Chapter 6

Control

6.1 Definitions

In this section, we provide basic definitions from Ogata [3] that will be used in the remainder of this chapter.

**system** A combination of components that act together and perform a certain objective.

**system state** A collection of variables that describe the current configuration of the system.

**control** The operation of measuring the value of a system variable and applying an action to correct or limit its deviation from a desired value.

**controlled variable** The system variable being measured and controlled.

**control signal** The action applied to affect the value of the controlled variable towards a desired reference value.

**disturbance** A signal that tends to adversely affect the value of the output of the system.

**closed-loop/feedback control** An operation that, in the presence of disturbances, tends to maintain a prescribed relationship between the output and a reference input by comparing them and using the difference as a means of control. Fig. 6.1 depicts the block diagram of a typical closed-loop control architecture. The input to the system is a reference signal $r$, whose value is compared with the measured value of the output $y + n$, where $y$ is the actual output and $n$ is measurement noise. This comparison yields an error signal
Figure 6.1: Typical closed-loop architecture in the form of a block diagram.

\[ e = r - (y + n), \] which is fed to the controller to produce a control signal \( u \).

A disturbance \( d \) is then incorporated to produce a modified control signal \( u' \) that interacts with the system.

open-loop control A control action that does not take into consideration the output of the system.

Comparing closed-loop control and open-loop control

The closed-loop design offers the ability to reject unpredictable disturbances and reduce the effect of internal variations in system parameters. Therefore, even with relatively inexpensive and inaccurate system components, it may be possible to achieve the desired system performance. Doing so with an open-loop design is impossible. However, if a closed-loop system compensates for disturbances improperly, there is a risk of exacerbating the problem rather than reducing it. Open-loop design does not have this issue since it does not account for system output. Furthermore, open-loop design offers the advantages of simplicity, ease of maintenance, lower cost, and lower power consumption, as less components are required in general. Open-loop design is thus preferred in cases where disturbances are known in advance.

6.2 The State Space Model

Let \( x \in \mathbb{R}^n \), \( u \in \mathbb{R}^p \), and \( y \in \mathbb{R}^q \) be vectors containing the state variables, control variables, and measured signal of a system respectively. Then the dynamics of the system can be modeled with the following system of differential equations:

\[
\frac{dx}{dt} = f(x, u), \quad y = h(x, u)
\] (6.1)
6.2. THE STATE SPACE MODEL

where \( f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n \) and \( h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q \). A model of this form is called a state space model. The dimension of the state vector is called the order of the system. Since the functions \( f \) and \( h \) do not depend explicitly on time \( t \), the system is time-invariant.

If the functions \( f, h \) are linear in \( x \) and \( u \), the system is linear time-invariant (LTI) and can be represented as

\[
\dot{x} = Ax + Bu, \quad y = Cx + Du,
\]

(6.2)

where \( A, B, C, D \) are constant matrices. \( A \) is called the dynamics matrix, \( B \) is called the control matrix, \( C \) is called the sensor matrix, and \( D \) is called the direct term.

Another way of representing the dynamics of a system is through the \( n \)th-order differential equation

\[
\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = u,
\]

(6.3)

where \( t \) is the independent variable, \( y(t) \) is the dependent (output) variable, and \( u(t) \) is the input. The notation \( \frac{d^k y}{dt^k} \) represents the \( k \)th derivative of \( y \) with respect to \( t \). The system of (6.3) can be converted to the state space form by considering the following definitions:

\[
x = \begin{bmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \end{bmatrix}^T = \begin{bmatrix} \frac{d^{n-1} y}{dt^{n-1}} & \frac{d^{n-2} y}{dt^{n-2}} & \cdots & \frac{dy}{dt} & y \end{bmatrix}^T. \quad (6.4)
\]

Then, from (6.3), we can form the following system of equations:

\[
\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} -a_1 x_1 - \cdots - a_n x_n \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix} + \begin{bmatrix} u \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad y = x_n. \quad (6.5)
\]

Finally, if we assume that the output is a linear combination of the states and the control input, i.e.

\[
y = b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + du,
\]

(6.6)
then the system can be written in the following general form:

\[
\frac{d}{dt} \begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    \vdots \\
    x_n
\end{bmatrix} = \begin{bmatrix}
    -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\
    1 & 0 & \cdots & 0 & 0 \\
    0 & 1 & \cdots & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & 0 & \cdots & 1 & 0
\end{bmatrix} \begin{bmatrix}
    1 \\
    0 \\
    0 \\
    \vdots \\
    0
\end{bmatrix} + \begin{bmatrix}
    0 \\
    0 \\
    0 \\
    \vdots \\
    0
\end{bmatrix} u
\] (6.7)

\[
y = [b_1 \ b_2 \ \cdots \ b_n] x + du. \quad (6.8)
\]

This form is called the **reachable canonical form**.

### 6.2.1 System Response

The general solution to a linear ordinary differential equation of the form

\[
\frac{d^n y}{dt^n} = \sum_{i=0}^{n-1} a_i(x) \frac{d^i y}{dt^i} + r(x)
\] (6.9)

is given by the equation

\[
y = y_c + y_p \quad (6.10)
\]

where \(y_c\) is the solution to the corresponding homogeneous equation (i.e. (6.9) with \(r(x) = 0\)) and \(y_p\) is an additional solution accounting for the term \(r(x)\). In the case of the state space system (6.2), the general solution corresponds to the system response.

**Initial Condition Response**

Consider the homogeneous response corresponding to the system

\[
\frac{dx}{dt} = Ax. \quad (6.11)
\]

For a scalar equation of the same form, i.e.

\[
\frac{dx}{dt} = ax, \quad x, a \in \mathbb{R}, \quad (6.12)
\]

the solution is given by

\[
x(t) = e^{at} x(0). \quad (6.13)
\]

When \(A\) is a matrix, this form of solution can be generalized through the use of the **matrix exponential**. The matrix exponential can be written as an infinite series, i.e.

\[
e^X = I + X + \frac{1}{2} X^2 + \frac{1}{3!} X^3 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!} X^k, \quad (6.14)
\]
6.2. THE STATE SPACE MODEL

where $X \in \mathbb{R}^{n \times n}$ and $I$ is the $n \times n$ identity matrix. Substituting $X$ with $At$ in (6.14), we get

$$e^{At} = I + At + \frac{1}{2}A^2t^2 + \frac{1}{3!}A^3t^3 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!}A^k t^k,$$ (6.15)

which, upon differentiation with respect to $t$, gives

$$\frac{d}{dt}e^{At} = A + A^2t + \frac{1}{2}A^3t^2 + \cdots$$ (6.16)

$$= A \left( I + At + \frac{1}{2}A^2t^2 + \frac{1}{3!}A^3t^3 + \cdots \right)$$ (6.17)

$$= A \sum_{k=0}^{\infty} \frac{1}{k!}A^k t^k = Ae^{At}. \quad (6.18)$$

If we postmultiply (6.18) by $x(0)$, we get

$$\frac{d}{dt} \left( e^{At} x(0) \right) = A \left( e^{At} x(0) \right), \quad (6.19)$$

and thus we conclude that

$$x(t) = e^{At} x(0) \quad (6.20)$$

is a solution to the system of equations of (6.11). Therefore, (6.20) gives the response of the system to an initial condition $x(0)$ under no control input $u$.

**Input/Output Response**

We wish to derive the system response under the application of a control input $u(t)$. This corresponds to the solution of the system

$$\frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du. \quad (6.21)$$

By differentiating both sides and employing the property of (6.18), we can show that the solution is

$$x(t) = e^{At} x(0) + \int_0^t e^{A(t-\tau)} Bu(\tau)d\tau, \quad (6.22)$$

and thus it follows that

$$y(t) = Ce^{At} x(0) + \int_0^t Ce^{A(t-\tau)} Bu(\tau)d\tau + Du(t). \quad (6.23)$$
Further Reading

For more on the state space model, please see Sections 2.2, 5.2, and 5.3 of Åström and Murray [1].

6.3 Stability

One of the central topics of interest in the study of dynamical systems (such as the state-space models we discussed in the previous section) is stability. Intuitively, given a system with dynamics described by a set of differential equations, stability addresses the problem of determining the system’s behavior under small perturbations of its initial conditions.

A solution to a differential equation $\dot{x} = f(x)$ with initial condition $a$, $x(t; a)$ is stable if other solutions that start near $a$ stay close to $x(t; a)$. Formally, the solution $x(t; a)$ is stable if for all $\epsilon > 0$, there exists a $\delta > 0$, such that:

$$
\|b - a\| \leq \delta \implies \|x(t; b) - x(t; a)\| < \epsilon \text{ for all } t > 0.
$$

This type of stability is referred to as stability in the sense of Lyapunov. An interpretation of this definition is that by starting sufficiently close to a certain initial condition, we are guaranteed to stay close to its corresponding solution. If a solution is stable in the sense of Lyapunov but trajectories starting from different initial conditions do not converge, then the solution is called neutrally stable.

An equilibrium point $x_e$ of a dynamical system represents a stationary condition for its dynamics. Formally, $x_e$ is an equilibrium point for a dynamical system $\frac{dx}{dt} = f(x)$ if $f(x_e) = 0$. This implies that a system with an initial condition $x(0) = x_e$ will stay at the equilibrium point, i.e. $x(t) = x_e$ for all $t \geq 0$. In general, a dynamical system can have zero, one, or more equilibrium points. When a solution $x(t; a) = x_e$ is an equilibrium solution, we say that the equilibrium point is stable.

A solution $x(t; a)$ is called asymptotically stable if it is stable in the sense of Lyapunov and also satisfies

$$
x(t; b) \to x(t; a) \text{ as } t \to \infty \text{ for } b \text{ sufficiently close to } a.
$$

This is the case where all nearby trajectories converge to the stable solution as time goes to infinity. Specifically for planar systems, asymptotically stable equilibrium points are commonly referred to as attractors, whereas equilibrium points that are stable but not asymptotically stable are called centers.

A solution is locally stable or locally asymptotically stable if it is stable for all initial conditions $x \in B_r(a)$, where $B_r(a) = \{x : \|x - a\| < r\}$ is a ball of radius $r$ around $a$ and $r > 0$. A system is globally stable if it is stable for all $r > 0$. 

A solution $x(t; a)$ is unstable if it is not stable. Formally, a solution $x(t; a)$ is unstable if given some $\epsilon > 0$, there does not exist a $\delta > 0$ such that if $\|b - a\| < \delta$, then $\|x(t; b) - x(t; a)\| < \epsilon$ for all $t$. An unstable equilibrium point of a planar system is typically referred to as a source, if all trajectories move away from the equilibrium point, or saddle, if some trajectories lead to the equilibrium point and others move away.

### 6.3.1 Stability of Linear Systems

For a linear dynamical system described by dynamics of the form

$$\dot{x} = Ax, \quad x(0) = x_0$$

(6.26)

the stability of the equilibrium at the origin (notice that the origin is always an equilibrium point for such a system) can be examined by looking at the eigenvalues of matrix $A$:

$$\lambda(A) = \{s \in \mathbb{C} : \det(sI - A) = 0\}$$

(6.27)

where $\mathbb{C}$ is the set of complex numbers, the polynomial $p_A = \det(sI - A)$ is the characteristic polynomial of matrix $A$, and its eigenvalues are the roots of the equation $p_A = 0$. Observe that stability is a property of this system since it depends on the matrix $A$.

In general, it can be shown that a linear system $\frac{dx}{dt} = Ax$ is asymptotically stable if and only if all eigenvalues of $A$ all have a strictly negative real part and is unstable if any eigenvalue of $A$ has a strictly positive real part.

### 6.3.2 Lyapunov Stability Analysis

For the general case of a (possibly nonlinear) system

$$\frac{dx}{dt} = f(x), \quad x \in \mathbb{R}^n$$

(6.28)

we are interested in proving that a given solution is stable, asymptotically stable, or unstable. Lyapunov Stability Analysis provides a formal tool for doing so, based on the definition of energy-like functions called the Lyapunov functions.

Consider a nonnegative function $V : \mathbb{R}^n \rightarrow \mathbb{R}^+$ that always decreases along system trajectories; thus its minimum is a locally stable equilibrium point. Therefore, if we find a function with those properties for a given system, we can prove that the system is stable. Such a function is called a Lyapunov function. To characterize the stability of a system in the sense of Lyapunov, we make use of the Lyapunov Stability Theorem. To formulate the theorem, we first need to go over a few definitions:
• A continuous function $V$ is *positive definite* if $V(x) > 0$ for all $x \neq 0$ and $V(0) = 0$.

• A continuous function $V$ is *negative definite* if $V(x) < 0$ for all $x \neq 0$ and $V(0) = 0$.

• A continuous function $V$ is *positive semidefinite* if $V(x) \geq 0$ for all $x$, but $V(x)$ can be zero at points other than just $x = 0$.

**Lyapunov Stability Theorem**

Let $V : \mathbb{R}^n \rightarrow \mathbb{R}^+$ and denote by $\dot{V}$ its time derivative along trajectories of system dynamics, as

$$\dot{V} = \frac{\partial V}{\partial x} dx \frac{dt}{dt} = \frac{\partial V}{\partial x} f(x).$$

Let also $B_r = B_r(0)$ be a ball of radius $r$ around the origin. If there exists $r > 0$ such that $V$ is positive definite and $\dot{V}$ is negative semidefinite for all $x \in B_r$, then $x = 0$ is *locally stable in the sense of Lyapunov*. If $V$ is positive definite and $\dot{V}$ is negative definite in $B_r$, then $x = 0$ is *locally asymptotically stable*.

**Example 6.1** Stability analysis of a mechanical system

Consider the system of Fig. 6.2 with a block of mass $m$ attached to a spring of stiffness $k$. The block is located at position $x = 0$ and is subject to frictional force of coefficient $b$ proportional to the block’s velocity. We can write the motion equation as

$$m\ddot{x} + b\dot{x} + kx = 0$$

and compute the total energy of the system as

$$V = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2.$$  \hspace{1cm} (6.31)

The rate of change of the total energy of the system can then be found by differentiating (6.31) with respect to time, as

$$\dot{V} = m\dot{x}\ddot{x} + kx\dot{x}.$$  \hspace{1cm} (6.32)

Substituting (6.30) for $m\ddot{x}$ in (6.32), we obtain

$$\dot{V} = -b\dot{x}^2,$$  \hspace{1cm} (6.33)

which is always nonpositive since $b > 0$. Therefore, energy always leaves the system unless $\dot{x} = 0$, which implies that the system will release the energy until it comes to rest regardless of initial state. Hence, we have shown that this spring-mass system will eventually come to rest at the equilibrium.
6.4 Feedback Control

In this section, we discuss how to shape the behavior of a system through its feedback, both in the case where the state is directly measurable through state feedback and in the case where we need to estimate the state through output feedback.

6.4.1 Why feedback?

Why should one use feedback? What are the advantages (and disadvantages) of feedback control over other control architectures? In practice, there are many trade-offs (including weight, added complexity, and reliability of sensors), but there are two fundamental benefits of feedback:

- **Modification of dynamics.** Feedback can modify the natural dynamics of a system. For instance, using feedback, one can improve the damping of an underdamped system, or stabilize an unstable operating condition, such as balancing an inverted pendulum. Open-loop or feed-forward approaches cannot do this.

- **Reduction of sensitivity.** Feedback can also reduce sensitivity to external disturbances or to changing parameters in the system itself (the plant). For instance, an automobile with a cruise control that senses its current speed can maintain the set speed in the presence of disturbances such as hills and headwinds. Feedback also compensates for changes in the system itself, such as the weight of a vehicle as the number of passengers changes. If feedback was not available and the system was only able to use a lookup table to select...
CHAPTER 6. CONTROL

the appropriate throttle setting based on a desired speed, the system would not be able to compensate for disturbances or internal changes.

6.4.2 Reachability and Controllability

Informally, the concepts of reachability and controllability examine respectively whether a given state can be reached from the origin and whether a given state can be driven to the origin. Formally, for a system with dynamics

\[ \dot{x}(t) = Ax(t) + Bu(t), \]  

(6.34)
a state \( x_d \neq 0 \) is said to be **reachable** in time \( T \) if there exists a finite time interval \([0, T]\) and an input \( u : [0, T] \to \mathbb{R}^p \) that can drive the system from \( x_0 = 0 \) to \( x_d \) in time \( T \), whereas a state \( x_0 \) is said to be **controllable** in time \( T \) if there exists a finite time interval \([0, T]\) and an input \( u : t \in [0, T] \to \mathbb{R}^p \) that can drive the system from \( x_0 \) to zero in time \( T \).

Let \( \mathcal{R} \) be the set of all reachable points, i.e., the set of all points in \( \mathbb{R}^n \) that can be reached within a finite amount of time, starting from \( x_0 = 0 \). The set \( \mathcal{R} \) is a linear subspace, i.e., if \( x_1, x_2 \in \mathbb{R}^n \) are reachable states, so is any linear combination of them. If the reachable set is the entire state space, i.e., if \( \mathcal{R} = \mathbb{R}^n \), then the system is called (completely) reachable. Likewise, if all states are controllable, then the system is called (completely) controllable. For continuous, linear time-invariant systems, a system is controllable if and only if it is reachable, i.e., controllability implies reachability and vice versa.

Testing for Reachability and Controllability

Consider an **impulse** input defined as

\[ \delta(t) = \lim_{\epsilon \to 0} p_\epsilon(t) \]  

(6.35)
for

\[ p_\epsilon(t) = \begin{cases} 
0 & t < 0 \\
1/\epsilon & 0 \leq t < \epsilon \\
0 & t \geq 0 
\end{cases} \]  

(6.36)
acting on a system with zero initial state. The system response can then be found to be

\[ x_\delta = \int_0^t e^{A(t-\tau)} Bu(\tau)d\tau = e^{At}B. \]  

(6.37)
6.4. FEEDBACK CONTROL

The response to the derivative of an impulse response can be shown to be given by taking the derivative of the impulse response, i.e.,

\[ x_\delta = \frac{dx_\delta}{dt} = Ae^{At}B. \quad (6.38) \]

Taking the higher derivatives of the impulse responses, we can construct the input

\[ u(t) = a_1 \delta(t) + a_2 \ddot{\delta}(t) + \cdots + a_n \delta^{(n-1)}(t), \quad (6.39) \]

which gives the expression for the corresponding response

\[ x(t) = a_1 e^{At}B + a_2 e^{At}B + \cdots + a_n e^{At}B. \quad (6.40) \]

Taking the limit as \( t \) goes to zero, we get the response

\[ \lim_{t \to 0^+} x(t) = a_1 B + a_2 A B + \cdots + a_n A^{n-1} B, \quad (6.41) \]

which is a linear combination of the columns of the matrix

\[ C = \begin{bmatrix} B & AB & \cdots & A^{n-1} B \end{bmatrix}. \quad (6.42) \]

This matrix is called the reachability/controllability matrix. In order to reach an arbitrary point in the state space, the reachability matrix needs to have \( n \) linearly independent columns, or, equivalently, \( C \) needs to be of rank \( n \). This result can be generalized for different (smoother) signals.

6.4.3 Stabilization by State Feedback

Fig. 6.3 depicts the block diagram of a system controlled with state feedback. The controller comprises two elements, \( k_r \) and \( K \), with the former acting on the reference \( r \) and the latter acting on the state \( x \). The input to the system is determined by the control law

\[ u = k_r r - K x \quad (6.43) \]

and the disturbance \( d \). Thus the controller adjusts the system dynamics as

\[ \frac{dx}{dt} = (A - BK)x + Bk_r r. \quad (6.44) \]

The system stability and the transient response of the controlled system are determined by the eigenvalues of the closed-loop state matrix \( A_{cl} = A - BK \). Therefore, when picking \( K \), we manipulate the eigenvalues to achieve desired properties. In particular, to pick \( K \), we typically write down a parametric expression of the characteristic equation of \( A_{cl} \),

\[ p_{A_{cl}}(s) = \det(sI - A_{cl}) = 0, \quad (6.45) \]
and select the entries of $K$ to place its roots (the eigenvalues of $A_{cl}$, also called poles) to be such that the desired performance specifications are achieved.

On the other hand, $k_r$ only affects the steady state response. This can be seen by computing the equilibrium point $x_e$ (by setting $dx/dt = 0$) and the steady-state output for the closed loop system, i.e.,

$$x_e = -A_{cl}^{-1}Bk_r r, \quad y_e = Cx_e + Du_e.$$  \hfill (6.46)

Substituting $x_e$ into the output expression and assuming that $D = 0$, we find the value of $k_r$ that ensures that $y_e = r$:

$$k_r = -\frac{1}{CA_{cl}^{-1}B}.$$  \hfill (6.47)

**Ackermann’s Formula**

For a controllable system, Ackermann’s formula provides a method for determining the gain matrix $K$:

$$K = \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix} C^{-1} p_{A_{cl}}(A),$$  \hfill (6.48)

where $C$ is the system’s controllability matrix and $p_{A_{cl}}(A)$ is the closed-loop characteristic polynomial evaluated at $s = A$. The controllability requirement can be understood from the inversion of the controllability matrix.
6.4. FEEDBACK CONTROL

6.4.4 Observability

So far, we have made the assumption that all states included in $x$ can be directly measured. In reality, this is not always true or practical. For instance, we may wish to measure the fluid flow around the surface of a wing, but the sensor readings $y$ consist of only a few pressure measurements. In this kind of situation, we attempt to reconstruct the internal states of the system through its external outputs and the applied input. The degree to which the states can be estimated is characterized by the **observability** of the system. Consider a linear system of the form

$$
\dot{q}(t) = Aq(t) + Bf(t), \quad (6.49)
$$
$$
y(t) = Cq(t). \quad (6.50)
$$

The system above is said to be observable if, for any $T > 0$, it is possible to determine the state of the system $x(T)$ through measurements of $y(t)$ and $u(t)$ on the interval $[0, T]$. It should be noted that we do not simply consider the output at a single time instant but rather watch the output over a time interval so that at least a short time history is available.

The underlying computations can be implemented with a dynamical system called an observer. Fig. 6.4 depicts a block diagram of a dynamical system with an observer. The output measurement $y$, which in general incorporates measurement noise $n$, is fed to the observer, which then outputs a state estimate $\hat{x}$.

![Block diagram of a system with an observer.](image)

**Testing for Observability**

Similarly to how we neglected the output when testing for reachability, we will neglect the effect of input when testing for observability; that is, we will consider the system dynamics

$$
\dot{x} = Ax, \quad y = Cx. \quad (6.51)
$$
CHAPTER 6. CONTROL

Notice that if the matrix $C$ is invertible, the observability problem can be directly solved by inverting $C$. To account for the general case, in which $C$ might not be invertible, we take the following approach. We take the output derivatives with respect to time. For the first derivative, we have

$$\frac{dy}{dt} = \frac{dx}{dt} = CAx,$$  \hspace{1cm} (6.52)

and we continue for the higher derivatives to construct the matrix form

$$\begin{bmatrix}
  y \\
  \dot{y} \\
  \ddot{y} \\
  \vdots \\
  y^{(n-1)}
\end{bmatrix} =
\begin{bmatrix}
  C \\
  CA \\
  CA^2 \\
  \vdots \\
  CA^{n-1}
\end{bmatrix}
\begin{bmatrix}
  x
\end{bmatrix}. \hspace{1cm} (6.53)

The state of the system can be determined if the observability matrix

$$\mathcal{O} =
\begin{bmatrix}
  C \\
  CA \\
  CA^2 \\
  \vdots \\
  CA^{n-1}
\end{bmatrix} \hspace{1cm} (6.54)

has $n$ independent rows or if $\mathcal{O}$ is full rank. It can be shown (by application of the Cayley-Hamilton theorem) that no derivatives or order higher than $n-1$ are needed. Note that this construction can be similarly implemented for systems with inputs; however, the observability criterion as formulated above is unchanged. Finally, note the the outlined method is not practically efficient, as the differentiation of the output can lead to large errors if the measurements are particularly noisy.

Further Reading

For more on state and output feedback, please see Sections 6.1, 6.2, and 7.1 of Åström and Murray [1].

6.5 PID Control

The general form of a proportional-integral-derivative controller (PID controller) can be defined as

$$u(t) = K \left( e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right), \hspace{1cm} (6.55)$$
6.5. PID CONTROL

where

\[ e(t) = r - y(t) \]  

(6.56)

is the error in the output, defined as the difference between the reference (set-point) \( r \) and the measured output \( y \). The control signal \( u(t) \) is the sum of three terms:

- the P-term, which is proportional to the error,
- the I-term, which is proportional to the integral of the error,
- and the D-term, which is proportional to the derivative of the error.

The controller parameters corresponding to these three terms respectively are (1) the proportional gain \( K \), (2) the integral time \( T_i \), and (3) the derivative time \( T_d \).

6.5.1 Effects of Proportional, Integral and Derivative Action

In the case of purely proportional control, which results from setting \( T_i = \infty \) and \( T_d = 0 \) in (6.55), there will always be a steady-state error (which can be proven with the final value theorem). Incorporating integral action (by decreasing the integral time \( T_i \)) leads to the disappearance of the steady state error, but it also leads to an undesirable increase in the system’s tendency for oscillations. This motivates the introduction of the derivative term, which adds damping to the system as \( T_d \) increases until a certain threshold determined by the dynamics of the system is reached.

More specifically, to see the effect of PI and PD feedback on these systems, consider a first-order system of the form

\[ \dot{y} + ay = f \]  

(6.57)

where \( f \) is the input, \( y \) is the output, and \( a \) is a parameter. Suppose that the control objective is to alter the dynamics of (6.57). We can write the PID controller as

\[ f(t) = -K_p y(t) - K_i \int_0^t y(\tau) d\tau - K_d \frac{dy}{dt}(t). \]  

(6.58)

Without feedback \((f = 0)\), the system has a pole at \(-a\), or a dynamic response of \( e^{-at} \). We may use feedback to alter the position of this pole. Choosing PI feedback \( f = -K_p y - K_i \int y dt \), the closed-loop system becomes

\[ \ddot{y} + (a + K_p) \dot{y} + K_i y = 0. \]  

(6.59)

The closed-loop system is now second order, and with this feedback, the poles are the roots of

\[ s^2 + (a + K_p)s + K_i = 0. \]  

(6.60)
Clearly, by appropriate choice of $K_p$ and $K_i$, we may place the two poles anywhere we desire in the complex plane since we have complete freedom over both coefficients in (6.60).

Next, consider a second-order system, a spring-mass system with equations of motion

$$m \ddot{y} + b \dot{y} + ky = f$$  \hspace{1cm} (6.61)

where $m$ is the mass, $b$ is the damping constant, and $k$ is the spring constant. With PD feedback $u = -K_p y - K_d \dot{y}$, the closed-loop system becomes

$$m \ddot{y} + (b + K_d) \dot{y} + (k + K_p) y = 0$$  \hspace{1cm} (6.62)

with closed-loop poles satisfying

$$ms^2 + (b + K_d)s + k + K_p = 0.$$  \hspace{1cm} (6.63)

We may place the poles anywhere desired since we have freedom to choose both of the coefficients in (6.63).

### 6.5.2 PID Performance and Tuning

A crucial issue in the use of PID control is how to control its performance by selecting appropriate gains – that is, how to tune the controller. There is not one single correct method by which to tune a PID controller in part because there is not one single correct PID tuning to control a given system. Rather, the choices of the three gains represents a set of tradeoffs.

One of the tradeoffs of control systems in general, and for PID controllers in particular, is the **damping ratio** $\zeta$, which measures the performance of the combination of the controller and the system being controlled. An **undamped** system ($\zeta = 0$) will oscillate forever, corresponding to a pure proportional ($K_p$ gain) controller. An **underdamped** system ($0 < \zeta < 1$) will oscillate for many cycles while eventually stabilizing to a steady state. This arrangement corresponds to a proportional-derivative (PD) controller in which the $K_d$ term is too small relative to the $K_p$ term. In contrast, an **overdamped** system ($\zeta > 1$) does not oscillate at all and approaches the setpoint very slowly. This situation is modeled by a PD controller in which the $K_d$ gain is too large relative to the $K_p$ term. Finally, in a **critically damped** system ($\zeta = 1$), there is no oscillation, but the system quickly converges to the setpoint and remains there. In this case, the $K_p$ and $K_d$ terms are appropriately chosen with respect to one another.

Another important characteristic of controllers, including PID controllers, is stiffness. It may be helpful here to think of a robot arm coming into contact with a hard metal table. A **stiff** system, meaning it is resistant to deviation from the
6.5. PID CONTROL

setpoint, can be characterized by a high proportional \((K_p)\) gain. In a stiff system at the setpoint, it takes a great deal of effort to remove it from the setpoint. If sufficient effort is applied, the system may oscillate at a high frequency (higher than the characteristic frequency of the uncontrolled system). A stiff arm in contact with a hard table is dangerous and likely to cause damage or injury. The advantage of a stiff controller is that it is usually very precise.

The opposite of a stiff system is a compliant system, which tends to have a small \(K_p\) gain. Unlike a stiff system, a compliant system is much more tolerant of not being at the setpoint. If allowed to oscillate, it will do so close to the characteristic or resonant frequency of the system being controlled. Compliant systems are preferred in situations where contact is involved, especially contact with humans.

It is also worth considering the choice of the integral \((K_i)\) gain. Since integral control can build up over time, it is useful for “sticky” systems that are easier to manipulate once they become unstuck. A common example of a “sticky” system in robotics is static friction. If trying to slide one part against another, it often requires more force to begin sliding the part than it does to continue sliding. In contrast, a system with a high sliding friction as well would be better controlled by increasing the \(K_p\) and \(K_d\) gains. \(K_i\) gains are considered somewhat dangerous because they can build up and spontaneously release a lot of energy into the system being controlled. For many applications, it is better to minimize or avoid the use of \(K_i\) gains for safety.

The Ziegler-Nichols Tuning Method

The most popular heuristic method for tuning a PID controller is the Ziegler-Nichols tuning method. This method involves several steps.

1. Set all gains to zero.
2. Increase the \(K_p\) gain until the system has stable and consistent oscillations. At each step increment, you will need to wait for the system to re-stabilize in order to know if it is in a steady oscillation. This value of the \(K_p\) at this point will be referred to as the ultimate gain, \(K_u\). Note the oscillation period \(T_u\) of the ultimate gain.
3. Set \(K_p = 0.6K_u\).
4. Set \(K_i = T_u/2\).
5. Set \(K_d = T_u/8\).

These values seem arbitrary, but they were arrived at through experimentation. Ziegler and Nichols tried hand-tuning PID controllers for a variety of systems until they got good performance. They then looked for a small set of parameters that would explain the patterns they saw – these were \(K_u\) and \(T_u\).
In hand-tuning their PID controllers, Ziegler and Nichols were tuning the controlled system to a *quarter amplitude decay ratio*, which means that the controller is designed to be underdamped – it overshoots the setpoint several times before converging, but the amplitude of each overshoot is one-quarter of the previous overshoot. This tuning is not guaranteed to be optimal for all controlled systems, and so Ziegler-Nichols is not guaranteed optimal either. But for many applications in robotics, it is a good starting point.

### 6.6 Path-Following Controllers

Whereas some controllers are designed to keep a system at a stationary setpoint, other kinds of controllers drive a system along a time-varying trajectory. An interesting and useful application of trajectory-following control in robotics is to mobile robots. Since many mobile robots have nonholonomic constraints, the control problem becomes challenging due to the fact that error corrections must comply with motion constraints.

The route to be followed by a mobile robot can be expressed as a trajectory parameterized by time,

\[ p_t(t) = [x_t(t) \quad y_t(t) \quad \theta_t(t)]^T, \]

or a path parameterized by length,

\[ p_s(s) = [x_s(s) \quad y_s(s) \quad \theta_s(s)]^T. \]

Translating between these functions is a matter of reparameterizing via a function \( r: [0, t_f] \rightarrow [0, s_f] \), so that \( p_t(t) = p_s(r(t)) \).

#### Path-Following Error Types

Suppose we have a target trajectory for a mobile robot to follow. If we define frame \( \{t\} \) at the desired instantaneous pose \( p_t(t) \), then the actual robot pose \( p_B^t \) represents the deviation of the robot’s state from the desired state. The deviation is described in terms of three errors that correspond to the three rigid-body freedoms:

- **The along-track error** \( \delta s \) is the distance ahead or behind the target in the instantaneous direction of motion. Intuitively, along-track error can be reduced simply by adjusting the robot’s velocity.

- **The cross-track error** \( \delta n \) is the portion of the position error orthogonal to the intended direction of motion. Thus, cross-track error is harder to correct since it requires a velocity in a direction that is forbidden by the nonholonomic constraint, if one exists.
6.6. PATH-FOLLOWING CONTROLLERS

- The **heading error** $\delta \theta$ is the difference between the desired heading and the actual heading.

All three errors are interrelated since an attempt to correct one has the potential to increase the others. Fig. 6.5 shows an illustration of the three error types.

**Open-Loop Path Following**

An open-loop path-following controller executes the command that would be required to follow the path if the mobile robot were located at the correct pose $p_t(t)$ at time $t$. We can write an open-loop controller for a robot that is naturally controlled via angular velocity, such as a differential-drive robot:

$$
u_{\omega, OL}(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_t(t)^2 + \dot{y}_t(t)^2} \\ \dot{\theta}_t(t) \end{bmatrix}.$$  \hspace{1cm} (6.66)

We can similarly control a mobile robot with car-like steering via curvature:

$$
u_{\kappa, OL}(t) = \begin{bmatrix} v(t) \\ \kappa(t) \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_t(t)^2 + \dot{y}_t(t)^2} \\ \frac{\dot{\theta}_t(t)}{\sqrt{\dot{x}_t(t)^2 + \dot{y}_t(t)^2}} \end{bmatrix}.$$  \hspace{1cm} (6.67)

Naturally, these open-loop controllers will not correct any errors that occur during execution due to wheel-slip or to models that map linear and angular velocity into low-level actuator control inputs. We can therefore expect that error increases over time proportional to the square root of distance traveled. In particular, a heading error will tend to amplify a cross-track error over time.

**Pure Pursuit**

The pure pursuit algorithm implements feedback control for path-following on a mobile robot. It is defined based on a correction to the open-loop path-following
controller, as

\[ u_{\kappa,CL}(t) = u_{\kappa,OL}(t) + \delta u(t) \]  \hspace{1cm} (6.68)

\[ = u_{\kappa,OL}(t) + K \begin{bmatrix} \delta s \\ \delta n \\ \delta \theta \end{bmatrix}, \]  \hspace{1cm} (6.69)

where \( K \) is a gain matrix that is used to tune the controller. Let

\[ K = \begin{bmatrix} -k_s & 0 & 0 \\ 0 & -k_n & -k_{\theta} \end{bmatrix}. \]  \hspace{1cm} (6.70)

The pure-pursuit controller produced by this gain matrix performs a hybrid of P- and PD-control. It uses a simple P-controller to correct along-track error by simply speeding up or slowing down the robot. The control on curvature \( \kappa \) is a PD-controller for cross-track error because \( \delta \theta \) is related to the derivative of \( \delta n \).

If we zero out \( k_{\theta} \), we get a P-controller on \( \kappa \), which will tend to oscillate left and right along the path. The correction in cross-track error generally induces a heading error, so that by the time cross-track error becomes zero, the heading is off and the robot misses the path entirely.

**Pure Pursuit Analysis**

Corrections on cross-track error and heading error tend to counteract one another. The gain \( k_{\theta} \) tends to decrease the rate of reduction in cross-track error, which is why it is a damping term. As the robot moves closer to the path, cross-track error shrinks, and the heading error dominates more. As a consequence, the controller will begin to drive the robot more parallel to the path. The emergent effect of the pure pursuit controller is that at any moment in time, it seeks to reacquire the path smoothly at some future point in time and at a forward position along the path.

**Pure Pursuit Implementation**

A geometric intuition about the pure pursuit controller (depicted in Fig. 6.6) may be helpful for its implementation. This model uses *receding horizon control*, in which the controller maintains an explicit target at some fixed lookahead time (or distance) along the path. As the robot moves, the target recedes so that its relative interval of time (or distance) along the path remains constant.

At each iteration of the controller, the robot computes a corrective path that will intersect the desired path at the horizon target. The form used for this corrective path is a constant-curvature arc, which is uniquely specified by the robot’s current...
position and heading (because it can instantaneously move only straight ahead) as well as the final target point. The curvature of that arc may change with each iteration to reflect the shape of curve necessary to intersect the path at the horizon.

With each iteration, the robot executes a constant velocity and curvature for a short duration of time. The subsequent changes in curvature lead to a nearly smooth traversal that reacquires the path. It can be shown that this geometric algorithm is equivalent to a tuning of the PD-controller described previously.

Figure 6.6: A pure pursuit controller can be implemented using a receding horizon (initially the red segment). Under this implementation, a constant-curvature arc is fitted between the current pose of the robot and the horizon point on the path. Each arc connects a small dot at the robot’s current pose to a large dot at the current horizon.

Further Reading

For more information on path-following algorithms, see Kelly [2].

Bibliography

