e}^{at}\cos(bt)$	$\frac{s-a}{(s-a)^2+b^2}$
21.	$\mathbf{e}^{at}\sinh(bt)$	$\frac{b}{(s-a)^2-b^2}$	22.	$\mathbf{e}^{at}\cosh(bt)$	$\frac{s-a}{(s-a)^2-b^2}$
23.	$t^n\mathbf{e}^{at},\quad n=1,2,3,\ldots$	$\frac{n!}{(s-a)^{n+1}}$	24.	$f(ct)$	$\frac{1}{c}F\left(\frac{s}{c}\right)$
25.	$u_c(t)=u(t-c)$ Heaviside Function	$\frac{\mathbf{e}^{-cs}}{s}$	26.	$\delta(t-c)$ Dirac Delta Function	$\mathbf{e}^{-cs}$
27.	$u_c(t)f(t-c)$	$\mathbf{e}^{-cs}F(s)$	28.	$u_c(t)g(t)$	$\mathbf{e}^{-cs}\mathcal{L}\{g(t+c)\}$
29.	$\mathbf{e}^{ct}f(t)$	$F(s-c)$	30.	$t^n f(t),\quad n=1,2,3,\ldots$	$(-1)^n F^{(n)}(s)$
31.	$\frac{1}{t}f(t)$	$\int_s^\infty F(u)du$	32.	$\int_0^t f(v)dv$	$\frac{F(s)}{s}$
33.	$\int_0^t f(t-\tau)g(\tau)d\tau$	$F(s)G(s)$	34.	$f(t+T)=f(t)$	$\frac{\int_0^T \mathbf{e}^{-st}f(t)dt}{1-\mathbf{e}^{-sT}}$
35.	$f'(t)$	$sF(s)-f(0)$	36.	$f''(t)$	$s^2F(s)-sf(0)-f'(0)$
37.	$f^{(n)}(t)$	$s^nF(s)-s^{n-1}f(0)-s^{n-2}f'(0)\cdots-sf^{(n-2)}(0)-f^{(n-1)}(0)$			

Figure 1.2: Table of Laplace Transforms

§1.5.2 Electrical Systems

For electrical systems, we used Kirchhoff's (nodal rule) and Ohm's laws. These are respectively given below:

$$\sum_j I_j = 0, \quad V = IR \tag{1.8}$$

We also saw how to model some basic linear electrical components such as the capacitor and inductor. These have following charge and current dependence relations.

$$\frac{dQ}{dt} = -\frac{1}{RC}Q, \tag{1.9}$$

$$\frac{dI}{dt} = -\frac{R}{L}I \tag{1.10}$$

with the convention that $I = -\frac{dQ}{dt}$.

§1.5.3 Fluid and Thermal Systems

Fluid (incompressible) and thermal systems. Continuity of flow and Newton's second law. To formulate differential equations for such systems, we used the quantities fluid *resistance* (R), *capacitance* (C) and *inertance* (I). These are defined respectively as follows:

$$P_a - P_b = QR, \tag{1.11}$$

$$Q = C\frac{d}{dt}(P_a - P_b), \tag{1.12}$$

$$P_a - P_b = I\frac{dQ}{dt} \tag{1.13}$$

where Q is the control volume of fluid. Similar, we have analogous quantities in thermal systems known as thermal *resistance* (R) and thermal *capacitance* (C). The respective equations are:

$$\theta_a - \theta_b = qR, \tag{1.14}$$

$$q = C\frac{d}{dt}(\theta_a - \theta_b) \tag{1.15}$$

§1.6 Linear Time-Invariant (LTI) Systems

From these models, we can find the *transfer functions* of the system via the Laplace transform. These aid in the analysis and control implementations for LTI systems.

Definition 1.6.1. Transfer Function: *The ratio of the Laplace transform of the output to the Laplace transform of the input under the assumption that all initial conditions are zero.*

$$TF(s) = \frac{N(s)}{D(s)} \tag{1.16}$$

where $N(s)$ is the output and $D(s)$ is the input.

§1.6.1 System Order and Type

Systems can be characterized by their order, which corresponds to the order of the differential equation describing the input system. It is also the degree of the polynomial of the denominator of the transfer function. In all such systems, we generally study the *transient* (time-evolution from initial to final state) and *steady-state* (response as $t \to \infty$) response.

Another means of characterizing a system/plant is via its *type*. The *type* of a system is defined as the number of poles at the origin. To be clear, given that we can always write the transfer function of an LTI system as:

$$G(s) \sim \frac{1}{s^N \cdot \Pi_j (T_j s + 1)} \tag{1.17}$$

We have that the system type is seen to be N.

§1.6.2 1^{st} Order LTI Systems

The simplest would be a system of lowest non-trivial order (i.e first order system). The *plant function* (Laplace transform of the system ODE) for a first order system can be written as:

$$G(s) = \frac{1}{Ts + 1} \tag{1.18}$$

where T is the time constant of the system, at which the system will attain $\sim 63.2\%$ of its steady-state value.

§1.6.3 2^{nd} Order LTI Systems

For second order LTI systems, it is often useful to write the plant function in terms of its standard form. This allows us to immediately extract the *characteristic equation* which gives us the *damping ratio* and *natural frequency* of the system. The standard form is written as follows:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{1.19}$$

where the denominator of $G(s)$ is referred to as the characteristic equation, ξ being the damping ratio and ω_n being the natural frequency. In second order systems, we also saw several different metrics/specifications that allow us to quantify different aspects of the system. These are quantities such as the *peak-time*, *steady-state error*, *maximum overshoot* and *settling time*. we have also seen empirical means to determining the damping ratio of a second order system when direct measures are not convenient/possible. One such method would be the *logarithmic decrement method*. Lastly, studying dominant poles (slowest, closest to the $j\omega$-axis) are important as they most significantly effect the system.

§1.7 Linearization

In more complex non-linear systems, we can instead look for fixed/stable points and perform a first order (linear approximation) Taylor expansion about those points. These stable points are also known as *operating points*, denoted by $(\bar{x})$. As such, given some arbitrary function $f(x)$, it's first order approximation would be:

$$f(x) \approx f(\bar{x}) + \left(\frac{df(x)}{dx}\right)_{x=\bar{x}} (x - \bar{x}) \tag{1.20}$$

§1.8 Block Diagram

Definition 1.8.1. *A pictorial representation of the functions performed by each component of the system. These consist of functional blocks which represent operations on input signals, and directed channels which indicate the flow of signals.*

An example of a negative feedback block diagram is given in figure 1.3 below.

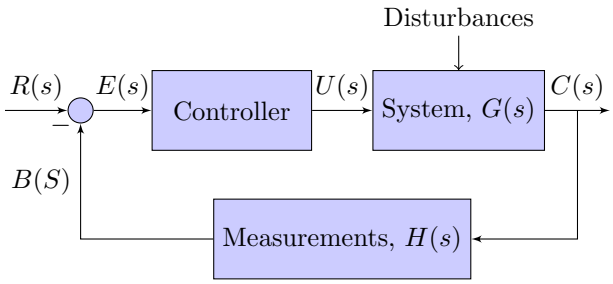

Figure 1.3: Sample Block Diagram

In general, it is more useful to consider feedback block diagrams, for which they can be decomposed into 3 transfer functions. These are the:

Open-Loop TF:

$$OLTF = \frac{B(s)}{E(s)} = G(s)H(s)$$

Feedforward TF:

$$FTF = \frac{C(s)}{E(s)} = G(s)$$

Closed-Loop TF:

$$CLTF = \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

§1.9 Automatic Controllers

An important and useful class of controllers are the *automatic controllers*.

§1.9.1 PID Controllers

We looked only at PID controllers in 30.104, standing for **P**roportional (present error), **I**ntegral (past errors) and **D**erivative (future errors) controllers. Recall that for a PID controller, the output from the controller into the system is given as:

Time-Domain

$$\begin{aligned} u(t) &= K_p e(t) + K_i \int_{-\infty}^{t} e(t)dt + K_d \frac{de(t)}{dt} \\ &= K_p \left[e(t) + \frac{1}{T_i} \int_{-\infty}^{t} e(t)dt + T_d \frac{de(t)}{dt} \right] \end{aligned} \tag{1.21}$$

Laplace (s)-Domain

$$U(s) = \left[K_p + \frac{K_i}{s} + K_d s \right] E(s) \tag{1.22}$$

$$G(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{1.23}$$

Note: the output from the controller to be fed into the system is an error dependent signal, where the error is denoted as $e(t)$.

§1.9.2 General Automatic Controllers

In general, an automatic controller compares the *actual value* of the plant/system output with the *desired value*, determines the deviation and produces a *control signal* that reduce to deviation an arbitrarily small value (or ideally zero). The block diagram representation of this is given in figure 1.4 below.

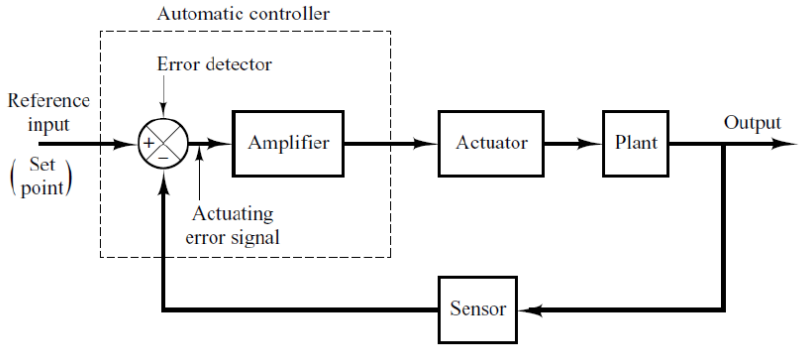

Figure 1.4: Automatic Controller Block Diagram

§1.10 System Stability

A key take away from our stability analysis from LTI systems in 30.104 is that the system is stable if all roots of the transfer function denominator have **negative real parts** (all poles lie on the left-half of the s-plane).

Alternatively, we can check what is known as the *Routh array*. Using this, an LTI system is stable if it satisfies the *Routh-Hurwitz stability criterion*, which states that the system is stable **iff** all elements in the first column of the Routh array are **positive**. Given some closed-loop transfer function ($H(s)$) of the form:

$$H(s) = \frac{B(s)}{A(s)} = \frac{\sum_{j=0}^{m} b_j s^{m-j}}{\sum_{j=0}^{n} a_j s^{n-j}} \tag{1.24}$$

Then the Routh array is given by:

$$\begin{array}{llllll} s^n & a_0 & a_2 & a_4 & a_6 & \cdots \\ s^{n-1} & a_1 & a_3 & a_5 & a_7 & \cdots \\ s^{n-2} & b_1 & b_2 & b_3 & b_4 & \cdots \\ s^{n-3} & c_1 & c_2 & c_3 & c_4 & \cdots \\ s^{n-4} & d_1 & d_2 & d_3 & d_4 & \cdots \\ \vdots & \vdots & \vdots & & & \\ s^2 & e_1 & e_2 & & & \\ s^1 & f_1 & & & & \\ s^0 & g_0 & & & & \end{array} \tag{1.25}$$

where we have the elements defined as:

$$\begin{aligned} b_j &= \frac{a_1 a_{2j} - a_0 a_{2j+1}}{a_1} \\ c_j &= \frac{b_1 a_{2j+1} - a_1 b_{j+1}}{b_1} \\ d_j &= \frac{c_1 b_{j+1} - b_1 c_{j+1}}{c_1} \\ &\vdots \end{aligned} \tag{1.26}$$

§1.10.1 Root Locus

Root Loci Construction Algorithm:

§1.10.2 Bode Diagram

Chapter 2

State-Space Method

This topic was briefly covered at the end of 30.104, however we will need to delve deeper into this technique for analyzing and controlling more sophisticated systems. The state-space approach is what falls under modern control theory (vs classical control theory, comprising largely of material learnt in 30.104).

§2.1 Extending Classical Control Theory

In the state-space approach dynamic systems are organized into a system of **first order** differential equations (may be non-linear). The general form for these ODE systems are:

$$\begin{aligned} \dot{\vec{x}} &= f(\vec{x}, \vec{u}, t) \\ \vec{y} &= g(\vec{x}, \vec{u}, t) \end{aligned} \tag{2.1}$$

If the system is LTI, then it takes the simpler form:

$$\begin{aligned} \dot{\vec{x}} &= \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \\ \vec{y} &= \mathbf{C}\vec{x} + \mathbf{D}\vec{u} \end{aligned} \tag{2.2}$$

where A is the *state matrix*, B the *input matrix*, C the *output matrix* and D the *direct transmission matrix* (**only A** has to be square). In this formalism, we can extend our intuition of the transfer function into what is known as the *transfer matrix* ($\mathbf{G}(s) = \mathbf{Y}(s)/\mathbf{U}(s)$).

$$\begin{aligned} & s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s) \\ \Rightarrow\ & (s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{B}\mathbf{U}(\mathrm{s}) \\ \Rightarrow\ & \mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(\mathrm{s}) \\ \Rightarrow\ & \mathbf{Y}(s) = \mathbf{C}\left[(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(\mathrm{s})\right] + \mathbf{D}\mathbf{U}(s) \\ \Rightarrow\ & \mathbf{Y}(s) = \left[\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\right]\mathbf{U}(s) \\ \Rightarrow\ & \boxed{\mathbf{G}(s) = \frac{\mathbf{Y}(s)}{\mathbf{U}(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}} \end{aligned} \tag{2.3}$$

where **X**, **U** and **Y** are the Laplace transforms of the state, input and output vectors respectively.

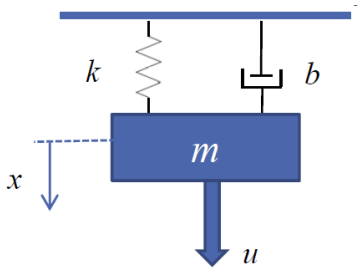

Figure 2.1: Spring-Mass-Damper System

Example:

Consider a spring-mass-damper system with an external input force u (figure 2.1). As such, the equation of motion describing this system is given by:

$$m\ddot{x} + b\dot{x} + kx = u \tag{2.4}$$

This system is seen to be second order. To use the state=space method, we require that we define new variable and reduce this to a system of first order differential equations. We can define our variables as follows;

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \end{aligned} \tag{2.5}$$

As such, this makes our second order ODE to take the following form:

$$\begin{aligned} & m\dot{x}_2 + bx_2 + kx_1 = u \\ \Rightarrow \quad & \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \end{aligned} \tag{2.6}$$

The poles of the uncontrolled/pre-controlled system can then be found by solving for the eigenvalues of **A**. In this case, we have that the poles are:

$$s_{\pm} = \frac{-\frac{b}{m} \pm \sqrt{\left(\frac{b}{m}\right)^2 - 4\left(\frac{k}{m}\right)}}{2} \tag{2.7}$$

The block diagram for this system is shown in figure 2.2 below.

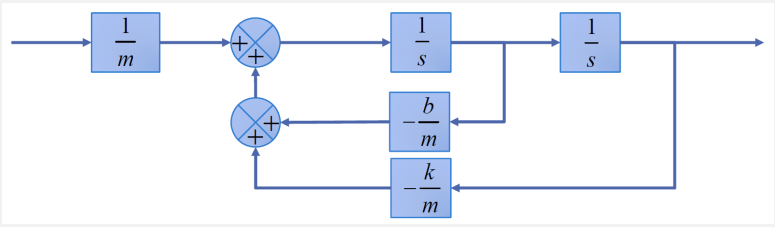

Figure 2.2: Spring-Mass-Damper Block Diagram

Note that the $\frac{1}{s}$ blocks indicate integration in the Laplace domain.

Note: Not every system is controllable. This will be touched on later on in the course.

Occasionally, the ODE describing the system (equation of motion) could involve derivatives of the inputs $u(t)$. However, the state-space approach is flexible and allows use to 'freely' define our coordinates $\{x_j\}$ such that these derivatives effectively drop out. In general, for some system described by an n^{th}-order linear differential equation of the form:

$$\overset{(n)}{y} + a_1 \overset{(n-1)}{y} + \cdots + a_{n-1}\dot{y} + a_n y = b_0 \overset{(n)}{u} + b_1 \overset{(n-1)}{u} + \cdots + b_{n-1}\dot{u} + b_n u \tag{2.8}$$

We can express this in the *controllable canonical form*. The name controllable canonical form will be further explained later in the course. For now, we can just take it as convention. There are other forms as well that are useful depending on the system in question. A list of the forms we will be looking at is given below:

1. **Controllable Canonical Form (CCF)**

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \tag{2.9}$$

$$y = \begin{bmatrix} b_n - a_n b_0 \\ b_{n-1} - a_{n-1} b_0 \\ \vdots \\ b_2 - a_2 b_0 \\ b_1 - a_1 b_0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + b_0 u \tag{2.10}$$

2. **Observable Canonical Form (OCF)**

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & -a_n \\ 1 & 0 & 0 & \dots & -a_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \dots & -a_2 \\ 0 & 0 & 0 & \dots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} b_n - a_n b_0 \\ b_{n-1} - a_{n-1} b_0 \\ \vdots \\ b_2 - a_2 b_0 \\ b_1 - a_1 b_0 \end{bmatrix} u \tag{2.11}$$

$$y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + b_0 u \tag{2.12}$$

   Notice that the transfer matrix for the OCF is the transpose of that for the CCF.

3. **Modal Canonical Forms**

- **Diagonal Canonical Form (DCF)** (only for distinct roots)

$$\begin{aligned} G(s) &= \frac{Y(s)}{U(s)} \\ &= \frac{b_0 s^{n-1} + \cdots + b_{n-1}s + b_n}{(s+p_1)(s+p_2)\cdots(s+p_n)} \\ &= b_0 + \frac{c_1}{s+p_1} + \frac{c_2}{s+p_2} + \cdots + \frac{c_n}{S+p_n} \end{aligned} \tag{2.13}$$

$$\Rightarrow \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -p_1 & 0 & 0 & \dots & 0 \\ 0 & -p_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & -p_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} u \tag{2.14}$$

$$y = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + b_0 u \tag{2.15}$$

If the system can be written in diagonal canonical form, this implies that the state-variables are completely decoupled (no-interacting 'subsystems').

- **Jordan Canonical Form (JCF)** (for degenerate roots)

$$\begin{aligned} G(s) &= \frac{Y(s)}{U(s)} \\ &= \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1}s + b_n}{(s+p_1)^3 (s+p_4)\cdots(s+p_n)} \\ &= b_0 + \frac{c_1}{(s+p_1)^3} + \frac{c_2}{(s+p_1)^2} + \frac{c_3}{s+p_1} + \cdots + \frac{c_n}{s+p_n} \end{aligned} \tag{2.16}$$

$$\Rightarrow \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{n-2} \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -p_1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -p_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & -p_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -p_{n-2} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & -p_{n-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & -p_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \tag{2.17}$$

§2.2 Transforming Between Canonical Forms

Given some system that we can model via a differential equation, after we have construction a linear system of differential equations from it, is there a way to easily move between one form to other canonical forms? Firsrt, consider some linear transformation T that takes our state-vector $\vec{x}$ to the desired form $\vec{z}$ (i.e $\vec{x} = T\vec{z}$). To derive the explicit form of this matrix, we first define some variables that we will require to work through the derivation. We define our initial state-space representation as:

$$\begin{aligned} \dot{\vec{x}} &= \mathbf{F}\vec{x} + \mathbf{G}\vec{u} \\ y &= \mathbf{H}\vec{x} + Ju \end{aligned} \tag{2.18}$$

We also define the canonical form representation that we want to transform to as:

$$\begin{aligned} \dot{\vec{z}} &= \mathbf{A}\vec{z} + \mathbf{B}\vec{u} \\ y &= \mathbf{C}\vec{z} + Du \end{aligned} \tag{2.19}$$

From here, we first establish the following:

$$\begin{aligned} & \dot{x} = T\dot{z} = \mathbf{FT}\vec{z} + \mathbf{G}u \\ \Rightarrow\ & \dot{\vec{z}} = \mathbf{T^{-1}FT}\vec{z} + \mathbf{T^{-1}G}u \equiv \mathbf{A}\vec{z} + \mathbf{B}\vec{u} \\ & y = \mathbf{HT}\vec{z} + Ju \equiv \mathbf{C}\vec{z} + Ju \end{aligned} \tag{2.20}$$

$$\Rightarrow \quad \boxed{\mathbf{A} = \mathbf{T^{-1}FT},\quad \mathbf{B} = \mathbf{T^{-1}G},\quad \mathbf{C} = \mathbf{HT},\quad D = J} \tag{2.21}$$

From here, depending on the specific canonical we intend to transform to, the transformation $\mathbf{T}$ can be solved as functions of $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and J differently. The respective solutions are discussed below (note that $\mathbf{G}$ is a vector).

§2.2.1 Transforming to CCF

Knowing the following relations,

$$\mathbf{AT^{-1}} = \mathbf{T^{-1}F} \tag{2.22}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix} \tag{2.23}$$

$$\mathbf{B} = \mathbf{T^{-1}G} \tag{2.24}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}^T \tag{2.25}$$

we also define the rows of the matrix $\mathbf{T}^{-1}$ as vectors $\vec{t}_j$. Explicitly, we have:

$$\mathbf{T}^{-1} = \begin{bmatrix} \dots & \vec{t}_1 & \dots \\ \dots & \vec{t}_2 & \dots \\ & \vdots & \\ \dots & \vec{t}_n & \dots \end{bmatrix} \tag{2.26}$$

from which we can see that:

$$\vec{t}_j = \vec{t}_{j-1}\mathbf{F}^{j-1} \tag{2.27}$$

$$\Rightarrow \boxed{\vec{t}_j = \vec{t}_1\mathbf{F}^{j-1}\mathbf{G} = \delta_{j,n}} \tag{2.28}$$

where $\delta_{j,n}$ is the Dirac-delta function. Writing the above result as a matrix, we also have:

$$\vec{t}_1 \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{G} & \mathbf{FG} & \dots & \mathbf{F}^{n-1}\mathbf{G} \\ \vdots & \vdots & \dots & \vdots \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix} \tag{2.29}$$

Then we see that in order to solve for the vectors $\vec{t}_j$, we require to be able to invert the matrix as seen above. As such, we define the matrix:

$$\boxed{\mathbf{C}_0 = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{G} & \mathbf{FG} & \dots & \mathbf{F}^{n-1}\mathbf{G} \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}} \tag{2.30}$$

as the **controllability matrix**. Given that this matrix is invertible, we can recover the row vectors of our transformation matrix $\mathbf{T}$ by the following relation:

$$\boxed{\vec{t}_j = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{G} & \mathbf{FG} & \dots & \mathbf{F}^{n-1}\mathbf{G} \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}^{-1} \mathbf{F}^{j-1}} \tag{2.31}$$

§2.2.2 Transforming to OCF

Here, we again start with the known transformation relations as follows: Knowing the following relations:

$$\mathbf{TA} = \mathbf{FT} \tag{2.32}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix}^T \tag{2.33}$$

$$\mathbf{C} = \mathbf{HT} \tag{2.34}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix} \tag{2.35}$$

We also define our transformation $\mathbf{T}$ via column vectors $\vec{t_j}$ instead (for mathematical convenience) as follows:

$$\mathbf{T} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \vec{t_1} & \vec{t_2} & \dots & \vec{t_n} \\ \vdots & \vdots & & \vdots \end{bmatrix} \tag{2.36}$$

Following a similar derivation as for the CCF transformation and utilizing the known OCF form of the matrix $\mathbf{A}$, we get that:

$$\vec{t_j} = \mathbf{F}^{j-1}\vec{t_1} \tag{2.37}$$

$$\Rightarrow \boxed{\mathbf{H}\vec{t_j} = \mathbf{HF}^{j-1}\vec{t_1} = \delta_{j,n}} \tag{2.38}$$

This result in matrix form is:

$$\begin{bmatrix} \dots & \mathbf{H} & \dots \\ \dots & \mathbf{HF} & \dots \\ & \vdots & \\ \dots & \mathbf{HF}^{n-1} & \dots \end{bmatrix} \vec{t_1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.39}$$

from which, we can define what is known as the *observability matrix*:

$$\boxed{\mathbf{O}_B = \begin{bmatrix} \dots & \mathbf{H} & \dots \\ \dots & \mathbf{HF} & \dots \\ & \vdots & \\ \dots & \mathbf{HF}^{n-1} & \dots \end{bmatrix}} \tag{2.40}$$

The transformation into observable canonical form is **only** possible if the observability matrix is non-singular (invertible). Using this result, we can then compute the columns of the transform

T which take the following relation:

$$\boxed{\vec{t_j} = \mathbf{F}^{j-1} \begin{bmatrix} \dots & \mathbf{H} & \dots \\ \dots & \mathbf{HF} & \dots \\ & \vdots & \\ \dots & \mathbf{HF}^{n-1} & \dots \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}} \tag{2.41}$$

§2.2.3 Transforming to DCF

Once again, knowing the following relations:

$$\mathbf{TA} = \mathbf{FT} \tag{2.42}$$

$$\mathbf{A} = \begin{bmatrix} -p_1 & 0 & 0 & \dots & 0 \\ 0 & -p_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & -p_n \end{bmatrix} \tag{2.43}$$

$$\mathbf{B} = \mathbf{T^{-1}G} \tag{2.44}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \tag{2.45}$$

where the transform matrix is defined as per in the OCF derivation ($\vec{t_j}$ being column vectors). From $\mathbf{TA} = \mathbf{FT}$, we get a set of eigenvector equations:

$$\boxed{\mathbf{F}\vec{t_j} = -p_j\vec{t_j} \equiv \tilde{p}_j\vec{t_j}} \tag{2.46}$$

As such, we can find the diagonal entries of **A** by solving for the eigenvalues of **F** ($\tilde{p}_j = -p_j$), while the eigenvectors ($\vec{v}_j$) constitute the columns of **T** up to some scale factor α_j. Explicitly, we have:

$$\boxed{\mathbf{A} = \begin{bmatrix} \tilde{p}_1 & 0 & 0 & \dots & 0 \\ 0 & \tilde{p}_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \tilde{p}_n \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} \vdots & & \vdots & \vdots \\ \alpha_1\vec{t}_1 & \dots & \alpha_{n-1}\vec{t}_{n-1} & \alpha_n\vec{t}_n \\ \vdots & & \vdots & \vdots \end{bmatrix}} \tag{2.47}$$

where the α_j's are chosen such that $[\mathbf{B}]_j = [\mathbf{T^{-1}G}]_j = 1$.

§2.3 Eigenvalues and the Characteristic Equation

Now that we know how to transform into the appropriate canonical form, we can actually very easily extract information about the system via the eigen-decomposition of **A** (as per the definition being used thus far). This comes from the fact that the eigenvalues of **A** are actually the **roots of the characteristic equation** of the system in the absence of input (control).

Note: The eigenvalues are *invariant* under linear transformation applied to $\mathbf{A}$.

$$\begin{aligned} \det\{\mathbf{T}(\mathbf{A}-s\mathbf{I})\mathbf{T}^{-1}\} &= \det\{\mathbf{T}\}\det\{(\mathbf{A}-s\mathbf{I})\}\det\{\mathbf{T}^{-1}\} \\ &= \det\{\mathbf{T}\}\det\{\mathbf{T}^{-1}\}\det\{(\mathbf{A}-s\mathbf{I})\} \\ &= \det\{\mathbf{A}-s\mathbf{I}\} \end{aligned} \tag{2.48}$$

Chapter 3

Systems of First Order ODEs

§3.1 Linear Algebra (Recap)

before we progress on, it would be useful to have a quick recap of the necessary linear algebra definitions and concepts for use in this course.

Definition 3.1.1. Square Matrix: *A matrix such that the number of rows and columns are the same. These are also called matrices of order n, where n is the number of rows/columns.*

Definition 3.1.2. Diagonal Matrix: *A matrix where all elements other than the main diagonal elements are zero.*

Definition 3.1.3. Singular Matrix: *A matrix where its determinant is zero, implying linear-dependence of at least one of the rows/columns.*

Definition 3.1.4. Transpose: *The transpose of a matrix, denoted $\mathbf{A}^T$, is whereby its columns and rows are interchanged (i.e. flipped about its main diagonal). If $\mathbf{A} = \mathbf{A}^T$, the matrix is said to be* ***symmetric****.*

Definition 3.1.5. Determinant: *The value associated with a square matrix $\mathbf{A}$, denoted as* $\det\{\mathbf{A}\} = |\mathbf{A}|$.

Determinant Properties:

1. $\det\{\mathbf{I}\} = 1$
2. $\det\{\mathbf{A}^T\} = \det\{\mathbf{A}\}$
3. $\det\{\mathbf{A}^{-1}\} = \frac{1}{\det\{\mathbf{A}\}}$
4. $\det\{\mathbf{AB}\} = \det\{\mathbf{A}\}\det\{\mathbf{B}\}$
5. $\det\{c\mathbf{A}\} = c^n \det\{\mathbf{A}\}$

§3.1.1 Matrix Exponentials

Recall that the exponential of a real parameter is defined by it's Taylor expansion.

$$e^x = \sum_{n=0}^{\infty} \frac{1}{n!} x^n \tag{3.1}$$

As such, we can extend this definition to the context of matrices, where we define a *matrix exponential* as follows:

$$\boxed{e^{\mathbf{A}t} = \sum_{n=0}^{\infty} \frac{\mathbf{A}^n t^n}{n!}} \tag{3.2}$$

Several useful **properties** of this are:

1. $\frac{d}{dt} e^{\mathbf{A}t} = \mathbf{A} e^{\mathbf{A}t}$
2. $e^{\mathbf{A}t} e^{\mathbf{A}s} = e^{\mathbf{A}(t+s)}$
3. $e^{(\mathbf{A}+\mathbf{B})t} = e^{\mathbf{A}t} e^{\mathbf{B}t} \iff \mathbf{AB} = \mathbf{BA}$
4. $\int_0^t e^{\mathbf{A}t} dt = \mathbf{A}^{-1} \left(e^{\mathbf{A}t} - \mathbf{I} \right)$

§3.2 Solving Homogeneous ODE systems

We can utilize matrix exponentials in order to generate solutions to system of first order differential equations for a LTI systems. Recall that for a single variable, homogeneous first order differential equation (LTI), we had:

$$\begin{aligned} \dot{x}(t) &= a \cdot x(t) \\ \Rightarrow \; x(t) &= x(0) e^{at} \end{aligned} \tag{3.3}$$

The idea is to now extend this to the multivariate case.

§3.2.1 Solution by Matrix Exponentials

For the multivariate-homogeneous first order differential equation (LTI), we very similarly can have:

$$\begin{aligned} \dot{\vec{x}}(t) &= \mathbf{A}\vec{x}(t) \\ \Rightarrow \; \vec{x}(t) &= e^{\mathbf{A}t} \vec{x}(0) \end{aligned} \tag{3.4}$$

A more general form of this solution can be written as:

$$\boxed{\vec{x}(t) = \mathbf{\Phi}(t, t_0) \vec{x}(t_0)} \tag{3.5}$$

where $\mathbf{\Phi}(t, t_0)$ is known as the *state-transition matrix*. A nice property of the state-transition matrix for $t_0 = 0$ is that $\mathbf{\Phi}^{-1}(t) = \mathbf{\Phi}(-t)$. However, an infinite series is not the most efficient means to compute a solution. As such, what we can do is *diagonalize* the matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} \tag{3.6}$$

where $\mathbf{D}$ is a diagonal matrix of ordered (decreasing from left to right) eigenvalues and the columns of $\mathbf{V}$ are the associated ordered eigenvectors. As such, we have that:

$$\begin{aligned} e^{\mathbf{A}t} &= e^{\mathbf{V}\mathbf{D}\mathbf{V}^{-1}t} \\ &= \mathbf{V}e^{\mathbf{D}t}\mathbf{V}^{-1} \end{aligned} \tag{3.7}$$

and since $\mathbf{D}$ is diagonal, $\exp(\mathbf{D}t)$ can be computed by simply exponentiating the individual diagonal entries of $\mathbf{D}$ with the added factor t in the exponent. Explicitly, we have:

$$\boxed{e^{\mathbf{A}t} = \mathbf{V}\begin{bmatrix} e^{\lambda_1 t} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 t} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_n t} \end{bmatrix}\mathbf{V}^{-1}} \tag{3.8}$$

where λ_j are the eigenvalues of $\mathbf{A}$. However, this method fails if we encounter a matrix with *degeneracies* (eigenvalues with multiplicity > 1). For these, we require finding the *generalized eigenvectors* and employ the use of the *Jordan canonical form* $\mathbf{J}$ (which replaces the diagonal matrix $\mathbf{D}$). For some matrix $\mathbf{A}$ with one unique eigenvalue of multiplicity n, we have:

$$A = \mathbf{S}\mathbf{J}\mathbf{S}^{-1} \tag{3.9}$$

where the explicit form of $\mathbf{J}$ is given by:

$$\mathbf{J} = \begin{bmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda \end{bmatrix} = \lambda\mathbf{I} + \mathbf{N} \tag{3.10}$$

and $\mathbf{N}$ is a matrix of ones on the upper off-diagonal, and has the property of *nilpotency* ($\mathbf{N}^m = \mathbf{0}$ for some $m \in \mathbb{Z}^+$). From this, it can be shown that:

$$\boxed{e^{\mathbf{A}t} = \mathbf{S}\begin{bmatrix} e^{\lambda t} & \frac{1}{1!}te^{\lambda t} & \cdots & \frac{1}{(n-1)!}t^{n-1}e^{\lambda t} \\ 0 & e^{\lambda t} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \frac{1}{1!}te^{\lambda t} \\ 0 & 0 & \cdots & e^{\lambda t} \end{bmatrix}\mathbf{S}^{-1}} \tag{3.11}$$

The associated generalized eigenvectors can then be computed by:

$$\boxed{(\mathbf{A} - \lambda\mathbf{I})\vec{v}_j = \vec{v}_{j-1}} \tag{3.12}$$

(i.e. finding the null space of $(\mathbf{A} - \lambda\mathbf{I})^j$ for generalized eigenvector $\vec{v})_j$). If our system is such that there are several **different** degenerate eigenvalues, then the matrix $e^{\mathbf{J}t}$ would be block diagonal (each block having the upper triangular form as above), and we find will need to find the corresponding generalized eigenvectors for each block separately.

§3.2.2 Solution by Laplace Transform

As per traditional control theory, we can also apply the Laplace transform to the first order homogeneous differential equation (LTI) to attain the state-transition matrix:

$$\begin{aligned} & \mathcal{L}\left\{\dot{\vec{x}}(t) = \mathbf{A}\vec{x}(t)\right\} \\ \Rightarrow\ & s\vec{X}(s) - \vec{x}(0) = \mathbf{A}\vec{X}(s) \\ \Rightarrow\ & \vec{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\vec{x}(0) \\ \Rightarrow\ & \boxed{\vec{x}(t) = \mathcal{L}^{-1}\left\{(s\mathbf{I} - \mathbf{A})^{-1}\right\}\vec{x}(0)} \end{aligned} \tag{3.13}$$

where the inverse Laplace operation on the matrix is done element wise. As such, we have the equality relation between the equivalent methods for computing the state-transition matrix:

$$\boxed{\mathbf{\Phi}(t) = \mathcal{L}^{-1}\left\{(s\mathbf{I} - \mathbf{A})^{-1}\right\} = e^{\mathbf{A}t}} \tag{3.14}$$

§3.3 Solving Non-Homogeneous ODE Systems

To actively control a system. we require that there be an input $u(t)$ into the system, causing the differential equation describing the system to take on a non-homogeneous form. As such we will need to know how to solve such systems. Let us first quickly run through solving a single-variable LTI non-homogeneous ODE. Given the LTI equation:

$$\dot{x} = ax + bu \tag{3.15}$$

By applying a factor of e^{-at}, using the product rule identity and integrating with respect to time gives us:

$$\begin{aligned} e^{-at}x(t) - x(0) &= \int_0^t e^{-a\tau}bu(\tau)d\tau \\ \Rightarrow\ x(t) &= e^{at}x(0) + e^{at}\int_0^t e^{-a\tau}bu(\tau)d\tau \end{aligned} \tag{3.16}$$

Now, we can extend this solution into higher dimensions, where starting from the equations:

$$\begin{aligned}
&\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \\
\Rightarrow\ &\dot{\vec{x}} - \mathbf{A}\vec{x} = \mathbf{B}\vec{u} \\
\Rightarrow\ &e^{-\mathbf{A}t}\left(\dot{\vec{x}} - \mathbf{A}\vec{x}\right) = e^{-\mathbf{A}t}\mathbf{B}\vec{u} \\
\Rightarrow\ &\frac{d}{dt}\left(e^{-\mathbf{A}t}\vec{x}(t)\right) = e^{-\mathbf{A}t}\mathbf{B}\vec{u} \\
\Rightarrow\ &(e^{-\mathbf{A}t}\vec{t} - \vec{x}(0) = \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}\vec{u}(\tau)d\tau \\
\Rightarrow\ &\boxed{\vec{x}(t) = e^{\mathbf{A}t}\vec{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\vec{u}(\tau)d\tau}
\end{aligned} \tag{3.17}$$

Or more generally:

$$\boxed{\vec{x}(t) = \mathbf{\Phi}(t)\vec{x}(0) + \int_0^t \mathbf{\Phi}(t-\tau)\mathbf{B}\vec{u}(\tau)d\tau} \tag{3.18}$$

where we have that the integral term is actually a *convolution*.

§3.3.1 Analytical Solution for Unit-Step Inputs

Consider the case where we have:

$$\vec{u}(t) = \vec{1}(t) = 1(t)\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} = 1(t)\vec{k} \tag{3.19}$$

where $1(t)$ indicates the unit-step function. Substituting this into the solution we saw above, we get:

$$\begin{aligned}
&\vec{x}(t) = e^{\mathbf{A}t}\vec{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\vec{1}(\tau)d\tau \\
\Rightarrow\ &\vec{x}(t) = e^{\mathbf{A}t}\left(\vec{x}(0) + \mathbf{A}^{-1}(\mathbf{I} - e^{-\mathbf{A}t})\right)\mathbf{B}\vec{k}
\end{aligned} \tag{3.20}$$

$$\Rightarrow\ \boxed{\vec{x}(t) = e^{\mathbf{A}t}\vec{x}(0) + \mathbf{A}^{-1}(e^{\mathbf{A}t} - \mathbf{I})\mathbf{B}\vec{k}} \tag{3.21}$$

§3.4 Essential LTI Properties

Controllability and *observability* are 2 major concepts of modern control theory. They were introduced by R. E. Kalman in 1959/60. To get a better picture of what these are, here is a high level overview of the implications of these 2 concepts.

- **Controllability**: *In order to be able to do whatever we want with the dynamic system under control inputs, the system must be controllable.*

- **Observability**: *In order to see what is going on inside the system under observation (the internal states), the system must be observable.*

The concepts of controllability and observability are derived from linear system theory, so knowledge of linear algebra is key to understanding them. Controllability and observability are dual concepts and should be understood as a whole rather than as separate properties.

§3.4.1 Complete State Controllability

A system is said to be state controllable at $t = t_0$ if it is possible to construct an *unconstrained control signal* that will transfer an initial state to any final state within some finite time interval ($t_0 < t < t_f$). If every state is controllable, the system is said to be *completely state controllable.* Before continuing on, let us look at an extremely powerful theorem and how it can help us in our control analysis.

Theorem 3.4.1. *The **Cayley-Hamilton's** theorem states that the matrix* $\mathbf{A}$ *satisfies its own characteristic equation. Explicitly, given that*:

$$|\mathbf{A} - \lambda\mathbf{I}| = \lambda^n + a_1\lambda^{n-1} + ... + a_{n-1}\lambda + a_n = 0 \tag{3.22}$$

We can also write:

$$\mathbf{A}^n + a_1\mathbf{A}^{n-1} + ... + a_{n-1}\mathbf{A} + a_n\mathbf{I} = 0 \tag{3.23}$$

A result of this theorem is that any $n \times n$ matrix $\mathbf{A}$, any power of $\mathbf{A}$ can be written as a linear combination of powers of $\mathbf{A}$ up to the $(n-1)$-th power. Explicitly, we have:

$$\mathbf{A}^m = \sum_{k=0}^{n-1} \alpha_k \mathbf{A}^k \tag{3.24}$$

This is useful for us because this gives us another computationally efficient means of getting matrix exponentials! That is to say, we can write any square matrix exponential of dimension $n \times n$ as:

$$\boxed{e^{\mathbf{A}t} = \sum_{k=1}^{n-1} \alpha_k(t)\mathbf{A}^k} \tag{3.25}$$

As such, given a system $\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u}$, we know from the previous section that the solution is:

$$\vec{x}(t) = e^{\mathbf{A}t}\vec{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\vec{u}(\tau)d\tau \tag{3.26}$$

Without loss of generality, we can define our coordinates such that we aim for our final state to

vanish $\vec{x}(t = t_f) = \vec{0}$. As such, we have:

$$\vec{0} = e^{\mathbf{A}t_f}\vec{x}(0) + \int_0^{t_f} e^{\mathbf{A}(t_f-\tau)}\mathbf{B}\vec{u}(\tau)d\tau \tag{3.27}$$

$$\begin{aligned}
\Rightarrow \quad \vec{x}(0) &= -e^{-\mathbf{A}t_f}\int_0^{t_f} e^{\mathbf{A}(t_f-\tau)}\mathbf{B}\vec{u}(\tau)d\tau \\
&= -\int_0^{t_f}\sum_{k=0}^{n-1}\alpha_k(\tau)\mathbf{A}^k\mathbf{B}u(\tau)d\tau \\
&= -\sum_{k=0}^{n-1}\left[\mathbf{A}^k\mathbf{B}\int_0^{t_f}\alpha_k(\tau)u(\tau)d\tau\right] \\
&= -\sum_{k=0}^{n-1}\left[\mathbf{A}^k\mathbf{B}\beta_k\right] \\
&= \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}
\end{aligned} \tag{3.28}$$

For which we can see that in order to solve such a system (solve of β_k values), the matrix $\begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}$ **must** be invertible! This matrix is known as the *controllability matrix*. Recall that to check invertibility, we can look at the matrix systematically using the following methods (sequentially):

1. Check if the matrix is square ($n \times n$).
2. Check if the columns are *linearly independent*.
3. Equivalent to the second step, check the *determinant* of the matrix.

Note: In the above derivation, we have defined:

$$\beta_k \equiv \int_0^{t_f}\alpha_k(\tau)u(\tau)d\tau \tag{3.29}$$

§3.5 Complete Output Controllability

In the practical design of control systems, it is usually desired to control the **output** rather than just the internal states of the system. Complete state controllability is neither necessary nor sufficient for controlling the output of the system and thus, a separate criterion is needed. System is said to be *completely output controllable* if it is possible to construct an *unconstrained control vector* that will transfer any given initial output to final output in a finite time interval. If the matrix:

$$\begin{bmatrix}\mathbf{CB} & \mathbf{CAB} & \mathbf{CA}^2\mathbf{B} & \ldots & \mathbf{CA}^{n-1}\mathbf{B}\end{bmatrix} \tag{3.30}$$

is rank m (where m is the number of outputs), it can be proven that the system is completely output controllable.

§3.5.1 Uncontrollability Systems and Stabilizability

An *uncontrollable system* has a subsystem that is physically disconnected from the input. That is to say, no matter that the input and is over some finite time interval, there is no way to affect the state(s) of this subsystem. However, some of these systems can still be controllable if the uncontrollable modes are stable and the unstable modes are controllable. This type of system is said to be *stabilizable*.

§3.6 Complete Observability

A system is said to be *completely observable* if every state at $t = t_0$ can be determined from the observation of $y(t)$ over a finite time interval $t_0 \leq t \leq t_f$ (i.e. every transition of the state eventually affects every element of the output vector). Concept of observability is important in practice where there are difficulties during state feedback control when some of the state variables are not accessible for direct measurement. Consider again some given system:

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \tag{3.31}$$

$$\vec{y} = \mathbf{C}\vec{x} + \mathbf{D}\vec{u} \tag{3.32}$$

We know that the solution of the output is given by:

$$\vec{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{x}(0) + \mathbf{C}\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u} \tag{3.33}$$

However, we also know that the last 2 terms in the expression above are completely accounted for, so it is sufficient to only consider the first term. As such, consider:

$$\tilde{\vec{y}}(t) = \mathbf{C}e^{\mathbf{A}t} \tag{3.34}$$

for which we can again apply the Cayley-Hamilton's theorem to get:

$$\begin{aligned} \tilde{\vec{y}}(t) &= \sum_{k=0}^{n-1} \alpha_k \mathbf{C}\mathbf{A}^k \vec{x}(0) \\ &= \begin{bmatrix} \alpha_0(t) & \alpha_1(t) & \dots & \alpha_{n-1}(t) \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}_{nm \times n} \vec{x}(0) \end{aligned} \tag{3.35}$$

As such, the matrix $\begin{bmatrix} \mathbf{C} & \mathbf{CA} & \dots & \mathbf{CA}^{n-1} \end{bmatrix}^T$ is known as the *observability matrix* $\mathbf{O}$. With this, we can see that the condition of *complete observability* of the system is for the observability matrix to be of rank n (or has at least n linearly independent column vectors).

§3.7 Principle of Duality

The notion of controllability and observability have a deep implicit relationship. This underlying relationship was made explicit by the *principle of duality* conceived by *Kalman*, connecting controllability and observability in an elegant framework. The principle of duality can be stated as follows. Considering we again have the system (S_1):

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \tag{3.36}$$

$$\vec{y} = \mathbf{C}\vec{x} \tag{3.37}$$

Then the dual system of S_1, (S_2) is written as:

$$\dot{\vec{z}} = \mathbf{A}^\dagger\vec{z} + \mathbf{C}^\dagger\vec{v} \tag{3.38}$$

$$\vec{y} = \mathbf{B}^\dagger\vec{z} \tag{3.39}$$

where † indicates the conjugate transpose of a matrix. What this allows us to see is that given S_1 is state-controllable (or observable), then we immediately know that S_2 is observable (state-controllable). Furthermore, for a partially observable system, if the unobservable modes are stable and the observable modes are unstable, the system is said to be *detectable*. Concept of *detectability* is dual to the concept of *stabilizability*.

Chapter 4

Regulator and Controller Design

In this chapter, we will be looking at how to design controllers for a stabilizable system. First things first, it is good to know the difference between regulator systems and control systems. Regulator systems have a reference input that is constant, whereas control systems have reference inputs that are time-varying. Here, we will be studying both regulator and control systems.

§4.1 Pole-Placement

The *Pole-Placement method* is a regulator system and is similar to root-locus based design of placing closed-loop poles at desired locations seen in *30.101 Systems and Control*. However, while root-locus method focuses on the dominant poles, pole placement places **all** closed loop poles at desired locations. This technique assumes all state variables are *measurable* (observable) and *available* for feedback. If a system is completely state controllable, then the poles of the closed-loop system may be placed at any desired location by means of *state feedback*. There are 3 ways to approach Pole-Placement design:

1. **Method 1**: Using the Controllable Canonical Form as a start.
2. **Method 2**: Using direct substitution and comparing with desired characteristic equation. This is generally **not** recommended for high order systems since it is not easily implemented/scalable on a classical computer program.
3. **Method 3**: Using *Ackermann's Formula*.

Because of our assertions above, it is necessary to check for complete state controllability prior to Pole-Placement design. The pole-placement approach to control is constructed as follows. For the system:

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \vec{B}u \tag{4.1}$$

$$\vec{y} = \mathbf{C}\vec{x} \tag{4.2}$$

we choose the inputs to be functions of the states:

$$u = -\vec{K}\vec{x} \tag{4.3}$$

where $\vec{K}$ is a row vector. We also take that there is no reference input, and the goal is simply to maintain a zero output, where the output without control would otherwise deviate from zero due to disturbances. As such, we aim to have the nonzero output be returned to zero via *state feedback*. Substituting 4.1 into 4.1, we get:

$$\begin{aligned} \dot{\vec{x}} &= \mathbf{A}\vec{x} + \vec{B}\left(-\vec{K}\vec{x}\right) \\ &= \left(\mathbf{A} - \vec{B}\vec{K}\right)\vec{x} \end{aligned} \tag{4.4}$$

for which the solution to this differential equation can be found via matrix exponentiation:

$$\boxed{\vec{x}(t) = e^{(\mathbf{A}-\vec{B}\vec{K})t}}\vec{x}(0) \tag{4.5}$$

The stability and transient response from this input is then determined by the eigenvalues of the matrix $\mathbf{A} - \vec{B}\vec{K}$. These eigenvalues are known as *regulator poles*. Let's now look to how we can actually implement this control technique via the 3 methods mentioned above.

§4.1.1 Method 1: Direct Computation from CCF

Since the assumption made for the pole-placement approach is that the system is controllable, it is natural that we explore the CCF for its analysis. Defining the vector $\vec{K}$ as:

$$\vec{K} = \begin{bmatrix} \beta_n & \beta_{n-1} & \dots & \beta_1 \end{bmatrix} \tag{4.6}$$

To compute the regulator poles, we look at the following determinant and associated characterisitic equation:

$$\begin{aligned} \det\left(s\mathbf{I} - \mathbf{A} + \vec{B}\vec{K}\right) &= \left| s\mathbf{I} - \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \beta_n & \beta_{n-1} & \dots & \beta_1 \end{bmatrix} \right| \\ &= \left| \begin{bmatrix} s & -1 & 0 & \dots & 0 \\ 0 & s & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1 \\ \beta_n + a_n & \beta_{n-1} + a_{n-1} & \beta_{n-2} + a_{n-2} & \dots & \beta_1 + a_1 \end{bmatrix} \right| \\ &= s^n + (a_1 + \beta_1)\, s^{n-1} + \cdots + (a_{n-1} + \beta_{n-1})\, s + (a_n + \beta_n) = 0 \end{aligned} \tag{4.7}$$

Let's say that the desired poles from implementing our controller are denoted by α_j, we can then write our desired characteristic equation as:

$$(s+\alpha_1)(s+\alpha_2)\dots(s+\alpha_n) = s^n + \mu_1 s^{n-1} + \dots + \mu_{n-1}s + \mu_n = 0 \tag{4.8}$$

where μ_j are the expansion coefficients. Comparing this with the result in 4.7, we get that:

$$\beta_j = \mu_j - a_j \tag{4.9}$$

As such, the gain vector $\vec{K}$ now has the closed-form solution:

$$\boxed{\vec{K} = \begin{bmatrix}(\mu_n - a_n) & (\mu_{n-1} - a_{n-1}) & \dots & (\mu_1 - a_1)\end{bmatrix}} \tag{4.10}$$

The main difficulty of this method is transforming the system into the CCF.

§4.1.2 Method 2: Direct Substitution

For simple systems of order $n \leq 3$, it is often simpler to directly substitute the gain matrix $\vec{K}$ into the desired characteristic equation. Let's look at a $n = 3$ system. The gain vector can be written as:

$$\vec{K} = \begin{bmatrix}k_1 & k_2 & k_3\end{bmatrix} \tag{4.11}$$

For which, the determinant of the desired matrix would be:

$$\det\left(s\mathbf{I} - \mathbf{A} + \vec{B}\vec{K}\right) = (s+\alpha_1)(s+\alpha_2)(s+\alpha_3) = 0 \tag{4.12}$$

Due to the low order, factorization of $\det\left(s\mathbf{I} - \mathbf{A} + \vec{B}\vec{K}\right)$ is rather simple to do, so we can directly compare coefficients to determine α_j and thus k_j. This method (used in 30.101) however, is extremely tedious for higher order systems.

§4.1.3 Method 3: Ackermann's Formula

For this method, we start at the general form:

$$\dot{\vec{x}} = \left(\mathbf{A} - \vec{B}\vec{K}\right)\vec{x} \equiv \tilde{\mathbf{A}}\vec{x} \tag{4.13}$$

We know that the characteristic equation to find the regulator poles is given by:

$$\begin{aligned}\det\left(s\mathbf{I} - \tilde{\mathbf{A}}\right) &= (s+\alpha_1)(s+\alpha_2)\dots(s+\alpha_n) \\ &= s^n + \mu_1 s^{n-1} + \dots + \mu_{n-1}s + \mu_n = 0\end{aligned} \tag{4.14}$$

from which, we can employ the Cayley-Hamilton theorem to get:

$$\phi(\tilde{\mathbf{A}}) = \tilde{\mathbf{A}}^n + \mu_1\tilde{\mathbf{A}}^{n-1} + \dots + \mu_{n-1}\tilde{\mathbf{A}} + \mu_n\mathbf{I} = \mathbf{0} \tag{4.15}$$

It can be worked out that;

$$\tilde{\mathbf{A}}^n = \mathbf{A}^n - \sum_{k=0}^{n-1} \mathbf{A}^{n-1-k}\vec{B}\vec{K}\tilde{\mathbf{A}}^k \tag{4.16}$$

$$\begin{aligned}\phi(\mathbf{A}) &= \sum_{k=0}^{n-1} \mathbf{A}^k \vec{B}\left(\sum_{l=0}^{n-1-k} \mu_{n-1-l}\vec{K}\tilde{\mathbf{A}}^l\right) \\ &= \begin{bmatrix} \mathbf{B} & \dots & \mathbf{A}^{n-2}\mathbf{B} & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \begin{bmatrix} \alpha_{n-1}\vec{K} + \alpha_{n-2}\vec{K}\tilde{\mathbf{A}} + \dots + \vec{K}\tilde{\mathbf{A}}^{n-1} \\ \vdots \\ \alpha_1\vec{K} + \vec{K}\tilde{\mathbf{A}} \\ \vec{K} \end{bmatrix}\end{aligned} \tag{4.17}$$

where we see that the left multiplied matrix above is indeed the state-controllability matrix. Since we have asserted that this system is indeed controllable, this means the controllability matrix is invertible, granting us the closed form solution for $\vec{K}$ being:

$$\boxed{\vec{K} = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}^{-1} \phi(\mathbf{A})} \tag{4.18}$$

The equation for $\phi(\mathbf{A})$ seems rather complicated, but actually, we can generate it by imposing our desired closed loops poles. Given some desired characteristic equation, we simply replace all the s (poles) variables with $\mathbf{A}$ matrices to get $\phi(\mathbf{A})$.

While it is often desired to place the CL poles far away from the $j\omega$ axis for fast system response, it is possible that the required input signals are extremely large, which could result in the system becoming nonlinear and require larger/heavier actuators. Another alternative to pole-placement is based on the *quadratic optimal control* approach which places the poles such that it balances the need for acceptable response and control energy necessary. Remember the gain matrix $\vec{K}$ is not unique to a given system and depends on the desired closed-loop poles. Ideally, higher order systems are examined by computer simulations, and the response characteristics of the system for different $\vec{K}$ matrices should be chosen for best overall performance.

§4.2 Servo Systems and Integral Control

Having looked at regulators, we will now be looking at *servo systems* and how it is incorporated into state feedback. These are different from the previously covered regulator systems, for which we will soon be learning this difference. As a first step, recall that the *type* of a system is defined as the number of poles at the origin (multiplicity of the $s = 0$ pole). This is **not** to be confused with the order of the system.

§4.2.1 Introduction to Servo Systems

What exactly is a servo system? In servo systems, we are tackling a more complex problem, which is tracking a reference input that is **time-dependent** (control the system with zero steady-state error). Some examples of this would be servo motors, veer-control cars, etc. We have studied one instance of servo systems in 30.104, known as the PID control system. This consisted of 3 components:

1. Proportional Gain (P)
2. Integral Gain (I)
3. Derivative Gain (D)

In the regulator problem we saw earlier, this could roughly equate to a rudimentary PD system (due to the defined coordinates and gain form). How do we then incorporate integral action into our control system?

Note: If a system is type 1, the system already has an embedded integrator ($1/s$), so no additional integral action is required. In fact, the type of the system tells us the number of innate integrators our system has.

We will be looking mostly at type 0 systems since these do not have innate integrators. Assuming that $y = x_1$ (without loss of generality), the corresponding configuration for a (SISO) servo control system is given by:

$$u = -\vec{K}\vec{x} + k_1 r \tag{4.19}$$

where $K = [k_1 \quad k_2 \quad \dots \quad k_n]$ and r is the value we aim to control x_1 at. As such, we have the input explicitly written as:

$$\boxed{u = -\begin{bmatrix}0 & k_2 & \dots & k_n\end{bmatrix}\begin{bmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{bmatrix} + k_1(r - y)} \tag{4.20}$$

Notice that if our reference is $r = 0$, we simply revert to the regulator problem. Plugging in this form into our state-space representation:

$$\begin{aligned}\dot{\vec{x}}(t) &= \mathbf{A}\vec{x} + \mathbf{B}(-\vec{K}\vec{x} + k_1 r) \\ &= (\mathbf{A} - \mathbf{B}\vec{K})\vec{x}(t) + \mathbf{B}k_1 \cdot r(t)\end{aligned} \tag{4.21}$$

First, let us list out the goals we want to achieve for this system to possible reduce the problem slightly (given that $r(t)$ is a step input). We want the system to be:

1. Asymptotically stable
2. $y(t \to \infty)$ approaches a constant value r
3. $u(t \to \infty)$ approaches zero

As such, let us look at the steady-state form of our servo system:

$$\dot{\vec{x}}(t) = (\mathbf{A} - \mathbf{B}\vec{K})\vec{x}(t \to \infty) + \mathbf{B}k_1 \cdot r(t \to \infty) \tag{4.22}$$

Furthermore, since we have asserted that $r(t)$ is a step input, we have that $r(t) = r(t \to \infty)$, which gives us:

$$\dot{\vec{x}}(t) - \dot{\vec{x}}(\infty) = (\mathbf{A} - \mathbf{B}\vec{K})[\vec{x}(t) - \vec{x}(\infty)] \tag{4.23}$$

Now, it is convenient to define the error vector:

$$\vec{e}(t) = \vec{x}(t) - \vec{x}(\infty) \tag{4.24}$$

for which we want this term to vanish. As such, we can rewrite our system that we intend to solve as:

$$\boxed{\dot{\vec{e}}(t) = (\mathbf{A} - \mathbf{B}\vec{K})\vec{e}(t)} \tag{4.25}$$

This is great because we now effectively are back to the regulator problem! We formally express the study of this system as the *error dynamics*. In this system, given that it is completely state controllable, we can again use the pole-placement technique to control/regulate the error. By definition, we see that the steady-state value of the states and inputs are found by taking the limit as $t \to \infty$. This gives us:

$$\begin{aligned} \mathbf{x}(\infty) &= -(\mathbf{A} - \mathbf{B}\mathbf{K})^{-1}\mathbf{B}k_1 r \\ u(\infty) &= -\mathbf{K}\mathbf{x}(\infty) + k_1 r \\ &= -k_1 x_1(\infty) + k_1 r \end{aligned} \tag{4.26}$$

For systems with no intrinsic integrator (type 0), we will want to insert an integrator in the feedforward path between the error *comparator* and *plant*. We also have 4 scalar parameters that we are concerned with:

1. **Control Signal**: $u(t) = -\vec{K}\vec{x} + k_1\xi$
2. **Output Signal**: $y(t) = \mathbf{C} \cdot \vec{x}$
3. **Reference Input**: r
4. **Integrator Output**: $\dot{\xi} \equiv r - y = r - \mathbf{C} \cdot \vec{x}$

With the way we defined the integrator output, we can also study what is known as the *integrator dynamics* via the differential equation above:

$$\dot{\xi} = r - \mathbf{C} \cdot \vec{x} \tag{4.27}$$

Combining this with the differential equation associated to the plant (system dynamics), we get a block linear system:

$$\begin{bmatrix} \dot{\vec{x}} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \vec{x} \\ \xi \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} u + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} r \tag{4.28}$$

Noting here that $\mathbf{C}$ is a row vector since we are still working with SISO systems. If we once again take r as a step function, then we can utilize the same trick we did in 4.2.1 to get:

$$\begin{bmatrix} \dot{\vec{x}}(t) - \dot{\vec{x}}(\infty) \\ \dot{\xi}(t) - \dot{\xi}(\infty) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \vec{x}(t) - \vec{x}(\infty) \\ \xi(t) - \xi(\infty) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} [u(t) - u(\infty)] \tag{4.29}$$

since $r(t) = r(\infty)$. From here, we can define the *augmented error vector* as:

$$\vec{e}(t) \equiv \begin{bmatrix} \vec{x}(t) - \vec{x}(\infty) \\ \xi(t) - \xi(\infty) \end{bmatrix} \tag{4.30}$$

$$\Rightarrow \boxed{\dot{\vec{e}}(t) = \tilde{\mathbf{A}}\vec{e}(t) + \tilde{\mathbf{B}}u_e, \quad u_e = -\vec{K}_e\vec{e}(t)} \tag{4.31}$$

Where in the boxed equations above, we have defined the new (tilde) matrices/vectors as follows:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}, \quad \tilde{\vec{K}} = \begin{bmatrix} \vec{K} & -k_1 \end{bmatrix} \tag{4.32}$$

recalling that over here, $\mathbf{B}$ is a column vector. The system effectively reduces to:

$$\boxed{\dot{\vec{e}}(t) = \left(\tilde{\mathbf{A}} - \tilde{\mathbf{B}}\vec{K}\right)\vec{e}(t)} \tag{4.33}$$

for which we can use one of the 3 pole-placement methods earlier discussed to solve for the gain vector $\vec{K}$.

Example

Consider a motor speed system defined by:

$$\frac{Y(s)}{U(s)} = \frac{1}{s+3} \tag{4.34}$$

We also say that the desired system to have integral control and 2 poles at $s = -5, -5$. How do we do this? Well, we first write out the EOM of the sytem:

$$\dot{y} + 3y = u \tag{4.35}$$

$$\Rightarrow \ \dot{x} = -3x + u, \quad y = x \tag{4.36}$$

So we see that $\mathbf{A} = -3$, $\mathbf{B} = 1$, $\mathbf{C} = 1$. We now want to make it into the error dynamics form, which by the formula, gives us:

$$\tilde{\mathbf{A}} = \begin{bmatrix} -3 & 0 \\ -1 & 0 \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{4.37}$$

Which allows us to find our closed-loop characteristic equation as follows:

$$\begin{aligned} & \tilde{\mathbf{K}} = \begin{bmatrix} k_1 & -k_1 \end{bmatrix} \\ \Rightarrow \ & \tilde{\mathbf{A}} - \tilde{\mathbf{B}}\mathbf{K} = \begin{bmatrix} -3-k_1 & k_1 \\ -1 & 0 \end{bmatrix} \\ \Rightarrow \ & \det\left(\tilde{\mathbf{A}} - \tilde{\mathbf{B}}\mathbf{K} - s\mathbf{I}\right) = s^2 + (3+k_1)s + k_1 = 0 \end{aligned} \tag{4.38}$$

§4.3 State Observers

When designing the pole-placement regulator system, all state variables were assumed to be present. In practice however, not all state variables are available for feedback or they may be noisy/unreliable. As such. we require a scheme to estimate unavailable or unknown state variables. First, let us begin with several definitions:

Definition 4.3.1. Observation: *the process of estimating state variables in the deterministic setting (e.g. Luenberger Observers).*

Definition 4.3.2. Estimation: *The process of estimating state variables in the stochastic setting (e.g. Kalman Filters).*

If the state observer observes **all** state variables, this is what is known as a *full-order state observer*. Many a time, this is unnecessary as observation is needed only for the unmeasurable states. An observer that estimates fewer that the dimension of the state vector is a *reduced-order state observer*, where if the order of the reduced-order state observer is the minimum possible, it is called the *minimum-order state observer*.

§4.3.1 Luenberger Observers

In general, the state observer estimates the state variables based on the measurements of the output and control variables. State observers are also called Luenberger Observers in honour of David Luenberger (1964). First, recall the concept of Observability, a state observer can only be designed if and only if the observability condition is satisfied. Now here is the problem. We are given the input u, output y and the system model $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. However, we are **not** explicitly given $\vec{x}$. So let us define our state estimate with a tilde $\tilde{\vec{x}}$ and the resulting output of the state estimates as $\tilde{y}$. As visualization of this is given in figure 4.1 below.

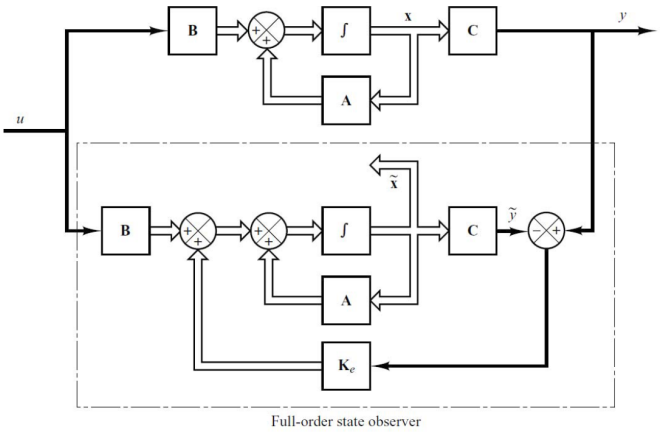

Figure 4.1: Estimation Block Diagram

From this, we have the following mathematical models:

Observer Dynamics	System Dynamics
$\dot{\tilde{x}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e(y - \mathbf{C}\tilde{\mathbf{x}})$	$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$
$\dot{\tilde{x}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\,\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e y$	$y = \mathbf{C}\mathbf{x}$

Table 4.1: Observer vs System Dynamics

where $\mathbf{K}_e$ is the observer gain used as a weighting matrix to the *observation error* $(y - \tilde{y})$. We then again define the error as $\vec{e} = \vec{x} - \tilde{\vec{x}}$. As such, we can take the difference between the observer and system dynamics equations to get:

$$\dot{\vec{e}}(t) = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\vec{e}(t) \tag{4.39}$$

This looks similar to what we have seen earlier, but not exactly! As such, we can take the conjugate transpose of the entire system to get:

$$\dot{\vec{e}}^{\dagger}(t) = \vec{e}^{\dagger}(t)(\mathbf{A}^{\dagger} - \mathbf{C}^{\dagger}\mathbf{K}_e^{\dagger}) \tag{4.40}$$

This again allows us to use the known pole-placement method to determine the required values of $\mathbf{K}_e$ for specified eigenvalues of the observer. Notice that the matrix $\mathbf{A}^{\dagger} - \mathbf{C}^{\dagger}\mathbf{K}_e^{\dagger}$ above is incidentally the state observability matrix of the original system! Hence, in order to design a full state observer for a system, it must be completely observable. Further recalling the concept of duality, we have that the above generic system is the dual system of the original system. To solve for the associated gains, we can use anyone one of the 3 regulator methods once more. The **Ackermann's formula** for this system would be:

$$\boxed{\vec{K}_e = \phi(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}} \tag{4.41}$$

where $\phi(\mathbf{A})$ here is derived from the observer's desired characteristic equation.

§4.3.2 State Feedback Control with Observers

In the Pole-Placement design process, the actual states $\vec{x}(t)$ were assumed to be available. However, $\vec{x}(t)$ may not be measurable in practice, so an observer was implemented and used the observed state for feedback. Design is now a 2 stage process:

1. Determine Feedback Gain K to yield desired (controller) closed-loop performance.

$$\vec{K} = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}^{-1} \phi(\mathbf{A}) \tag{4.42}$$

2. Determine Observer Gain K_e to yield desired (observer) closed-loop performance.

$$\vec{K}_e = \phi(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{4.43}$$

For this to be possible, the system **needs to be completely state controllable and completely observable**.

> **Note**: In order to have a good controller system, we require a good observer system as well since the controller is designed based on the observed states.

§4.4 Minimum-Order Observer

In practice, it is often the case where you can observe certain states and not others. As such, we can partition the state vector into 2 parts:

1. x_a: The measurable portion of the state vector.
2. $\vec{x}_b$: The measurable portion of the state vector.

For this class, we will take x_a as just one entry (scalar) and $\vec{x}_b$ to be an $(n-1)$ vector. The partitioned system can be represented in state-space now as:

$$\begin{bmatrix} \dot{x}_a \\ \dot{\mathbf{x}}_b \end{bmatrix} = \begin{bmatrix} A_{aa} & \mathbf{A}_{ab} \\ \mathbf{A}_{ba} & \mathbf{A}_{bb} \end{bmatrix} \begin{bmatrix} x_a \\ \mathbf{x}_b \end{bmatrix} + \begin{bmatrix} B_a \\ \mathbf{B}_b \end{bmatrix} u \tag{4.44}$$

$$y = \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_a \\ \mathbf{x}_b \end{bmatrix} \tag{4.45}$$

$$\Rightarrow \quad \boxed{\dot{x}_a - A_{aa}x_a - B_a u = \mathbf{A}_{ab}\mathbf{x}_b} \tag{4.46}$$

where the terms on the LHS of the boxed equation above are measurable but those on the RHS are not. Comparing the full-order and minimum order observer systems, we have them written as:

Full-order observer	**Minimum-order observer**
$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$	$\dot{\mathbf{x}}_b = \mathbf{A}_{bb}\mathbf{x}_b + \mathbf{A}_{ba}x_a + \mathbf{B}_b u$
$y = \mathbf{C}\mathbf{x}$	$\dot{x}_a - A_{aa}x_a - B_a u = \mathbf{A}_{ab}\mathbf{x}_b$

Table 4.2: Full-Order vs Minimum-Order Observer

Again, similar to the design of the full-order state observer, the minimum-order system needs to be completely observable, for which the *minimum-order observability matrix* is given by:

$$\boxed{\mathbf{O}_{BM} = \begin{bmatrix} \mathbf{A}_{ab}^{\dagger} & \mathbf{A}_{bb}^{\dagger}\mathbf{A}_{ab}^{\dagger} & \cdots & \left(\mathbf{A}_{bb}^{\dagger}\right)^{n-2}\mathbf{A}_{ab}^{\dagger} \end{bmatrix}} \tag{4.47}$$

for which this matrix **must** be of rank $n-1$ in order to do pole-placement to arbitrary positions in the s-plane.

Chapter 5

Linear-Quadratic Regulators

Consider the following scenario. Imagine you're training for the long distance run of a physical fitness test. What is the maximal speed you should adopt in order to achieve the best possible time, considering fatigue, temperature and other relevant parameters? More generally, this can be scoped out as a problem having given some constraints, we ask what the best course of action one could take to achieve some optimal result. This can in fact be thought of as a rudimentary artificially intelligent system. For controllers, we also want to keep in mind the balancing of performance and energy consumption.

§5.1 Optimal Controllers

the pole-placement technique was great in determining the required controller gains, however it requires an iterative approach. The quadratic optimal control approach provides a much more systematic means of computing the feedback control gain. To do this, we define a *cost function* (*performance index* which we want to **minimize**). For the general linear system:

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}u \tag{5.1}$$

and a desired optimal control vector:

$$u(t) = -\mathbf{K}\vec{x} \tag{5.2}$$

we define the *performance index* as:

$$J = \int_0^\infty \left(\vec{x}^\dagger \mathbf{Q}\vec{x} + \vec{x}^\dagger \mathbf{R}\vec{x}\right) dt \tag{5.3}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are *positive-definite* (real symmetric) matrices that we define (for our application). These matrices determine the relative importance of state vector error and expenditure of energy due to the required control input u respectively. The solution, if it can be computed, is the *optimal control law*. Let's dive in to solving this equation. First, we insert the control law into our system:

$$\dot{\vec{x}} = (\mathbf{A} - \mathbf{BK})\vec{x} \tag{5.4}$$

Assuming that $\mathbf{A} - \mathbf{BK}$ is stable, we substitute this into our cost function:

$$\begin{aligned} J &= \int_0^\infty \left(\vec{x}^\dagger \mathbf{Q}\vec{x} + \vec{x}^\dagger \mathbf{K}^\dagger \mathbf{R}\mathbf{K}\vec{x}\right) dt \\ &= \int_0^\infty \vec{x}^\dagger (\mathbf{Q} + \mathbf{K}^\dagger \mathbf{R}\mathbf{K})\vec{x} dt \\ &\equiv \int_0^\infty -\frac{d}{dt}\vec{x}^\dagger(\mathbf{P})\vec{x} dt \end{aligned} \tag{5.5}$$

The definition above allows us to solve get a nice form for J being:

$$\begin{aligned} J &= \int_0^\infty \mathbf{x}^\dagger (\mathbf{Q} + \mathbf{K} * \mathbf{R}\mathbf{K})\mathbf{x} dt \\ &= \int_0^\infty -\frac{d}{dt}(\mathbf{x}^\dagger \mathbf{P}\mathbf{x}) dt \\ &= -\mathbf{x}^\dagger(\infty)\mathbf{P}\mathbf{x}(\infty) + \mathbf{x}^\dagger(0)\mathbf{P}\mathbf{x}(0) \\ &= \mathbf{x}^\dagger(0)\mathbf{P}\mathbf{x}(0) \end{aligned} \tag{5.6}$$

since $\vec{x}(\infty) \to 0$. To also ensure we know the explicit form of $\mathbf{P}$, we compare the definition and the explicit form:

$$\begin{aligned} J &= -\int_0^\infty (\dot{\mathbf{x}}^\dagger \mathbf{P}\mathbf{x} - \mathbf{x}^\dagger \mathbf{P}\dot{\mathbf{x}}) dt \\ &= -\int_0^\infty \{[(\mathbf{A} - \mathbf{BK})\mathbf{x}]^\dagger \mathbf{P}\mathbf{x} - \mathbf{x}^\dagger \mathbf{P}(\mathbf{A} - \mathbf{BK})\mathbf{x}\} dt \\ &= -\int_0^\infty \{\mathbf{x}^\dagger [(\mathbf{A} - \mathbf{BK}) * \mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{BK})]\mathbf{x}\} dt \\ \Rightarrow \qquad -\left(\mathbf{Q} + \mathbf{K}^\dagger \mathbf{R}\mathbf{K}\right) &= (\mathbf{A} - \mathbf{BK})^\dagger \mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{BK}) \end{aligned} \tag{5.7}$$

Since we have asserted that $\mathbf{R}$ is positive-definite, we can write it as:

$$\mathbf{R} = \mathbf{T}^\dagger \mathbf{T} \tag{5.8}$$

Using this and after much manipulation, we get that the integrand (without contraction with the $\vec{x}$ vectors) to be:

$$\mathbf{A}^\dagger \mathbf{P} + \mathbf{P}\mathbf{A} + \left[\mathbf{TK} - \left(\mathbf{T}^\dagger\right)^{-1} \mathbf{B}^\dagger \mathbf{P}\right]^\dagger \left[\mathbf{TK} - \left(\mathbf{T}^*\right)^{-1} \mathbf{B}^\dagger \mathbf{P}\right] - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\dagger \mathbf{P} + \mathbf{Q} = 0 \tag{5.9}$$

Now to minimize the performance index J with respect to $\mathbf{K}$, we just need to minimize the terms that depend on $\mathbf{K}$ in the integral:

$$\Rightarrow \quad \min_{\mathbf{K}}\{J\} = \min_{\mathbf{K}} \left\{ \int_0^\infty \mathbf{x} * \left[\mathbf{TK} - \left(\mathbf{T}^*\right)^{-1} \mathbf{B} * \mathbf{P}\right] * \left[\mathbf{TK} - \left(\mathbf{T}^*\right)^{-1} \mathbf{B} * \mathbf{P}\right] \mathbf{x} dt \right\} \tag{5.10}$$

$$\Rightarrow \quad \boxed{\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^\dagger \mathbf{P}} \tag{5.11}$$

Substituting this back into the integrand above, we get that the matrix $\mathbf{P}$ **must** satisfy the following expression:

$$\boxed{\mathbf{A}^\dagger\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\dagger\mathbf{P} + \mathbf{Q} = \mathbf{0}} \tag{5.12}$$

This is known as the *continuous-time algebraic Riccati equation (CARE).*

Example

Consider a RLC circuit where the capacitor and inductor are in parallel, both in series to the resistor. we want to design an optimal controller that regulates the input current u to minimize the energy wasted in the resistor and also brings the system to rest. We will do this in 3 parts.

1. *Find a relationship between u, x_1, x_2, R, L and C.*
2. *Obtain the state space representation of the system.*
3. *Setting all electrical parameters to 1, $R = L = C = 1$, design the optimal feedback gain matrix $\mathbf{K}$ such that the following performance index is minimized.*

$$J = \int_0^\infty \left(x_1^2 + x_2^2 + u^2\right) dt, \quad \mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{R} = r = 1 \tag{5.13}$$

To solve this problem, we first define x_1 to be the current though the inductor and x_2 to be the voltage across and the capacitor and inductor. We also recall that:

$$V_L = L\frac{dI_L}{dt}, \quad I_C = C\frac{dV_C}{dt} \tag{5.14}$$

As such, we have that:

$$x_2 = L\frac{dx_1}{dt} = L\dot{x}_1 \tag{5.15}$$

$$\dot{x}_2 = \frac{I_c}{C} = \frac{u - x_1}{C} \tag{5.16}$$

As such, this allows us to put this into state-space representation as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1/L \\ -1/C & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/C \end{bmatrix} u \tag{5.17}$$

$$\Rightarrow \quad \mathbf{A} = \begin{bmatrix} 0 & 1/L \\ -1/C & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1/C \end{bmatrix} \tag{5.18}$$

To solve this, we first use the CARE to solve for $\mathbf{P}$ with $R = L = C = 1$:

$$\mathbf{A}^\dagger\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\dagger\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{5.19}$$

$$\Rightarrow \quad \mathbf{P} = \begin{bmatrix} \sqrt{8} - 2 & \sqrt{2} - 1 \\ \sqrt{2} - 1 & \sqrt{2\sqrt{2} - 1} \end{bmatrix} \tag{5.20}$$

With this, we solve for the optimal gain matrix:

$$\begin{aligned}\mathbf{K} &= \mathbf{R}^{-1}\mathbf{B}^{\dagger}\mathbf{P} \\ &= 1 \cdot \begin{bmatrix} 0 & 1/C \end{bmatrix} \begin{bmatrix} \sqrt{8}-2 & \sqrt{2}-1 \\ \sqrt{2}-1 & \sqrt{2\sqrt{2}-1} \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{2}-1 & \sqrt{2\sqrt{2}-1} \end{bmatrix}\end{aligned} \tag{5.21}$$

It is good to note that the performance index can also be presented in terms of the outputs y instead of states $\vec{x}$. Since we have the familiar form:

$$y = \mathbf{C}\vec{x} \tag{5.22}$$

We have that:

$$\begin{aligned}J &= \int_0^{\infty} (\mathbf{y}^{\dagger}\mathbf{Q}\mathbf{y} + \mathbf{u}^{\dagger}\mathbf{R}\mathbf{u}) dt \\ &= \int_0^{\infty} \left(\mathbf{x}^{\dagger}\mathbf{C}^{\dagger}\mathbf{Q}\mathbf{C}\mathbf{x} + \mathbf{u}^{\dagger}\mathbf{R}\mathbf{u}\right) dt \\ &= \int_0^{\infty} \left(\mathbf{x}^{\dagger}\tilde{\mathbf{Q}}\mathbf{x} + \mathbf{u}^{\dagger}\mathbf{R}\mathbf{u}\right) dt\end{aligned} \tag{5.23}$$

which is an equivalent representation of the performance index. Also, in the cases where $\mathbf{A} - \mathbf{B}\mathbf{K}$ cannot be made stable, there is no positive-definite $\mathbf{P}$ that can satisfy the Riccati equation (no solution).

Chapter 6

Discrete-Time and Digital Signals

Thus far, we have studied controllers under the premise that all components operate under analog regimes (continous-time). This allowed for the application of continuous-time controllers to be integrated seamlessly with real world physical systems, those also being continuous-time (described by differential equations). Following from this traditional control paradigm, there is a current trend towards digital control of dynamic systems due to the availability of low-cost digital computers and many advantages of working with digital signals (e.g. noise attenuation, error checking, low-power, etc...). Because of this, complex and more intelligent controllers can be more easily implemented with digital components. They are also more flexible, allowing adjustments on the fly.

§6.1 Digital Control

Along with all the pros of using digital signals (as opposed to analog signals), there are of course drawbacks. These largely manifest in 3 major ways:

1. Integration and differentiation operations are only done approximately due to finite sampling. These approximations are known as *difference equations*.
2. Sampling and data hold also introduces an inherent *delay* to the system, which has serious implication in closed-loop control.
3. The amplitude of digital signals suffer from quantization error due to the finite bits used to represent data (e.g. 10 bit D/A converter has 1024 values).

For naturally continuous-time systems, we can perform a mapping into discrete-time space known as an *analog-to0digital conversion*.

Definition 6.1.1. Analog-to-Digital Conversion (ADC): *The process of converting a continuous physical quantity to a digital number representing the quantity's amplitude. It consists of (in chronological order):*

1. ***Discretization/Sampling****: Segmenting the input signals into finite time intervals.*
2. ***Amplitude Quantization****: Assigning permitted output values within some specification to continuous input signals.*
3. ***Encoding****: Embedding the input signal into quantized output states (normally a binary code).*

By extension, the Digital-to-Analog conversion (DAC) is the converse process (inverse function) which converts a digital number to a continuous-time signal. DACs consist of:

1. **Decoder**
2. **Transform** from an analog signal into a digital signal (numerically coded data).

§6.1.1 Discretization/Sampling Process

The sampling of continuous-time signal replaces the original continuous-time signal by a sequence of values at discrete time intervals/points. As terminology goes, a signal whose independent variable t is discrete, is called a discrete-time signal. If the signal is further quantized and encoded, it is known as a digital signal. The term '*discretization*' and '*sampling*' are used interchangeably though the former is frequently used in MIMO Systems (state-space). There are several types of sampling operations, namely:

1. **Periodic Sampling**: Sampling instants are equally spaced.
2. **Multiple-Order Sampling**: Pattern of $t_k = kT$ for some time interval T and integer k is repeated periodically.
3. **Multiple-Rate Sampling**: For control systems with multiple loops, there might be different sampling rates/periods in each subsystem.
4. **Random Sampling**: Sampling interval is random.

§6.1.2 Quantization

Computers operate on binary number systems (bits: {ON, OFF}, $\{1, 0\}$, $\mathbb{Z}$ mod 2), where the fundamental constituent of such systems are known as *bits*. A binary system with n bits can represent $2n$ discrete amplitude levels. As such, we define the *quantization level* Q as the range between 2 adjacent *decision points*:

$$\boxed{Q = \frac{\text{FSR}}{2^n - 1} = \text{LSB}} \tag{6.1}$$

where FSR is the full-scale range and in most cases, the least significant bit (LSB) is equal to the quantization level. We also define a quantization error denoted $e(t)$, which is given by:

$$e(t) = x(t) - y(t) \tag{6.2}$$

where $x(t)$ is the analog input signal and $y(t)$ is the discretized output signal. The magnitude of the error must always be bounded by the following inequality:

$$0 \leq |e(t)| \leq \frac{1}{2}Q \tag{6.3}$$

Note: By convention, the term quantization implies a discretization of signal amplitudes, whereas discretization/sampling is the discretization of time.

§6.1.3 Encoding

With the quantized output, the last step is to relate it to a numerical code that can be processed easily in digital systems (modern computers). A very common method is using hexadecimals (base 16), due to its close relationship to the binary system (base 2) although we humans use decimal systems (base 10). For illustration, let us consider a 2-bit system used to represent 0 – 3 (voltage). Total number of discrete amplitude levels the system can represent would be 2^2, which have the explicit encodings $\{00, 01, 10, 11\}$. As such, the quantization level and maximum quantization error would be:

$$Q = \frac{3 - 0}{2^2 - 1} = 1\text{V} \tag{6.4}$$

$$|e(t)| = \frac{Q}{2} = 0.5\text{V} \tag{6.5}$$

Today's ADCs are usually at least 10 bits (1024 levels), where good ones are generally 16 bit (65536 levels) and professional standards dictate 24 bits (16777216 levels).

§6.2 Difference Equations

As mentioned, discrete-time systems adopt difference equations to describe their dynamics as opposed to differential equations. These rely on approximations, for which the simplest of these is the *Euler's approximation* to derivatives:

$$\frac{d}{dt}x(t) \cong \frac{x(k+1) - x(k)}{T} \tag{6.6}$$

where T is the *sampling period* and k are the time-step indices. With this and other viable approximations to derivatives, you arrive with a set of equations that can be solved by a digital computer. These equations are called *difference equations* and are solved iteratively with time steps of T. In principle, the difference equation is evaluated initially at $k = 0$ and then sequentially at $k = 1, 2, 3, ...$ There is no requirement to store all values in memory and for first order difference equations since you only need variables from the current and past values (Markovian). A difference equation can be expressed generally as:

$$y(k) = f\left[u(0), \cdots, u(k); y(0), \cdots, y(k-1)\right] \tag{6.7}$$

For which we define a *linear constant-coefficient difference equation* (CCDE) as:

$$\begin{aligned} y(k) = &- a_1 y(k-1) - a_2 y(k-2) - \cdots - a_n y(k-n) \\ &+ b_0 u(k) + b_1 u(k-1) + \cdots + b_m u(k-m) \end{aligned} \tag{6.8}$$

We would now like to find a elegant way to approach solving such difference equations.

Note: The stability of a CCDE is ascertained if all poles lie within the unit circle about the origin.

§6.3 Z-Transforms

In discrete-time systems, we want to find some method to solving difference equations that is analogous to the Laplace transform for differential equations. There indeed exists such a technique, known as the *z-transform*. The z-transform is defined as:

$$\boxed{\mathcal{Z}[x(t)] = X(z) = \sum_{k=0}^{\infty} x(kT) z^{-k} = \sum_{k=0}^{\infty} x(k) z^{-k}} \tag{6.9}$$

Where z is a complex variable, $\mathcal{Z}$ is the z-transform operator and $X(z)$ is the z-transform of $x(t)$. Similar to the Laplace transform, the z-transform satisfies some nice properties that we can make use of in control analysis. Some of the commonly used ones are listed below.

z-Transform Properties:

1. **Linearity**:

$$\mathcal{Z}\left[a_1 x_1(t) + a_2 x_2(t)\right] = a_1 X_1(z) + a_2 X_2(z) \tag{6.10}$$

2. **Multiplicative Scaling**:

$$\mathcal{Z}\left[a^k x(k)\right] = X\left(a^{-1} z\right) \tag{6.11}$$

3. **Time-Shift**:

$$\mathcal{Z}[x(t - nT)] = z^{-n} X(z) \tag{6.12}$$

$$\mathcal{Z}[x(t + nT)] = z^n \left[X(z) - \sum_{k=0}^{n-1} x(kT) z^{-k} \right] \tag{6.13}$$

4. **Initial Value Theorem**:

Theorem 6.3.1. *If $x(t)$ has the z transform $X(z)$ and if the limit $\lim_{z\to\infty} X(z)$ exists, then the initial value $x(0)$ is given by:*

$$x(0) = \lim_{z\to\infty} X(z) \tag{6.14}$$

5. **Final Value Theorem**:

Theorem 6.3.2. *Suppose that $x(k)$, where $x(k) = 0$ for $k < 0$, has the z transform $X(z)$ and all the poles of $X(z)$ lie inside the unit circle, with exception of the $z = 1$ pole, the final value of $x(k)$ as $k \to \infty$ is:*

$$\lim_{k\to\infty} x(k) = \lim_{z\to 1} \left[\left(1 - z^{-1}\right) X(z)\right] \tag{6.15}$$

Just as the Laplace transforms and inverse Laplace transforms play crucial roles in understanding and designing continuous-time control systems, so does the z and inverse z-transforms to discrete-time systems. The inverse z-transform of $X(z)$ will yield the corresponding time sequence $x(k)$ but does **not** infer information of $x(t)$ (that information is not encoded into the z-transform).

Note: The inverse z transform of $X(z)$ yields a unique $x(k)$ but **not** a unique $x(t)$. The inverse z-transform yields a time sequence that specifies values of $x(t)$ **only** at discrete instances of time ($t = 0, T, 2T, ...$) but says nothing about the values of $x(t)$ at all other intermediate times.

After taking the z-transform, we are able to contruct a transfer function just as we did for continuous-time systems and Laplace transforms. These look something like:

$$\begin{aligned} G(z) &= \frac{Y(z)}{X(z)} \\ &= \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_m}{z^n + a_1 z^{n-1} + \cdots + a_n} \\ &= \frac{b_0 \left(z - z_1\right)\left(z - z_2\right)\cdots\left(z - z_m\right)}{\left(z - p_1\right)\left(z - p_2\right)\cdots\left(z - p_n\right)} \end{aligned} \tag{6.16}$$

where p_j are the poles and z_j are the zeros of $G(z)$. However, in control engineering and signal processing, it is usually convention to write the transfer function as ratios of polynomials in z^{-1}:

$$\boxed{G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 z^{-(n-m)} + b_1 z^{-(n-m+1)} + \cdots + b_m z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}} \tag{6.17}$$

where z^{-1} can be interpreted as the *unit-delay operator* (time-shift by unit interval).

Note: We use the first form (polynomials in z) to find the poles and zeros, whereas the second form polynomials in z^{-1} is used for analysis.

In practice, there are 4 main methods to compute the inverse z-transform from a given transfer function in the z-domain.

1. **Direct Division Method**
2. **Computational Method**
3. **Partial Fraction Method**

4. **Inversion Integral Method**

These will be presented and discussed below.

§6.3.1 Direct Division Method

For this method, our goal is to derive the coefficients of the sum terms that defines the z-transform:

$$X(z) = \sum_{k=0}^{\infty} x(kT)z^{-k} = x(0) + x(T)z^{-1} + x(2T)z^{-2} + \cdots + x(kT)z^{-k} + \cdots \tag{6.18}$$

To do this, we do an explicit long division of the transfer function (second form), which will be an infinite series of terms in powers of z^{-1}. The respective coefficients for z^{-k} are then exactly the $x(kT)$ values:

$$\begin{array}{rl} & x(0) + x(T)z^{-1} + x(2T)z^{-2} + \cdots + x(kT)z^{-k} + \cdots \\ X(z) \quad) & \overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad} \\ & Y(z) \end{array} \tag{6.19}$$

Note that this method does **not** produce a closed-form expression for the inverse, hence would normally only be used when:

- finding only the first few terms of $x(k)$.
- $x(k)$ only has values for the first few k's.
- $x(k)$ has a pattern that can be understood with the first few k values.

§6.3.2 Computational Method

This method is applied to the z-transform transfer function of the system and is generally used when the degree of the polynomial in the numerator is greater than or equal to that of the denominator. This method aims to study the **impulse-response** of a system and as such, we get that $X(z) = 1 \Rightarrow G(z) = Y(z)$. The steps to perform the computational method are as follows:

1. Rearrange the transfer function such that it is written in the $G(z)/z$ form (i.e. divide RHS and LHS by z).
2. Decompose $G(z)/z$ into partial fractions multiply both sides again by z (i.e. get the partial fractions form of $G(z)$).

§6.3.3 Partial Fraction Method

This method is heavily reliant on the use of the inverse z-transform table. We start with:

$$X(z) = \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_{m-1} z + b_m}{(z - p_1)(z - p_2) \cdots (z - p_n)} \tag{6.20}$$

For which we can factorize this to give the partial fraction form:

$$\frac{X(z)}{z} = \sum_{j=1}^{m} \frac{c_j}{(z-p_1)^j} + \frac{a_2}{(z-p_2)} + \cdots + \frac{a_n}{(z-p_n)} \tag{6.21}$$

$$\Rightarrow \quad \boxed{c_j = \left\{ \frac{d^{j-1}}{dz^{j-1}} \left[(z-p_1)^2 \frac{X(z)}{z} \right] \right\}_{z=p_1}, \quad a_i = \left[(z-p_i) \frac{X(z)}{z} \right]_{z=p_i}} \tag{6.22}$$

which is general to the case of repeated poles. Notice the $1/z$ reciprocal term attached to $X(z)$ for mathematical convenience. From here, we just apply the inverse z-transform identities found in the reference table to each term in the expansion.

§6.3.4 Inversion Integral Method

This method utilizes the identity:

$$\begin{aligned} \mathcal{Z}^{-1}[X(z)] &= x(k) \\ &= \frac{1}{2\pi j} \int X(z) z^{k-1} dz \\ &= \sum_{i=1}^{m} \left[\text{ residue of } X(z)z^{k-1} \text{ at pole } z = z_i \text{ of } X(z)z^{k-1} \right] \\ &= K_1 + K_2 + \ldots + K_m \end{aligned} \tag{6.23}$$

For which each K_j term is given by:

$$\boxed{K = \frac{1}{(q-1)!} \lim_{z \to z_j} \frac{d^{q-1}}{dz_j^{q-1}} \left[(z-z_j)^q X(z) z^{k-1} \right]} \tag{6.24}$$

where q is the order (degeneracy) of the pole z_j. As such, when $q = 1$ we get the simple form:

$$K = \lim_{z \to z_i} \left[(z-z_i) X(z) z^{k-1} \right] \tag{6.25}$$

Note: Be wary of the poles at the origin. When deriving the expressions for $X(z)z^{k-1}$, it is important to check if the poles of that expression changes due to values of k. This is particularly important when $X(z)$ has poles or zeros at the origin and this will affect the expression of $X(z)z^{k-1}$ and increase the number of residues needed to construct the expression for $x(kT)$.

Note: It is **not** advisable to use the inversion integral method when there are **complex** poles.

§6.4 Solving Difference Equations

Difference equations can be solved easily using a digital computer since it is already in an iterative form. However, closed-form expressions for $x(k)$ cannot be obtained by using numerical methods. The z transform methods enable us to obtain closed-form expressions for $x(k)$ which is why it is still valuable to pursue such analytical routes. Consider a linear time-invariant discrete-time system characterized by a (CCDE):

$$x(k) + a_1 x(k-1) + \cdots + a_n x(k-n) = b_0 u(k) + b_1 u(k) + \cdots + b_n u(k-n) \tag{6.26}$$

where $u(k)$ and $x(k)$ are the system input and output. To solve the CCDE, the terms need to first be transformed into the z-domain using z-transforms which we now know how to do!

$$\begin{aligned} \mathcal{Z}[x(k)] &= X(z) \\ \mathcal{Z}[u(k)] &= U(z) \end{aligned} \tag{6.27}$$

From here, we simply follow a process flow similar to that of using Laplace transforms to solve ODEs to find the final closed-form expression for $x(k)$ if possible:

Difference Equation Procedure:

1. Apply the z-transform to the CCDE and utilize the z-transform properties to simplify it.
2. Rearrange the z-domain CCDE such that we make $X(z) = \mathcal{Z}[x(k)]$ the subject (it is often convenient to instead rearrange it into the $X(z)/z$ form as mentioned in the partial fractions method).
3. Apply 1 of the 4 possible methods to compute the inverse z-transform of all terms.

§6.5 Mapping Between the S and Z-Planes

Both s (continuous-time) and z (discrete-time) are complex variables and they share a close relationship. As seen before, transient performance and stability of linear time-invariant continuous-time systems depend on the poles and zeros in the s-plane. Similarly, the transient performance and stability of linear time-invariant discrete-time systems depend on the poles and zeros in the z-plane. A key difference however, is that in discrete systems, the sampling period T comes into play and affects both transient performance and stability. Changing the sampling period modifies the poles and zeros, which causes response behavior to this change. Interestingly enough, we find that s is actually the generator of z which gives the relation:

$$\boxed{z = e^{sT}} \tag{6.28}$$

where T is the sampling period of the system. We can derive this by the technique known as *impules-sampling*.

§6.5.1 Derivation by Impulse-Sampling

Discrete-time control systems may operate partly in discrete time and partly in continuous time. That is to say, some signals are represented as discrete-time functions, $x(k)$, while other signals as continuous functions, $y(t)$. Impulse sampling is the ideal process of obtaining a sampled (discrete) function from a continuous signal where the output of this process is train of impulses, beginning at $t = 0$, ($x(t) = 0$ for $t < 0$), and sampling are regular periods of $t = T$. The strength of each impulse equal the sampled value of the continuous-time signal at the corresponding sampling instant. The impulse-sampled output will be an infinite sequence of impulses and denoted by:

$$x \star (t) = \sum_{k=0}^{\infty} x(kT)\delta(t - kT) \tag{6.29}$$

where this is known as *starred* notation. A visualization of this sampling conversion process is given in figure 6.1 below.

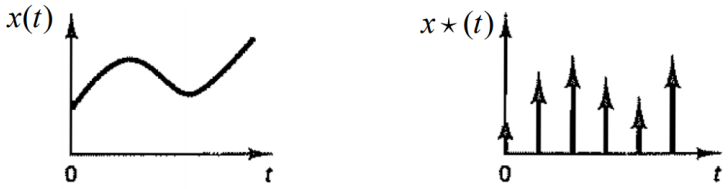

Figure 6.1: Impulse-Sampling Visualization

Since this is a sum over continuous Dirac-delta functions, we can take the Laplace transform of this sampled function to get:

$$\begin{aligned} X \star (s) &= \mathcal{L}\left\{\sum_{k=0}^{\infty} x(kT)\delta(t - kT)\right\} \\ &= \sum_{k=0}^{\infty} x(kT)e^{-kTs} \end{aligned} \tag{6.30}$$

This looks very close to the definition of a z-transform, and if we indeed substitute $z = e^{sT}$ into the expression above, we get:

$$X \star (s) = X(z) = \sum_{k=0}^{\infty} x(kT)z^{-k} \tag{6.31}$$

which is exactly the z-transform! Notice that since we can write s as $s = \sigma + i\omega$, we have:

$$\begin{aligned} z &= e^{(\sigma+i\omega)T} \\ &= e^{\sigma T}e^{i\omega T} \\ &= e^{\sigma T}e^{i\left(\omega+\frac{2\pi}{T}k\right)T} \end{aligned} \tag{6.32}$$

where $k \in \mathbb{Z}$. This means that poles and zeros in the s-plane where frequencies differ in integer multiples of the sampling frequency $2\pi/T$, they are mapped into the **same** location on the z-plane! This means that the imaginary axis in the s-plane corresponds to a unit circle in the z-plane. The left-half s-plane then corresponds to the interior of the z-plane unit circle. Hence, we can think of the left half plane of the s-plane being divided into an infinite number of periodic regions when considering the z correspondence (figure 6.2).

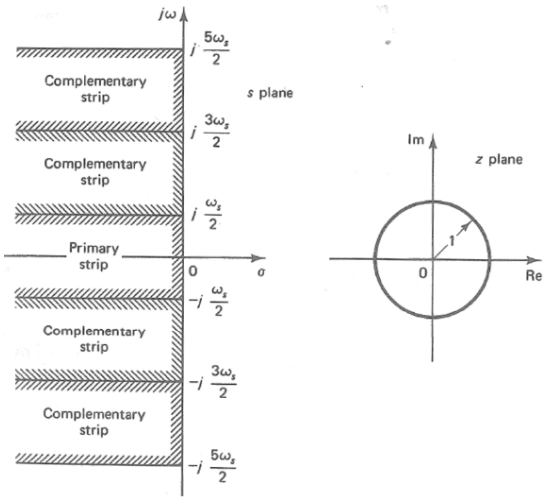

Figure 6.2: s vs z-planes

There is a 'primary' (main) strip that intersects the σ axis, and an infinite number of 'complementary' strips. The movement of a pole in the s and z-planes have the following diagrammatic representations (traversing from $1 \to 2 \to ... \to 5$) as shown in figure 6.3.

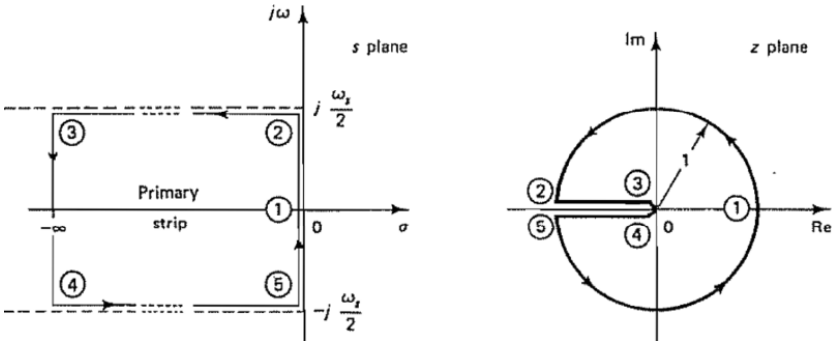

Figure 6.3: Pole movement in the s and z-plane

This implies that in discrete-time (z-domain), a system can actually get arbitrarily fast, since it just corresponds to the pole being arbitrarily close to the origin.

§6.5.2 Data-Holds

Data holding is the process of generating a continuous-time signal $h(t)$ from a discrete-time sequence $x(kT)$ ('inverse' of impulse-sampling). A discrete-time sequence only provides information at specific instants and governed by the sampling period T, so it does **not** provide any information for the signal between sampled pulses. As such, we want to approximate the signal between these T period intervals with a polynomial:

$$h(kT+\tau) = a_n\tau^n + a_{n-1}\tau^{n-1} + \cdots + a_1\tau + x(kT), \quad 0 \leq \tau < T \tag{6.33}$$

The simplest of these polynomials are 0-th order holds (ZOH), which is simply written as:

$$\boxed{h(kT+\tau) = x(kT)}, \quad 0 \leq \tau < T \tag{6.34}$$

In this approximation, our function then looks like a Riemann-sum approximation of the actual continuous-time function. The ZOH approximation function $h(t)$ can then be written as:

$$h(t) = \sum_{k=0}^{\infty} x(kT)[1(t-kT) - 1(t-(k+1)T)] \tag{6.35}$$

For which it's Laplace transform is given by:

$$H(s) = \frac{1-e^{-T_s}}{s} X\star(s) \tag{6.36}$$

$$\Rightarrow \quad G_{ZOH}(s) = \frac{1-e^{-T_s}}{s} \tag{6.37}$$

$G_{zOH}(s)$ gives us an effective continuous approximation to the signals, for which we then feed into our plants $G(s)$, to get the system states in the s-domain:

$$X(s) = G_{ZOH}(s)G(s) \tag{6.38}$$

The eventual goal is still to find $X(z)$, for which we can again use the relation $z = e^{sT}$ to get the following relation:

$$\boxed{X(z) = \left(1-z^{-1}\right)\mathcal{Z}\left[\frac{G(s)}{s}\right]} \tag{6.39}$$

where $\mathcal{Z}[G(s)/s]$ is the z-domain equivalent form of $G(s)/s$ which can be found in the table of Laplace and z-transforms. While the transfer function relates the Laplace Transform of the continuous-time output to that of the continuous-time input, there is an analogous function known as the *pulse transfer function* that relates the z-transform of the sampled output to that of the sampled input. This is written as:

$$G(z) = \frac{Y(z)}{X(z)} \tag{6.40}$$

From the starred notation, we can derive the sampled output function as follows:

$$\begin{aligned} Y\star(s) &= [G(s)X\star(s)]\star \\ &= G\star(s)X\star(s) \end{aligned} \tag{6.41}$$

$$\Rightarrow \quad Y(z) = G(z)X(z) \tag{6.42}$$

where we used a convolution property to do this. A diagram of the sampling process to get the pulse transfer function is given in figure 6.4 below.

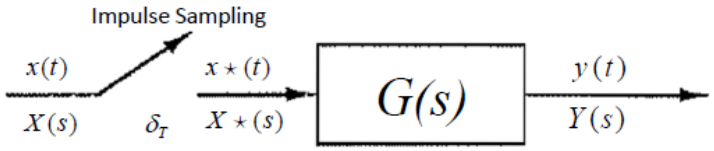

Figure 6.4: Pulse-Transfer Function

§6.5.3 Pulse-Transfer Function of Cascaded Elements

Often times, we are faced with a cascading system of input samplers. As long as there are samplers in every intermediate signal transfer between plants, we can write the effective $Y(s)$ as:

$$Y(s) = [\Pi_j G_j(s)]\, X(s) \tag{6.43}$$

where $G_j(s)$ are the cascading plants. A visual example of a 2 plant cascading system is shown in figure 6.5 below.

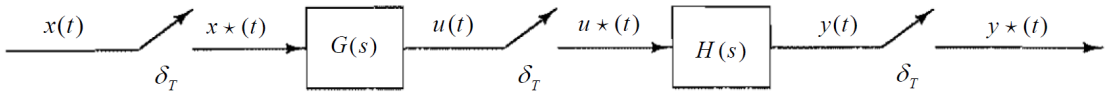

Figure 6.5: Cascading Pulse Transfer Function

§6.6 Z-Plane Stability

For discrete-time systems, the poles of the system must lie within the unit circle in the z-plane. If it is on the unit circle ($z = 1$) (non-degenerate poles), it is marginally stable (also known as *neutral stability*). If there exists repeated conjugate poles on the unit circle or any of the pole are outside the unit circle, the system is **unstable**. To check this for high order systems, we look for an method analogous to the Routh-Hurwitz stability criterion. This actually exists and is known as the *Jury stability criterion* devised in 1981 by Eliahu Ibraham Jury. This is done as follows. Given a (closed-loop) discrete-time system with characteristic equation given by:

$$P(z) = a_0 z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_{n-1} z + a_n = 0 \tag{6.44}$$

with positive a_0, we have to check the following 3 conditions:

1. $|a_n| < a_0$
2. $P(z)|_{z=1} > 0 \quad \Rightarrow \quad \sum_{j=0}^{n} a_j > 0$
3. $P(z)|_{z=-1} = \sum_{j=0}^{n}(-1)^j a_j \begin{cases} > 0 & \text{for } n \text{ even} \\ < 0 & \text{for } n \text{ odd} \end{cases}$

If all are satisfied, we can then proceed to construct the *Jury table*:

Row	z^0	z^1	z^2	z^3	…	z^{n-2}	z^{n-1}	z^n
1	a_n	a_{n-1}	a_{n-2}	a_{n-3}	…	a_2	a_1	a_0
2	a_0	a_1	a_2	a_3	…	a_{n-2}	a_{n-1}	a_n
3	b_{n-1}	b_{n-2}	b_{n-3}	b_{n-4}		b_1	b_0	
4	b_0	b_1	b_2	b_3		b_{n-2}	b_{n-1}	
5	c_{n-2}	c_{n-3}	c_{n-4}	c_{n-5}		c_0		
6	c_0	c_1	c_2	c_3		c_{n-2}		
…								
$2n-5$	p_3	p_2	p_1	p_0				
$2n-4$	p_0	p_1	p_2	p_3				
$2n-3$	q_2	q_1	q_0					

Figure 6.6: Jury Table

After which, we check the final condition:

$$|b_{n-1}| > |b_0|, |c_{n-2}| > |c_0|, \cdots, |q_2| > |q_0| \tag{6.45}$$

The Jury table is constructed with the following equations:

$$\begin{aligned} b_k &= \begin{vmatrix} a_n & a_{n-1-k} \\ a_0 & a_{k+1} \end{vmatrix}, \quad k = 0, 1, 2, \ldots, n-1 \\ c_k &= \begin{vmatrix} b_{n-1} & b_{n-2-k} \\ b_0 & b_{k+1} \end{vmatrix}, \quad k = 0, 1, 2, \ldots, n-2 \\ q_k &= \begin{vmatrix} p_3 & p_{2-k} \\ p_0 & p_{k+1} \end{vmatrix}, \quad k = 0, 1, 2 \end{aligned} \tag{6.46}$$

To check if individual z-domain poles are stable, we simply check if the absolute value $|z|$ is less than 1 (within the unit circle).

§6.7 Closed-Loop Pulse Transfer Functions

To do active control of a possibly unstable system, we need to formulate a closed-loop transfer function. The block diagram for this is given below:

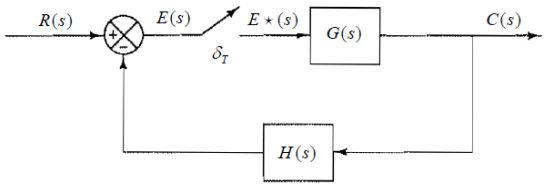

Figure 6.7: Closed-Loop Block Diagram (z-Domain)

From this, we find that the closed-loop transfer function in the z-domain is:

$$\frac{C(z)}{R(z)} = \frac{G(z)}{1 + G(z)H(z)} \tag{6.47}$$

§6.7.1 Z-Domain PID Controller

A very standard and common controller we already have seen before is the PID controller. Recall that the feedback of the PID is given by:

$$m(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right] = m_p(t) + m_i(t) + m_d(t) \tag{6.48}$$

where $e(t)$ is the error term. Performing the z-transform on this, we get the following pulse-transfer function:

$$\boxed{\frac{M(z)}{E(z)} = K_P + \frac{K_I}{1 - z^{-1}} + K_D\left(1 - z^{-1}\right)} \tag{6.49}$$

where we have that $K_P = K - \frac{KT}{2T_i} = K - \frac{K_I}{2}$ is the *proportional gain*, $K_I = \frac{KT}{T_i}$ is the *integral gain* and $K_D = \frac{KT_d}{T}$ is the *derivative gain*.

> **Note**: The proportional gain has a component of the integral gain for discrete-time PID controllers. This is due to the numerical integration approximation.

§6.8 Dead-Beat Response and Control

If the response of a closed-loop control system to a step input exhibits the **minimum** possible settling time (extremely fast), zero steady-state error and no ripples between sampling instances, this response is called a *Dead-Beat Response*. To achieve this, we need to place all poles of the discrete-time system at the **origin**. We also need to ensure that the system is at least type 1 for zero-steady state error. For an n-th order system, the minimum number of steps required would be n, assuming the system is completely state controllable.

Our goal then, is to design our controller $G_D(z)$ such that we have the denominator of our closed-loop pulse transfer function as z^n, where n is the number of poles (order of the system).

$$\begin{aligned} \Rightarrow \quad \frac{C(z)}{R(z)} &= F(z) \\ &= \frac{G_D(z)G(z)}{1 + G_D(z)G(z)} \\ &= \frac{a_0 z^N + a_1 z^{N-1} + \cdots + a_N}{z^N} \\ &= a_0 + a_1 z^{-1} + \cdots + a_N z^{-N} \end{aligned} \tag{6.50}$$

Also note that we can write the controller transfer function as:

$$G_D(z) = \frac{F(z)}{G(z)[1 - F(z)]} \tag{6.51}$$

which indicates to us that to construct a physically realizable controller, The order of the numerator of $G_D(z)$ **must** be equal or lower to that of the denominator (otherwise the controller will require future input to produce current output which cannot be done in practice). Furthermore, if $G(z)$ is expanded into a series in z^{-1}, the lowest-power term of the series expansion of $F(z)$ in z^{-1} must be at least as large as that of $G(z)$.

Note: In practice, it is not feasible to control an unstable pole by adding a term in the numerator of the controller that 'cancels' out the unstable pole. This is so because this would require absolute precision of your controller, even though theoretically possible.

The steps to compute the dead-beat controller is as follows.

1. First, find the effective closed-loop pulse transfer function of the system:

$$G(z) = \mathcal{Z}\left[\frac{1 - e^{-sT}}{s} G_p(s)\right] = (1 - z^{-1})\mathcal{Z}\left[\frac{G(s)}{s}\right] \tag{6.52}$$

2. Determine the form of $F(z)$ by looking at the numerator of $G(z)$.

3. Ensure that this system has zero-steady state error by checking that we can define some function $N(z)$ such that:

$$a - F(z) = (1 - z^{-1})N(z) \tag{6.53}$$

4. Check the conditions of $G_D(z)$, which are that no poles or zeros lie outside the unit circle.

5. Determine the number of samples required to reach steady sate ($c(t \geq T) =$ constant) by looking at the order of the system. For an n-order system, we have that the control output will be of the form:

$$U(z) = \left[\sum_{k=0}^{n-1} b_k (z^{-1})^k\right] + \left[b \sum_{k=n}^{\infty} z^{-k}\right] \tag{6.54}$$

6. Express the control output $U(z)$ in terms of $F(z)$, $R(z)$ and $G(z)$:

$$U(z) = F(z)\frac{R(z)}{G(z)} \tag{6.55}$$

7. Using the relation above, find the corresponding form of $F(z)$ and determine any free-parameters to be solved for.
8. Substitute this form of $F(z)$ back into the expression of $U(z)$.
9. Given that we know the form of $F(z)$, determine the polynomial in z^{-1} form of $F(z)$ and solve for the necessary coefficients a_j ($F(z) = a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}$).
10. From the above result, find the expression for $N(z)$.
11. Substitute all the results back into $G_D(z)$ with the relation:

$$G_D(z) = \frac{F(z)}{G(z)\,(1 - z^{-1})\,N(z)} \tag{6.56}$$

12. We have that the computed closed-loop pulse transfer function is given by:

$$F(z) = \frac{C(z)}{R(z)} = \frac{G_D(z)G(z)}{1 + G_D(z)G(z)} \tag{6.57}$$

Chapter 7

Discrete-Time State-Space Methods

To extend our work from the continuous-time state-space methods we are now so familiar with, we can adopt a state-variable approach for describing difference equations for discrete-time systems. Here, dynamic systems are organized into a set of (only) first-order difference equations (DE). These DEs do ***not*** *need to be linear or time-invariant (compared to Pulse TF approach) and can be easily extended to MIMO systems (Pulse TF is SISO).*

§7.1 Discrete LTI Systems

Given a linear time-invariant system, we can always express it as a system of first order difference equations that take the form:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \end{aligned} \tag{7.1}$$

where we use $\mathbf{G}, \mathbf{H}$ instead of $\mathbf{A}, \mathbf{B}$ to dsitinguish from the continuous-time system. This allows us to formulate our *closed-loop pulse transfer matrix* $\mathbf{F}(z)$ as:

$$\mathbf{F}(z) = \frac{\mathbf{Y}(z)}{\mathbf{U}(z)} = \mathbf{C}(z\mathbf{I} - \mathbf{G})^{-1}\mathbf{H} + \mathbf{D} \tag{7.2}$$

§7.2 Discretization of Continuous-Time State Equations

When deriving state space equations for physical systems, they are normally expressed in continuous-time. However, many applications require that the controller be discrete-time or digital. Hence there is a need to convert continuous-time state space equations into discrete-time state-space equations! Recall that for the following state equation $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, we have the general

solutions given as:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + e^{\mathbf{A}t}\int_0^t e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{7.3}$$

From this, we want to extend this intuition to find a discretized representation of the continuous-time state equation, subject to the following 2 assumptions:

1. Constant sampling period, T.
2. $u(t)$ is sampled and fed through a **zero-order hold** so that components of $u(t)$ are constant over the interval between 2 consecutive sampling instants.

As such, the current goal is that from the continuous time model of the system, we want to find a difference equation of the form $\mathbf{x}((k+1)T) = \mathbf{G}(T)\mathbf{x}(kT) + \mathbf{H}(T)\mathbf{u}(kT)$. If we substitute the sampled input into the continuous-time solution for $t = kT$ and $t = (k+1)T$, this gives us:

$$\mathbf{x}((k+1)T) = e^{\mathbf{A}(k+1)T}\mathbf{x}(0) + e^{\mathbf{A}(k+1)T}\int_0^{(k+1)T} e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{7.4}$$

$$\mathbf{x}(kT) = e^{\mathrm{A}kT}\mathbf{x}(0) + e^{\mathrm{A}kT}\int_0^{kT} e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{7.5}$$

from which we can subtract these from each other and do some manipulation to get:

$$\boxed{\mathbf{x}((k+1)T) = e^{\mathbf{A}T}\mathbf{x}(kT) + e^{\mathbf{A}T}\int_0^T e^{-\mathbf{A}t}\mathbf{B}\mathbf{u}(kT)dt} \tag{7.6}$$

Comparing this with our generic difference equation, we get that:

$$\boxed{\mathbf{G}(T) = e^{\mathbf{A}T}, \quad \mathbf{H}(T) = \left(\int_0^T e^{\mathbf{A}\lambda}d\lambda\right)\mathbf{B}} \tag{7.7}$$

where $\lambda = T - t$. The output of the system is simply given by:

$$y(kT) = \mathbf{C}\mathbf{x}(kT) + \mathbf{D}\mathbf{u}(kT) \tag{7.8}$$

Note: If $\mathbf{A}$ is invertible, $\mathbf{H}$ can be further simplified to:

$$\mathbf{H}(T) = \left(\int_0^T e^{\mathbf{A}\lambda}d\lambda\right)\mathbf{B} = \mathbf{A}^{-1}\left(e^{\mathbf{A}T} - \mathbf{I}\right)\mathbf{B} \tag{7.9}$$

Again extending from continuous-time analysis, we can write the *pulse-transfer matrix* as mentioned earlier:

$$\boxed{\mathbf{F}(z) = \mathbf{C}\left(z\mathbf{I} - \mathbf{G}\right)^{-1}\mathbf{H} + \mathbf{D}} \tag{7.10}$$

Example:

Given the continuous-time system:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} u, \quad F(s) = \frac{Y(s)}{U(s)} = \frac{1}{s(s+2)} \tag{7.11}$$

we want to find the discrete-time state, output and pulse-transfer function. To do this, we first compute the **G** and **H** matrices by using the formulas we have found above:

$$\begin{aligned}\mathbf{G} &= e^{\mathbf{A}T}\\ &= \begin{bmatrix}1 & -0.5\\ 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0\\ 0 & e^{-2T}\end{bmatrix}\begin{bmatrix}1 & 0.5\\ 0 & 1\end{bmatrix}\end{aligned} \tag{7.12}$$

§7.3 Pole-Placement in Discrete-Time Systems

Lucky for us, pole-placement in discrete-time systems is almost completely identical to that done for continuous-time systems. As before, pole-placement techniques assume **all** state variables are measurable and are available for feedback. If a system is completely state controllable, then poles of the closed-loop system may be placed at any desired location by means of state feedback in 3 ways:

1. Using the Controllable Canonical Form as a start.
2. Using direct substitution and comparing with desired c.e. (not recommended for high order systems since it is not easily implemented/scalable on a computer program).
3. Using the Ackermann's Formula.

Remember it is necessary to check for complete state controllability **prior** to employing pole-Placement design. Thankfully, complete state controllability can be done identical to how we did it for continous-time systems!

§7.4 State-Controllability

§7.5 State-Observation and Estimation in Discrete-Time

Just like in continuous-time systems, when designing discrete-time feedback controllers (Pole-Placement), all state variables were assumed to be present. In practice however, not all state variables are available for feedback or they might be noisy. As such, we need to estimate unavailable or unknown state variables. Recall the following definitions:

Definition 7.5.1. Observation: *The process of estimating state variables in the deterministic case (e.g. Luenberger Observers)*

Definition 7.5.2. Estimation: *The process of estimating state variables in the stochastic case (e.g. Kalman Filters).*

If the state observer is able to observe **all** state variables, this is known as *full-order state observation*. Many times this is unnecessary as observation is needed only for the unmeasurable states. An observer that estimates fewer that the dimension of the state vector is then called a *reduced-order state observer*. If the order of the reduced-order state observer is the minimum possible, it is called the *minimum-order state observer*. To check observability, we check the rank of the *observability matrix*:

$$\mathbf{O}_B = \begin{bmatrix} \mathbf{C}^* & \mathbf{G}^*\mathbf{C}^* & \cdots & (\mathbf{G}^*)^{n-1}\mathbf{C}^* \end{bmatrix} \tag{7.13}$$

§7.5.1 Full-Order State Observers

The goal here is for the observer to compensate for inaccuracies in the plant matrices $\mathbf{G}$ and $\mathbf{H}$ as well as lack of knowledge of the initial error $e(0)$. To do this, we define the observation/estimation of the states to be $\vec{\tilde{x}}(kT)$. We consider the error of the output to be:

$$y(kT) - \tilde{y}(kT) = y(kT) - \mathbf{C}\tilde{\vec{x}}(kT) \tag{7.14}$$

For which we can the observer dynamics to be:

$$\begin{aligned} \tilde{\vec{x}}((k+1)T) &= \mathbf{G}\tilde{\vec{x}}(kT) + \mathbf{H}u(kT) + \mathbf{K}_e(y(kT) - \mathbf{C}\tilde{\vec{x}}(kT)) \\ &= (\mathbf{G} - \mathbf{K}_e\mathbf{C})\,\tilde{\vec{x}}(kT) + \mathbf{H}u(kT) + \vec{K}_e y(kT) \end{aligned} \tag{7.15}$$

From here, we consider the state error terms as follows:

$$\begin{aligned} \vec{x}((k+1)T) - \tilde{\vec{x}}((k+1)T) &= \mathbf{G}\vec{x}(kT) + \mathbf{H}u(kT) - \mathbf{G}\tilde{\vec{x}}(kT) - \mathbf{H}u(kT) - \mathbf{K}_e(\mathbf{C}\vec{x}(kT) - \mathbf{C}\tilde{\vec{x}}(kT)) \\ &= (\mathbf{G} - \mathbf{K}_e\mathbf{C})\,(\vec{x}(kT) - \tilde{\vec{x}}(kT)) \end{aligned} \tag{7.16}$$

We then define $\mathbf{e}(kT) = \mathbf{x}(kT) - \tilde{\mathbf{x}}(kT)$, which simplifies our result above to:

$$\boxed{\vec{e}((k+1)T) = (\mathbf{G} - \mathbf{K}_e\mathbf{C})\,\vec{e}(kT)} \tag{7.17}$$

To determine the observer gain $\mathbf{K}_e$, we simply can now apply the pole-placement methods as before.

Chapter 8

Estimation

All sensors used in the physical world are never perfectly precise. There is always noise, both in the system and the measurements taken. Optimal estimation provides the fundamental understanding for deriving the Kalman Filter, one of the most important concepts we will be covering in this course. The first method of approximation we will be considering is the ***least-squares estimation****.*

§8.1 Least-Square Estimation

"*The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum*"

– Karl Friedrich Gauss, 1795

Suppose you have several noisy measurement of a parameter, how would you estimate it optimally? Assume that you have a collection of k measurements and you are trying to estimate a vector x (containing n elements). this can be represented by a system of equations can be obtained.

$$\begin{aligned} y_1 &= H_{11}x_1 + \cdots + H_{1n}x_n + v_1 \\ &\vdots \\ y_k &= H_{k1}x_1 + \cdots + H_{kn}x_n + v_k \end{aligned} \tag{8.1}$$

$$\Rightarrow \quad \vec{y} = \mathbf{H}\vec{x} + \vec{v} \tag{8.2}$$

where v_k are the measurement noises at each measurement. We also define the *residuals* as:

$$\vec{\varepsilon}_y = \vec{y} - \mathbf{H}\hat{\vec{x}} \tag{8.3}$$

where $\hat{\vec{x}}$ are the measured values. From this, we define a *cost function* as follows:

$$\begin{aligned} J &= \varepsilon_{y1}^2 + \cdots + \varepsilon_{yk}^2 \\ &= \vec{\varepsilon}_y^{\mathrm{T}}\vec{\varepsilon}_y \end{aligned} \tag{8.4}$$

Substituting the defintion of our residual, we get:

$$\begin{aligned} J &= (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}})^{\mathrm{T}}(\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) \\ &= \mathbf{y}^{\mathrm{T}}\vec{y} - \hat{\vec{x}}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\vec{y} - \mathbf{y}^{\mathrm{T}}\mathbf{H}\hat{\vec{x}} + \hat{\vec{x}}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{H}\hat{\vec{x}} \end{aligned} \tag{8.5}$$

To minimize this function, we apply the first order necessary condition:

$$\frac{\partial J}{\partial \hat{\vec{x}}} = -\vec{y}^{\mathrm{T}}\mathbf{H} - \vec{y}^{\mathrm{T}}\mathbf{H} + 2\hat{\vec{x}}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{H} = 0 \tag{8.6}$$

$$\Rightarrow \quad \boxed{\mathbf{H}^{\mathrm{T}}\mathbf{H}\hat{\vec{x}} = \mathbf{H}^{\mathrm{T}}\vec{y}} \tag{8.7}$$

§8.1.1 Weighted Least-Square Approximation

Up till now, it was assumed that there was an equal amount of confidence in all the measurements. However, if some measurements are more precise than others, a weighted least squares estimator would definitely yield better results. Mathematically, it can be proven that we should never throw away any measurements no matter how unreliable they are.

For this weight least-squares method, we assume that the measurement of $\mathbf{y}$ is a linear combination of $\mathbf{x}$ with the addition of noise and the variance of noise measurement is different. Further assume that the noise of each measurement has a zero-mean and is always independent. We define the *measurement covariance matrix* as:

$$\mathbf{R} = E\left(\mathbf{v}\mathbf{v}^{\mathrm{T}}\right) \tag{8.8}$$

$$= \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k^2 \end{bmatrix} \tag{8.9}$$

where σ_j^2 are the covariances (weights) we want to associate to each residual. Larger values of the covariances mean that we do **not** want to trust that measurement as much. As such, the goal will now be trying to minimize the **weighted** sum of squares:

$$J = \frac{\varepsilon_{y1}^2}{\sigma_1^2} + \cdots + \frac{\varepsilon_{yk}^2}{\sigma_k^2} = \vec{\varepsilon}_y^{\mathrm{T}}\mathbf{R}^{-1}\vec{\varepsilon}_y \tag{8.10}$$

For which the solution to this minimization problem is given by:

$$\boxed{\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\hat{\vec{x}} = \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\vec{y}} \tag{8.11}$$

While the optimal estimate of a constant can be obtained using multiple measurements of $\hat{\vec{x}}$.

§8.1.2 Recursive Least-Square Estimation

However, the more measurements you take, the more you will need to update the $\mathbf{H}$ matrix (as well as the $\vec{y}$ vector) and recompute the estimate. This makes it computationally taxing. As

such, we then want to develop an estimator that *recursively* computes the weighted least squares estimates, so that after $(k-1)$ measurements, and now we get a new measurement y_k, which we can use to update our estimate rather than reworking the estimation equation. A *linear recursive estimator* can be expressed as:

$$\begin{aligned} \vec{y}_k &= \mathbf{H}_k \vec{x} + \vec{v}_k \\ \hat{\vec{x}}_k &= \hat{\vec{x}}_{k-1} + \mathbf{K}_k \left(\vec{y}_k - \mathbf{H}_k \hat{\vec{x}}_{k-1} \right) \end{aligned} \tag{8.12}$$

where K_k is called the *estimator gain matrix*, and $\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k-1}$ is the *correction term*. With this, we can define the *estimation error mean* as follows:

$$\begin{aligned} E\left(\varepsilon_{\mathrm{x},k}\right) &= E\left(\mathbf{x} - \hat{\mathbf{x}}_k\right) \\ &= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) E\left(\varepsilon_{\mathbf{x},k-1}\right) - \mathbf{K}_k E\left(\mathbf{v}_k\right) \end{aligned} \tag{8.13}$$

To now determine the optimal gain matrix, we define a *estimation covariance error* $\mathbf{P}$:

$$\begin{aligned} \mathbf{P}_k &= E\left(\varepsilon_{\mathrm{x},k} \varepsilon_{\mathrm{x},k}^{\mathrm{T}}\right) \\ &= E\left\{ \left[\left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \varepsilon_{\mathrm{x},k-1} - \mathbf{K}_k \mathbf{v}_k\right] \left[\left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \boldsymbol{\varepsilon}_{\mathbf{x},k-1} - \mathbf{K}_k \mathbf{v}_k\right]^{\mathrm{T}} \right\} \end{aligned} \tag{8.14}$$

We can again do this recursively via the following recursive relation:

$$\boxed{\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_{k-1} \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right)^{\mathrm{T}} + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^{\mathrm{T}}} \tag{8.15}$$

This matrix tells us at any one time how good our estimate values of $\mathbf{x}$ ($\hat{\mathbf{x}}$) are. With this definition, we assert the *optimality criterion* to minimize the sum of the variances of the estimation errors at time k:

$$\min_{\mathbf{K}_k} \{J_k\} = \min_{\mathbf{K}_k} \{\mathrm{Tr}\, \mathbf{P}_k\} \tag{8.16}$$

The solution to this minimization problem using the first order necessary condition is given by:

$$\boxed{\mathbf{K}_k = \mathbf{P}_{k-1} \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1}} \tag{8.17}$$

To better understand what we have done, we can write a succinct algorithm for the recursive techniques applied to least-square estimation as follows.

Least-Square Recursive Algorithm:

1. Initialize the initial estimate of $\hat{\mathbf{x}}_0 = E(\mathbf{x})$ and the initial estimate of the error covariance matrix $\mathbf{P}_0 = E\left[\left(\mathbf{x} - \hat{\mathbf{x}}_0\right)\left(\mathbf{x} - \hat{\mathbf{x}}_0\right)^{\mathrm{T}}\right]$.
2. For $k = 1, 2, 3, ...$:

(a) Obtain the measurement $\mathbf{y}_k$ using:

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x} + \mathbf{v}_k \tag{8.18}$$

where $\mathbf{v}_k$ is a zero-mean noise vector with covariance matrix $\mathbf{R}_k$.

(b) Update the estimate of $\mathbf{x}$ and covariance matrix by computing the optimal gain (8.19), update the estimation (8.20) and updating the error covariance matrix (8.21):

$$\mathbf{K}_k = \mathbf{P}_{k-1}\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_{k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1} \tag{8.19}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k-1}\right) \tag{8.20}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k-1}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^{\mathrm{T}} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^{\mathrm{T}} \tag{8.21}$$

Note: In the initialization step above, if:
1. If no knowledge of $\mathbf{x}$ is available, then assume $\mathbf{P}_0 = \infty\mathbf{I}$.
2. If perfect knowledge of $\mathbf{x}$ is available, then assume $\mathbf{P}_0 = 0$.

§8.2 Kalman Filters

The *Kalman Filter*, developed by Rudolf E. Kálmán in 1960, is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. The Kalman filter extends the idea of optimal estimation to dynamic systems with inherent process noise.

> "*The Kalman Filter in its various forms is clearly established as a fundamental tool for analyzing and solving a broad class of estimation problems.*"

– Leonard McGee & Stanley Schmidt

Again, we start off with the following linear discrete-time system:

$$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k\mathbf{u}_k + \mathbf{w}_k \tag{8.22}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{8.23}$$

We assume that the system and measurement noise process are white, zero-mean, uncorrelated and have known covariances:

$$\begin{aligned} \mathbf{w}_k &\sim N\left(0, \mathbf{Q}_k\right) \\ \mathbf{v}_k &\sim N\left(0, \mathbf{R}_k\right) \end{aligned} \qquad \begin{aligned} E\left[\mathbf{w}_k\mathbf{w}_j^{\mathrm{T}}\right] &= \mathbf{Q}_k\delta_{k-j} \\ E\left[\mathbf{v}_k\mathbf{v}_j^{\mathrm{T}}\right] &= \mathbf{R}_k\delta_{k-j} \\ E\left[\mathbf{v}_k\mathbf{w}_j^{\mathrm{T}}\right] &= 0 \end{aligned} \tag{8.24}$$

The goal of the Kalman Filter is to estimate $\mathbf{x}_k$ based on the knowledge of the system dynamics and the availability of noisy measurements. We now introduce the notion of *a priori* (superscript

-) and *a posteriori* (superscript +) measurements.

$$\textbf{a posteriori}: \quad \hat{\mathbf{x}}_k^+ = E\left[\mathbf{x}_k | \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\right] \tag{8.25}$$

$$\textbf{a priori}: \quad \hat{\mathbf{x}}_k^- = E\left[\mathbf{x}_k | \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{k-1}\right] \tag{8.26}$$

The first measurement of the system is defined at $k = 1$. Hence the initial estimate of $\mathbf{x}_0$ is done **without** any measurement and would just be $\hat{\mathbf{x}}_0^+ = E\left[\mathbf{x}_0\right]$. The update of $\mathbf{x}$ must be paired with the update of the covariance of the estimation error $\mathbf{P}_k$:

$$\mathbf{P}_0^+ = E\left[\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+\right)\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+\right)^{\mathrm{T}}\right] \tag{8.27}$$

$$\begin{aligned} \mathbf{P}_k^- &= E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)^{\mathrm{T}}\right] \\ \mathbf{P}_k^+ &= E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^+\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^+\right)^{\mathrm{T}}\right] \end{aligned} \tag{8.28}$$

The respective *time update equations* for the system state estimates and covariance error estimates are then:

$$\boxed{\hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\mathbf{u}_{k-1}} \tag{8.29}$$

$$\boxed{\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}} \tag{8.30}$$

Following this, once a new measurement is available, we incorporate this new information using the recursive weighted least-square approximation method done previously! This will be modified slightly to suit the a priori and a posteriori paradigms as follows:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1} \tag{8.31}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-\right) \tag{8.32}$$

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^{\mathrm{T}} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^{\mathrm{T}} \tag{8.33}$$

The overall algorithm for clarity of the Kalman filterning process is given by:

Kalman Filter Algorithm:

1. Model and linearize the system such that it fits the form:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{8.34}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{8.35}$$

2. Initialize the system state estimates and covariance error estimates:

$$\hat{\mathbf{x}}_0^+ = E\left[\mathbf{x}_0\right] \tag{8.36}$$

$$\mathbf{P}_0^+ = E\left[\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+\right)\left(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+\right)^{\mathrm{T}}\right] \tag{8.37}$$

3. For $k = 1, 2, 3, ...$:
   (a) obtain the estimated dynamics of the via the Kalman filter update and mea-

surement update equations:

$$\textbf{Time-Update}: \quad \begin{cases} \hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \\ \mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \end{cases} \tag{8.38}$$

$$\textbf{Measurement-Update}: \quad \begin{cases} \hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-\right) \\ \mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^{\mathrm{T}} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^{\mathrm{T}} \\ \qquad = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^- \end{cases} \tag{8.39}$$

$$\textbf{Kalman Filter Gain}: \quad \begin{cases} \mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1} \\ \qquad = \mathbf{P}_k^+\mathbf{H}_k^{\mathrm{T}}\mathbf{R}_k^{-1} \end{cases} \tag{8.40}$$

The a priori and a posteriori portions of the Kalman filter can be combined to give rise to the *discrete-time algebraic Riccati equation*:

$$\begin{aligned} \mathbf{P}_{k+1}^- &= \mathbf{F}_k\left(\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\right)\mathbf{F}_k^T + \mathbf{Q}_k \\ &= \mathbf{F}_k\mathbf{P}_k\mathbf{F}_k^{\mathrm{T}} - \mathbf{F}_k\mathbf{P}_k\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1}\mathbf{H}_k\mathbf{P}_k\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k \end{aligned} \tag{8.41}$$