Feedback Systems

An Introduction for Scientists and Engineers
SECOND EDITION

Karl Johan Åström
Richard M. Murray

Version v3.0h (25 Sep 2016)

This is the electronic edition of *Feedback Systems* and is available from http://www.cds.caltech.edu/~murray/FBS. Hardcover editions may be purchased from Princeton University Press, http://press.princeton.edu/titles/8701.html.

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Chapter Three
System Modeling

> *... I asked Fermi whether he was not impressed by the agreement between our calculated numbers and his measured numbers. He replied, "How many arbitrary parameters did you use for your calculations?" I thought for a moment about our cut-off procedures and said, "Four." He said, "I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk."*
>
> Freeman Dyson on describing the predictions of his model for meson-proton scattering to Enrico Fermi in 1953 [Dys04].

A model is a precise representation of a system's dynamics used to answer questions via analysis and simulation. The model we choose depends on the questions we wish to answer, and so there may be multiple models for a single dynamical system, with different levels of fidelity depending on the phenomena of interest. In this chapter we provide an introduction to the concept of modeling and present some basic material on two specific methods commonly used in feedback and control systems: differential equations and difference equations.

3.1 Modeling Concepts

A *model* is a mathematical representation of a physical, biological or information system. Models allow us to reason about a system and make predictions about how a system will behave. In this text, we will mainly be interested in models of dynamical systems describing the input/output behavior of systems, and we will often work in "state space" form. As pointed out already in Chapter 1, when using models it is important to keep in mind that they are an *approximation* of the underlying system. Analysis and design using models must always be done carefully to insure that the limits of the model are respected.

Roughly speaking, a dynamical system is one in which the effects of actions do not occur immediately. For example, the velocity of a car does not change immediately when the gas pedal is pushed nor does the temperature in a room rise instantaneously when a heater is switched on. Similarly, a headache does not vanish right after an aspirin is taken, requiring time for it to take effect. In business systems, increased funding for a development project does not increase revenues in the short term, although it may do so in the long term (if it was a good investment). All of these are examples of dynamical systems, in which the behavior of the system evolves with time.

In the remainder of this section we provide an overview of some of the key concepts in modeling. The mathematical details introduced here are explored more

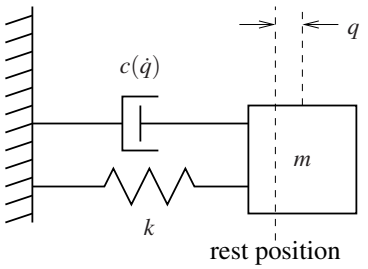


Figure 3.1: Spring–mass system with nonlinear damping. The position of the mass is denoted by q, with $q = 0$ corresponding to the rest position of the spring. The forces on the mass are generated by a linear spring with spring constant k and a damper with force dependent on the velocity $\dot{q}$.

fully in the remainder of the chapter.

The Heritage of Mechanics

The study of dynamics originated in attempts to describe planetary motion. The basis was detailed observations of the planets by Tycho Brahe and the results of Kepler, who found empirically that the orbits of the planets could be well described by ellipses. Newton embarked on an ambitious program to try to explain why the planets move in ellipses, and he found that the motion could be explained by his law of gravitation and the formula stating that force equals mass times acceleration. In the process he also invented calculus and differential equations.

One of the triumphs of Newton's mechanics was the observation that the motion of the planets could be predicted based on the current positions and velocities of all planets. It was not necessary to know the past motion. The *state* of a dynamical system is a collection of variables that completely captures the past motion of a system for the purpose of predicting future motion. For a system of planets the state is simply the positions and the velocities of the planets. We call the set of all possible states the *state space*.

A common class of mathematical models for dynamical systems is ordinary differential equations (ODEs). In mechanics, one of the simplest such differential equations is that of a spring–mass system with damping:

$$m\ddot{q} + c(\dot{q}) + kq = 0. \tag{3.1}$$

This system is illustrated in Figure 3.1. The variable $q \in \mathbb{R}$ represents the position of the mass m with respect to its rest position. We use the notation $\dot{q}$ to denote the derivative of q with respect to time (i.e., the velocity of the mass) and $\ddot{q}$ to represent the second derivative (acceleration). The spring is assumed to satisfy Hooke's law, which says that the force is proportional to the displacement. The friction element (damper) is taken as a nonlinear function $c(\dot{q})$, which can model effects such as stiction and viscous drag. The position q and velocity $\dot{q}$ represent the instantaneous state of the system. We say that this system is a *second-order system* since it has two states which we combine in the *state vector* $x = (q, \dot{q})$.

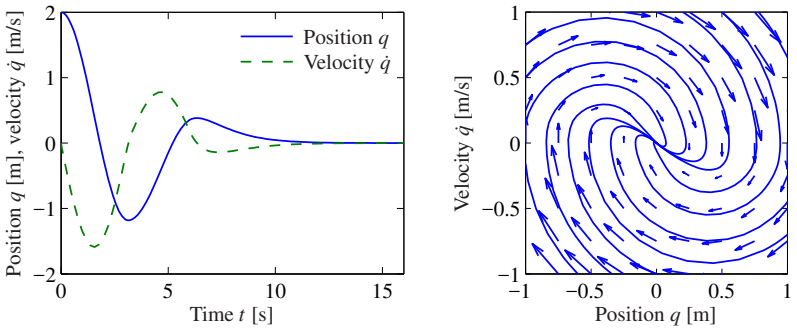


Figure 3.2: Illustration of a state model. A state model gives the rate of change of the state as a function of the state. The plot on the left shows the evolution of the state as a function of time. The plot on the right, called a *phase portrait*, shows the evolution of the states relative to each other, with the velocity of the state denoted by arrows.

The evolution of the position and velocity can be described using either a time plot or a phase portrait, both of which are shown in Figure 3.2. The *time plot*, on the left, shows the values of the individual states as a function of time. The *phase portrait*, on the right, shows the traces of some of the states from different initial conditions: it illustrates how the states move in the state space. In the phase portrait we have also shown arrows that represent the velocity $\dot{x}$ of the state x in a few points. The phase portrait gives a strong intuitive representation of the equation as a vector field or a flow.While systems of second order (two states) can be represented in this way, unfortunately it is difficult to visualize equations of higher order using this approach.

The differential equation (3.1) is called an *autonomous* system because there are no external influences. (Note that this usage of "autonomous" is slightly different than in the phrase "autonomous vehicle".) Such a model is natural for use in celestial mechanics because it is difficult to influence the motion of the planets. In many examples, it is useful to model the effects of external disturbances or controlled forces on the system. One way to capture this is to replace equation (3.1) by

$$m\ddot{q} + c(\dot{q}) + kq = u, \tag{3.2}$$

where u represents the effect of external inputs. The model (3.2) is called a *forced* or *controlled differential equation*. It implies that the rate of change of the state can be influenced by the input $u(t)$. Adding the input makes the model richer and allows new questions to be posed. For example, we can examine what influence external disturbances have on the trajectories of a system. Or, in the case where the input variable is something that can be modulated in a controlled way, we can analyze whether it is possible to "steer" the system from one point in the state space to another through proper choice of the input.

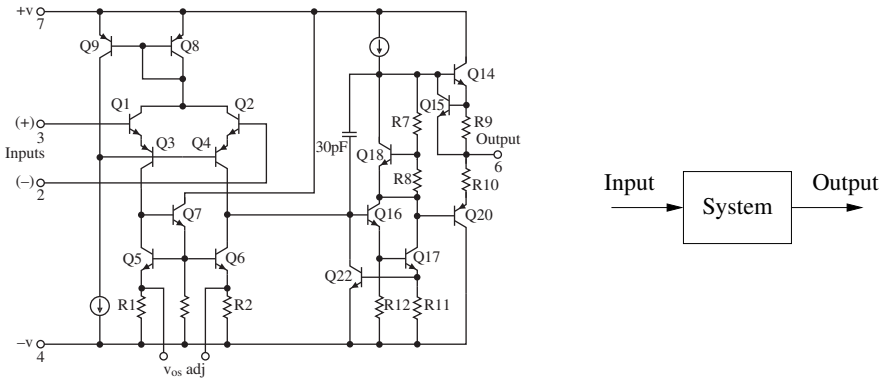


Figure 3.3: Illustration of the input/output view of a dynamical system. The figure on the left shows a detailed circuit diagram for an electronic amplifier; the one on the right is its representation as a block diagram.

The Heritage of Electrical Engineering

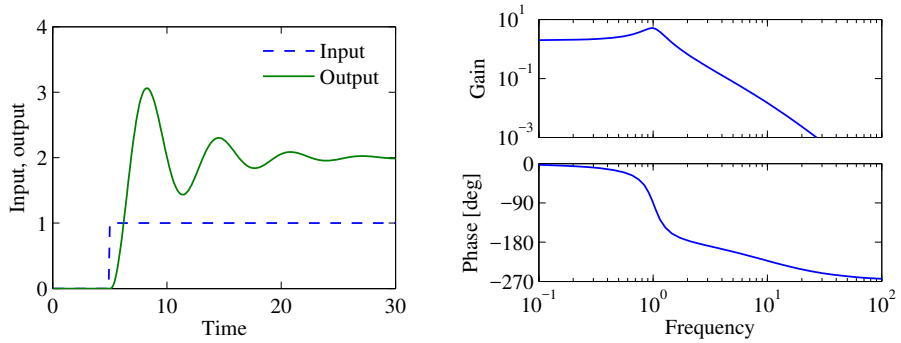
A different view of dynamics emerged from electrical engineering, where the design of electronic amplifiers led to a focus on input/output behavior. A system was considered a device that transforms inputs to outputs, as illustrated in Figure 3.3. Conceptually an input/output model can be viewed as a giant table of inputs and outputs. Given an input signal $u(t)$ over some interval of time, the model should produce the resulting output $y(t)$.

The input/output framework is used in many engineering disciplines since it allows us to decompose a system into individual components connected through their inputs and outputs. Thus, we can take a complicated system such as a radio or a television and break it down into manageable pieces such as the receiver, demodulator, amplifier and speakers. Each of these pieces has a set of inputs and outputs and, through proper design, these components can be interconnected to form the entire system.

The input/output view is particularly useful for the special class of *linear time-invariant systems*. This term will be defined more carefully later in this chapter, but roughly speaking a system is linear if the superposition (addition) of two inputs yields an output that is the sum of the outputs that would correspond to individual inputs being applied separately. A system is time-invariant if the output response for a given input does not depend on when that input is applied.

Many electrical engineering systems can be modeled by linear time-invariant systems, and hence a large number of tools have been developed to analyze them. One such tool is the *step response*, which describes the relationship between an input that changes from zero to a constant value abruptly (a step input) and the corresponding output. As we shall see later in the text, the step response is very useful in characterizing the performance of a dynamical system, and it is often used to specify the desired dynamics. A sample step response is shown in Figure 3.4a.

Another way to describe a linear time-invariant system is to represent it by its



(a) Step response (b) Frequency response

Figure 3.4: Input/output response of a linear system. The step response (a) shows the output of the system due to an input that changes from 0 to 1 at time $t = 5$ s. The frequency response (b) shows the amplitude gain and phase change due to a sinusoidal input at different frequencies.

response to sinusoidal input signals. This is called the *frequency response*, and a rich, powerful theory with many concepts and strong, useful results has emerged. The results are based on the theory of complex variables and Laplace transforms. The basic idea behind frequency response is that we can completely characterize the behavior of a system by its steady-state response to sinusoidal inputs. Roughly speaking, this is done by decomposing any arbitrary signal into a linear combination of sinusoids (e.g., by using the Fourier transform) and then using linearity to compute the output by combining the response to the individual frequencies. A sample frequency response is shown in Figure 3.4b.

The input/output view lends itself naturally to experimental determination of system dynamics, where a system is characterized by recording its response to particular inputs, e.g., a step or a set of sinusoids over a range of frequencies.

The Control View

When control theory emerged as a discipline in the 1940s, the approach to dynamics was strongly influenced by the electrical engineering (input/output) view. A second wave of developments in control, starting in the late 1950s, was inspired by mechanics, where the state space perspective was used. The emergence of space flight is a typical example, where precise control of the orbit of a spacecraft is essential. These two points of view gradually merged into what is today the state space representation of input/output systems. In the 1970s the development was influenced by advances in automation, which emphasized the need to include logic and sequencing.

The development of state space models involved modifying the models from mechanics to include external actuators and sensors and utilizing more general

forms of equations. In control, the model given by equation (3.2) was replaced by

$$\frac{dx}{dt} = f(x,u), \qquad y = h(x,u), \tag{3.3}$$

where x is a vector of state variables, u is a vector of control signals and y is a vector of measurements. The term dx/dt represents the derivative of the vector x with respect to time, and f and h are (possibly nonlinear) mappings of their arguments to vectors of the appropriate dimension. For mechanical systems, the state consists of the position and velocity of the system, so that $x = (q, \dot{q})$ in the case of a damped spring–mass system. Note that in the control formulation we model dynamics as first-order differential equations, but we will see that this can capture the dynamics of higher-order differential equations by appropriate definition of the state and the maps f and h.

Adding inputs and outputs has increased the richness of the classical problems and led to many new concepts. For example, it is natural to ask if possible states x can be reached with the proper choice of u (reachability) and if the measurement y contains enough information to reconstruct the state (observability). These topics will be addressed in greater detail in Chapters 7 and 8.

A final development in building the control point of view was the emergence of disturbances and model uncertainty as critical elements in the theory. The simple way of modeling disturbances as deterministic signals like steps and sinusoids has the drawback that such signals cannot be predicted precisely. A more realistic approach is to model disturbances as random signals. This viewpoint gives a natural connection between prediction and control. The dual views of input/output representations and state space representations are particularly useful when modeling systems with uncertainty since state models are convenient to describe a nominal model but uncertainties are easier to describe using input/output models (often via a frequency response description). Uncertainty will be a constant theme throughout the text and will be studied in particular detail in Chapter 13.

An interesting observation in the design of control systems is that feedback systems can often be analyzed and designed based on comparatively simple models. The reason for this is the inherent robustness of feedback systems. However, other uses of models may require more complexity and more accuracy. One example is feedforward control strategies, where one uses a model to precompute the inputs that cause the system to respond in a certain way. Another area is system validation, where one wishes to verify that the detailed response of the system performs as it was designed. Because of these different uses of models, it is common to use a hierarchy of models having different complexity and fidelity.

Multidomain Modeling

Modeling is an essential element of many disciplines, but traditions and methods from individual disciplines can differ from each other, as illustrated by the previous discussion of mechanical and electrical engineering. A difficulty in systems engineering is that it is frequently necessary to deal with heterogeneous systems

from many different domains, including chemical, electrical, mechanical and information systems.

To model such multidomain systems, we start by partitioning a system into smaller subsystems. Each subsystem is represented by balance equations for mass, energy and momentum, or by appropriate descriptions of information processing in the subsystem. The behavior at the interfaces is captured by describing how the variables of the subsystem behave when the subsystems are interconnected. These interfaces act by constraining variables within the individual subsystems to be equal (such as mass, energy or momentum fluxes). The complete model is then obtained by combining the descriptions of the subsystems and the interfaces.

Using this methodology it is possible to build up libraries of subsystems that correspond to physical, chemical and informational components. The procedure mimics the engineering approach where systems are built from subsystems that are themselves built from smaller components. As experience is gained, the components and their interfaces can be standardized and collected in model libraries. In practice, it takes several iterations to obtain a good library that can be reused for many applications.

State models or ordinary differential equations are not suitable for component-based modeling of this form because states may disappear when components are connected. This implies that the internal description of a component may change when it is connected to other components. As an illustration we consider two capacitors in an electrical circuit. Each capacitor has a state corresponding to the voltage across the capacitors, but one of the states will disappear if the capacitors are connected in parallel. A similar situation happens with two rotating inertias, each of which is individually modeled using the angle of rotation and the angular velocity. Two states will disappear when the inertias are joined by a rigid shaft.

This difficulty can be avoided by replacing differential equations by *differential algebraic equations*, which have the form

$$F(z,\dot{z}) = 0,$$

where $z \in \mathbb{R}^n$. A simple special case is

$$\dot{x} = f(x,y), \qquad g(x,y) = 0, \tag{3.4}$$

where $z = (x,y)$ and $F = (\dot{x} - f(x,y), g(x,y))$. The key property is that the derivative $\dot{z}$ is not given explicitly and there may be pure algebraic relations between the components of the vector z. Modeling using differential algebraic equations is also called *equation-based modeling*, *acausal modeling* or *behavioral modeling*.

The model (3.4) captures the examples of the parallel capacitors and the linked rotating inertias. For example, when two capacitors are connected, we simply add the algebraic equation expressing that the voltages across the capacitors are the same.

Modelica is a language that has been developed to support component-based modeling. Differential algebraic equations are used as the basic description, and object-oriented programming is used to structure the models. Modelica is used to

model the dynamics of technical systems in domains such as mechanical, electrical, thermal, hydraulic, thermofluid and control subsystems. Modelica is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users. A large set of free and commercial Modelica component libraries are available and are used by a growing number of people in industry, research and academia. For further information about Modelica, see http://www.modelica.org or Tiller [Til01].

Finite State Machines and Hybrid Systems

A final type of modeling has been developed within the computer-controlled systems community. A hybrid system (also called a *cyberphysical system*) is one that combines continuous dynamics with discrete logic. The discrete portion of the system represents logical variables that reside in a computer, such as the mode of a system (on, off, degraded, etc.).

Discrete state dynamics are often represented using a *finite state machine* that consists of a finite set of discrete states $\alpha \in \mathbb{Q}$. We can think of α as the "mode" of the system. The dynamics of a finite state machine are defined in terms of transitions between the states. One convenient representation is as a *guarded transition system*:

$$g_i(\alpha, \beta) \implies \alpha' = r_i(\alpha), \qquad i = 1, \ldots, N.$$

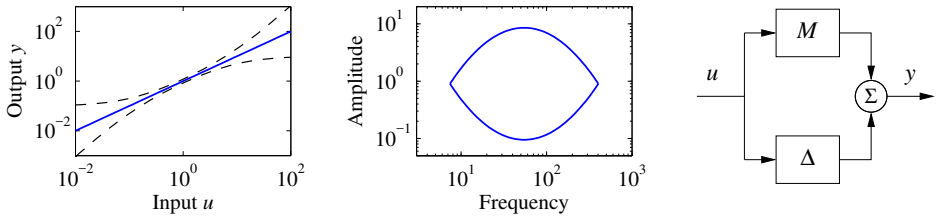
Here the function g is a Boolean (true/false) function that depends on the current system mode α and an input β, which might represent an environmental event (button press, component failure, etc). If the guard g_i is true then the system transitions from the current state α to a new state α', determined by the rule (transition map) r_i. A guarded transition system can have many different rules, depending on the system state and external input.

It is also possible to combine systems that have finite states with those having discrete states, creating a *hybrid system*. For example, if a system has a continuous state x and discrete state α, we might write the overall system dynamics as

$$\frac{dx}{dt} = f_\alpha(x, u, v), \quad g_i(x, \alpha, \beta) \implies \alpha' = r_i(x, \alpha), \qquad i = 1, \ldots, N.$$

In this representation, the continuous dynamics (with state x) are governed by an ordinary differential equation that may depend on the system mode α (indicated by the subscript in f_α). The discrete transition system is also influenced by the continuous state, so that the guards g_i and rules r_i now depend on the continuous state.

Many other representations are possible for hybrid systems, including models that allow a non-continuous change in the continuous variables when a change in the discrete state occurs (so-called *reset logic*). Computer modeling packages for hybrid systems include StateFlow (part of the MATLAB suite of tools), Modelica and Ptolemy [Pto14].



(a) Static uncertainty (b) Uncertainty lemon (c) Model uncertainty

Figure 3.5: Characterization of model uncertainty. Uncertainty of a static system is illustrated in (a), where the solid line indicates the nominal input/output relationship and the dashed lines indicate the range of possible uncertainty. The uncertainty lemon [GPD59] in (b) is one way to capture uncertainty in dynamical systems emphasizing that a model is valid only in the amplitude and frequency ranges within the shaded region. In (c) a model is represented by a nominal model M and another model Δ representing the uncertainty analogous to the representation of parameter uncertainty.

Model Uncertainty

Reducing uncertainty is one of the main reasons for using feedback, and it is therefore important to characterize uncertainty. When making measurements, there is a good tradition to assign both a nominal value and a measure of uncertainty. It is useful to apply the same principle to modeling, but unfortunately it is often difficult to express the uncertainty of a model quantitatively.

For a static system whose input/output relation can be characterized by a function, uncertainty can be expressed by an uncertainty band as illustrated in Figure 3.5a. At low signal levels there are uncertainties due to sensor resolution, friction and quantization. For example, some models for queuing systems or cells are based on averages that exhibit significant variations for small populations. At large signal levels there are saturations or even system failures. The signal ranges where a model is reasonably accurate vary dramatically between applications, but it is rare to find models that are accurate for signal ranges larger than 10^4.

Characterization of the uncertainty of a dynamic model is much more difficult. We can try to capture uncertainties by assigning uncertainties to parameters of the model, but this is often not sufficient. There may be errors due to phenomena that have been neglected, e.g., small time delays. In control the ultimate test is how well a control system based on the model performs, and time delays can be important. There is also a frequency aspect. There are slow phenomena, such as aging, that can cause changes or drift in the systems. There are also high-frequency effects: a resistor will no longer be a pure resistance at very high frequencies, and a beam has stiffness and will exhibit additional dynamics when subject to high-frequency excitation. The *uncertainty lemon* [GPD59] shown in Figure 3.5b is one way to conceptualize the uncertainty of a system. It illustrates that a model is valid only in certain amplitude and frequency ranges.

We will introduce some formal tools for representing uncertainty in Chapter 13

using figures such as Figure 3.5c. These tools make use of the concept of a transfer function, which describes the frequency response of an input/output system. For now, we simply note that one should always be careful to recognize the limits of a model and not to make use of models outside their range of applicability. For example, one can describe the uncertainty lemon and then check to make sure that signals remain in this region. In early analog computing, a system was simulated using operational amplifiers, and it was customary to give alarms when certain signal levels were exceeded. Similar features can be included in digital simulation.

Algebraic Loops

When analyzing or simulating a system described by a block diagram, we need to form the differential equations that describe the complete system. In many cases the equations can be obtained by combining the differential equations that describe each subsystem and substituting variables. This simple procedure cannot be used when there are closed loops of subsystems that all have a direct connection between inputs and outputs, known as an *algebraic loop*. A *direct connection* means that a change in the output u gives an instantaneous change in the output y.

To see what can happen, consider a system with two blocks, a first-order nonlinear system,

$$\frac{dx}{dt} = f(x,u), \qquad y = h(x), \tag{3.5}$$

and a proportional controller described by $u = -ky$. There is no direct connection since the function h does not depend on u. In that case we can obtain the equation for the closed loop system simply by replacing u by $-ky = -kh(x)$ in equation (3.5) to give

$$\frac{dx}{dt} = f(x,-kh(x)), \qquad y = h(x),$$

which is an ordinary differential equation.

The situation is more complicated if there is a direct connection. If $y = h(x,u)$, then replacing u by $-ky$ gives

$$\frac{dx}{dt} = f(x,-ky), \qquad y = h(x,-ky).$$

To obtain a differential equation for x, the algebraic equation $y = h(x,-ky)$ must be first be solved to give $y = \alpha(x)$, which in general is a complicated task.

When algebraic loops are present, it is necessary to solve algebraic equations to obtain the differential equations for the complete system. The resulting model becomes a set of differential algebraic equation, similar to equation 3.4. Resolving algebraic loops is a nontrivial problem because it requires the symbolic solution of algebraic equations. Most block diagram-oriented modeling languages cannot handle algebraic loops, and they simply give a diagnosis that such loops are present. In the era of analog computing, algebraic loops were eliminated by introducing

fast dynamics between the loops. This created differential equations with fast and slow modes that are difficult to solve numerically. Advanced modeling languages like Modelica use several sophisticated methods to resolve algebraic loops.

3.2 State Space Models

In this section we introduce the two primary forms of models that we use in this text: differential equations and difference equations. Both make use of the notions of state, inputs, outputs and dynamics to describe the behavior of a system. We also briefly discuss modeling of finite state systems.

Ordinary Differential Equations

The state of a system is a collection of variables that summarize the past of a system for the purpose of predicting the future. For a physical system the state is composed of the variables required to account for storage of mass, momentum and energy. A key issue in modeling is to decide how accurately this storage has to be represented. The state variables are gathered in a vector $x \in \mathbb{R}^n$ called the *state vector*. The control variables are represented by another vector $u \in \mathbb{R}^p$, and the measured signal by the vector $y \in \mathbb{R}^q$. A system can then be represented by the differential equation

$$\frac{dx}{dt} = f(x,u), \qquad y = h(x,u), \tag{3.6}$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ and $h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q$ are smooth mappings. We call a model of this form a *state space model*.

The dimension of the state vector is called the *order* of the model. The model given in equation (3.6) is called *time-invariant* because the functions f and h do not depend explicitly on time t; there are more general time-varying systems where the functions do depend on time. The model consists of two functions: the function f gives the rate of change of the state vector as a function of state x and control u, and the function h gives the measured values as functions of state x and control u.

A model is called a *linear* state space model (or often just a "linear system") if the functions f and h are linear in x and u. A linear state space model can thus be represented by

$$\frac{dx}{dt} = Ax + Bu, \qquad y = Cx + Du, \tag{3.7}$$

where A, B, C and D are constant matrices. Such a model is said to be *linear and time-invariant*, or LTI for short. (In this text we will usually omit the term time-invariant and just say the model is linear.) The matrix A is called the *dynamics matrix*, the matrix B is called the *control matrix*, the matrix C is called the *sensor matrix* and the matrix D is called the *direct term*. Frequently models will not have a direct term, indicating that the control signal does not influence the output directly.

A different form of linear differential equations, generalizing the second-order dynamics from mechanics, is an equation of the form

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = u, \tag{3.8}$$

where t is the independent (time) variable, $y(t)$ is the dependent (output) variable and $u(t)$ is the input. The notation $d^k y/dt^k$ is used to denote the kth derivative of y with respect to t, sometimes also written as $y^{(k)}$. The controlled differential equation (3.8) is said to be an nth-order model. This model can be converted into state space form by defining

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d^{n-1}y/dt^{n-1} \\ d^{n-2}y/dt^{n-2} \\ \vdots \\ dy/dt \\ y \end{pmatrix},$$

and the state space equations become

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} -a_1 x_1 - \cdots - a_n x_n \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{pmatrix} + \begin{pmatrix} u \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \qquad y = x_n.$$

With the appropriate definitions of A, B, C and D, this equation is in linear state space form.

An even more general model is obtained by letting the output be a linear combination of the states of the model, i.e.,

$$y = b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + du.$$

This model can be represented in state space as

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} u, \tag{3.9}$$

$$y = \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix} x + du.$$

This particular form of a linear state space model is called *reachable canonical form* and will be studied in more detail in later chapters. Many other representations for a model are possible and we shall see several of these in Chapters 6–8. It is also possible to expand the form of equation (3.8) to allow derivatives of the input to appear, as we saw briefly in Chapter 2.

(a) Segway (b) Saturn rocket (c) Cart–pendulum system

Figure 3.6: Balance systems. (a) Segway Personal Transporter, (b) Saturn rocket and (c) inverted pendulum on a cart. Each of these examples uses forces at the bottom of the system to keep it upright.

Example 3.1 Balance systems

An example of a type of system that can be modeled using ordinary differential equations is the class of *balance systems*. A balance system is a mechanical system in which the center of mass is balanced above a pivot point. Some common examples of balance systems are shown in Figure 3.6. The Segway® Personal Transporter (Figure 3.6a) uses a motorized platform to stabilize a person standing on top of it. When the rider leans forward, the transportation device propels itself along the ground but maintains its upright position. Another example is a rocket (Figure 3.6b), in which a gimbaled nozzle at the bottom of the rocket is used to stabilize the body of the rocket above it. Other examples of balance systems include humans or other animals standing upright or a person balancing a stick on their hand.

Balance systems are a generalization of the spring–mass system we saw earlier. We can write the dynamics for a mechanical system in the general form

$$M(q)\ddot{q} + C(q,\dot{q}) + K(q) = B(q)u,$$

where $M(q)$ is the inertia matrix for the system, $C(q,\dot{q})$ represents the Coriolis forces as well as the damping, $K(q)$ gives the forces due to potential energy and $B(q)$ describes how the external applied forces couple into the dynamics. Note that q may be a vector, rather than just a scalar, and represents the configuration variables of the system. The specific form of the equations can be derived using Newtonian mechanics. Each of the terms depends on the configuration of the system q and these terms are often nonlinear in the configuration variables.

Figure 3.6c shows a simplified diagram for a balance system consisting of an inverted pendulum on a cart. To model this system, we choose state variables that represent the position and velocity of the base of the system, p and $\dot{p}$, and the angle and angular rate of the structure above the base, θ and $\dot{\theta}$. We let F represent the force applied at the base of the system, assumed to be in the horizontal direc-

tion (aligned with p), and choose the position and angle of the system as outputs. With this set of definitions, the dynamics of the system can be computed using Newtonian mechanics and have the form

$$\begin{pmatrix} (M+m) & -ml\cos\theta \\ -ml\cos\theta & (J+ml^2) \end{pmatrix} \begin{pmatrix} \ddot{p} \\ \ddot{\theta} \end{pmatrix} + \begin{pmatrix} c\dot{p} + ml\sin\theta\,\dot{\theta}^2 \\ \gamma\dot{\theta} - mgl\sin\theta \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}, \qquad (3.10)$$

where M is the mass of the base, m and J are the mass and moment of inertia of the system to be balanced, l is the distance from the base to the center of mass of the balanced body, c and γ are coefficients of viscous friction and g is the acceleration due to gravity.

We can rewrite the dynamics of the system in state space form by defining the state as $x = (p, \theta, \dot{p}, \dot{\theta})$, the input as $u = F$ and the output as $y = (p, \theta)$. If we define the total mass and total inertia as

$$M_t = M + m, \qquad J_t = J + ml^2,$$

the equations of motion then become

$$\frac{d}{dt}\begin{pmatrix} p \\ \theta \\ \dot{p} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{p} \\ \dot{\theta} \\ \dfrac{-mls_\theta\dot{\theta}^2 + mg(ml^2/J_t)s_\theta c_\theta - c\dot{p} - (\gamma/J_t)mlc_\theta\dot{\theta} + u}{M_t - m(ml^2/J_t)c_\theta^2} \\ \dfrac{-ml^2 s_\theta c_\theta\dot{\theta}^2 + M_t gls_\theta - clc_\theta\dot{p} - \gamma(M_t/m)\dot{\theta} + lc_\theta u}{J_t(M_t/m) - m(lc_\theta)^2} \end{pmatrix},$$

$$y = \begin{pmatrix} p \\ \theta \end{pmatrix},$$

where we have used the shorthand $c_\theta = \cos\theta$ and $s_\theta = \sin\theta$.

In many cases, the angle θ will be very close to 0, and hence we can use the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. Furthermore, if $\dot{\theta}$ is small, we can ignore quadratic and higher terms in $\dot{\theta}$. Substituting these approximations into our equations, we see that we are left with a *linear* state space equation

$$\frac{d}{dt}\begin{pmatrix} p \\ \theta \\ \dot{p} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & m^2l^2g/\mu & -cJ_t/\mu & -\gamma lm/\mu \\ 0 & M_t mgl/\mu & -clm/\mu & -\gamma M_t/\mu \end{pmatrix} \begin{pmatrix} p \\ \theta \\ \dot{p} \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ J_t/\mu \\ lm/\mu \end{pmatrix} u,$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} x,$$

where $\mu = M_t J_t - m^2 l^2$. ∇

Example 3.2 Inverted pendulum

A variation of the previous example is one in which the location of the base p does not need to be controlled. This happens, for example, if we are interested only in stabilizing a rocket's upright orientation without worrying about the location of the

base of the rocket. The dynamics of this simplified system are given by

$$\frac{d}{dt}\begin{pmatrix}\theta\\ \dot\theta\end{pmatrix} = \begin{pmatrix}\dot\theta\\ \dfrac{mgl}{J_t}\sin\theta - \dfrac{\gamma}{J_t}\dot\theta + \dfrac{l}{J_t}u\cos\theta\end{pmatrix}, \qquad y=\theta, \tag{3.11}$$

where γ is the coefficient of rotational friction, $J_t = J + ml^2$ and u is the force applied at the base. This system is referred to as an *inverted pendulum*. ∇

Difference Equations

In some circumstances, it is more natural to describe the evolution of a system at discrete instants of time rather than continuously in time. If we refer to each of these times by an integer $k = 0,1,2,\ldots$, then we can ask how the state of the system changes for each k. Just as in the case of differential equations, we define the state to be the set of variables that summarizes the past of the system for the purpose of predicting its future. Systems described in this manner are referred to as *discrete-time systems*.

The evolution of a discrete-time system can be written in the form

$$x[k+1] = f(x[k],u[k]), \qquad y[k] = h(x[k],u[k]), \tag{3.12}$$

where $x[k] \in \mathbb{R}^n$ is the state of the system at time k (an integer), $u[k] \in \mathbb{R}^p$ is the input and $y[k] \in \mathbb{R}^q$ is the output. As before, f and h are smooth mappings of the appropriate dimension. We call equation (3.12) a *difference equation* since it tells us how $x[k+1]$ differs from $x[k]$. The state $x[k]$ can be either a scalar- or a vector-valued quantity; in the case of the latter we write $x_j[k]$ for the value of the jth state at time k.

Just as in the case of differential equations, it is often the case that the equations are linear in the state and input, in which case we can describe the system by

$$x[k+1] = Ax[k] + Bu[k], \qquad y[k] = Cx[k] + Du[k].$$

As before, we refer to the matrices A, B, C and D as the dynamics matrix, the control matrix, the sensor matrix and the direct term. The solution of a linear difference equation with initial condition $x[0]$ and input $u[0],\ldots,u[T]$ is given by

$$\begin{aligned} x[k] &= A^k x[0] + \sum_{j=0}^{k-1} A^{k-j-1}Bu[j], \\ y[k] &= CA^k x[0] + \sum_{j=0}^{k-1} CA^{k-j-1}Bu[j] + Du[k], \end{aligned} \qquad k > 0. \tag{3.13}$$

Difference equations are also useful as an approximation of differential equations, as we will show later.

Example 3.3 Predator–prey

As an example of a discrete-time system, consider a simple model for a predator–prey system. The predator–prey problem refers to an ecological system in which

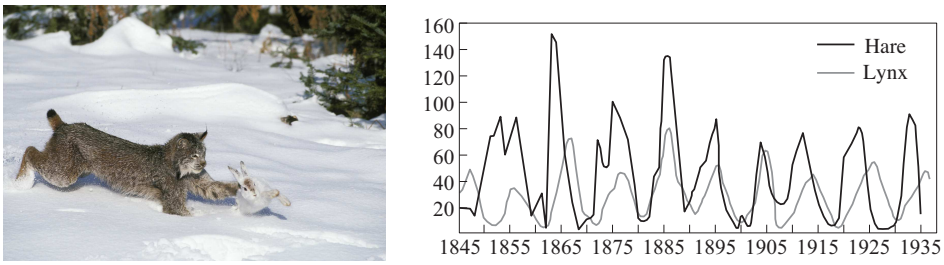


Figure 3.7: Predator versus prey. The photograph on the left shows a Canadian lynx and a snowshoe hare, the lynx's primary prey. The graph on the right shows the populations of hares and lynxes between 1845 and 1935 in a section of the Canadian Rockies [Mac37]. The data were collected on an annual basis over a period of 90 years. (Photograph copyright Tom and Pat Leeson.)

we have two species, one of which feeds on the other. This type of system has been studied for decades and is known to exhibit interesting dynamics. Figure 3.7 shows a historical record taken over 90 years for a population of lynxes versus a population of hares [Mac37]. As can been seen from the graph, the annual records of the populations of each species are oscillatory in nature.

A simple model for this situation can be constructed using a discrete-time model to keep track of the rate of births and deaths of each species. Letting H represent the population of hares and L represent the population of lynxes, we can describe the state in terms of the populations at discrete periods of time. Letting k be the discrete-time index (e.g., the day or month number), we can write

$$\begin{aligned} H[k+1] &= H[k] + b_r(u)H[k] - aL[k]H[k], \\ L[k+1] &= L[k] + cL[k]H[k] - d_f L[k], \end{aligned} \tag{3.14}$$

where $b_r(u)$ is the hare birth rate per unit period and is a function of the food supply u, d_f is the lynx mortality rate and a and c are the interaction coefficients. The interaction term $aL[k]H[k]$ models the rate of predation, which is assumed to be proportional to the rate at which predators and prey meet and is hence given by the product of the population sizes. The interaction term $cL[k]H[k]$ in the lynx dynamics has a similar form and represents the rate of growth of the lynx population. This model makes many simplifying assumptions—such as the fact that hares decrease in number only through predation by lynxes—but it often is sufficient to answer basic questions about the system.

To illustrate the use of this system, we can compute the number of lynxes and hares at each time point from some initial population. This is done by starting with $x[0] = (H_0, L_0)$ and then using equation (3.14) to compute the populations in the following period. By iterating this procedure, we can generate the population over time. The output of this process for a specific choice of parameters and initial conditions is shown in Figure 3.8. While the details of the simulation are different from the experimental data (to be expected given the simplicity of our assumptions), we see qualitatively similar trends and hence we can use the model to help

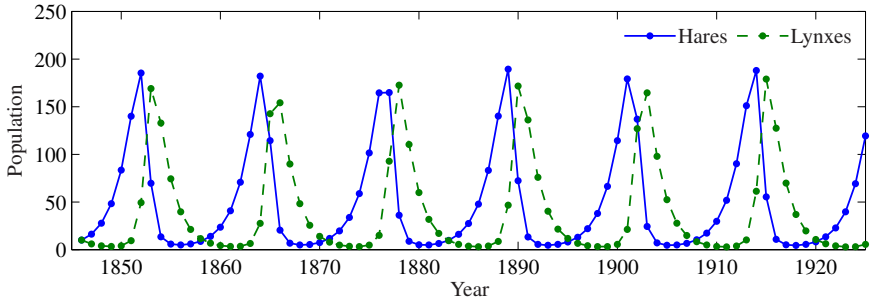


Figure 3.8: Discrete-time simulation of the predator–prey model (3.14). Using the parameters $a = c = 0.014$, $b_r(u) = 0.6$ and $d = 0.7$ in equation (3.14) with daily updates, the period and magnitude of the lynx and hare population cycles approximately match the data in Figure 3.7.

explore the dynamics of the system. ∇

Example 3.4 E-mail server

The IBM Lotus server is a collaborative software system that administers users' e-mail, documents and notes. Client machines interact with end users to provide access to data and applications. The server also handles other administrative tasks. In the early development of the system it was observed that the performance was poor when the central processing unit (CPU) was overloaded because of too many service requests, and mechanisms to control the load were therefore introduced.

The interaction between the client and the server is in the form of remote procedure calls (RPCs). The server maintains a log of statistics of completed requests. The total number of requests being served, called `RIS` (RPCs in server), is also measured. The load on the server is controlled by a parameter called `MaxUsers`, which sets the total number of client connections to the server. This parameter is controlled by the system administrator. The server can be regarded as a dynamical system with `MaxUsers` as the input and `RIS` as the output. The relationship between input and output was first investigated by exploring the steady-state performance and was found to be linear.

In [HDPT04] a dynamic model in the form of a first-order difference equation is used to capture the dynamic behavior of this system. Using system identification techniques, they construct a model of the form

$$y[k+1] = ay[k] + bu[k],$$

where $u = \texttt{MaxUsers} - \overline{\texttt{MaxUsers}}$ and $y = \texttt{RIS} - \overline{\texttt{RIS}}$. The parameters $a = 0.43$ and $b = 0.47$ are parameters that describe the dynamics of the system around the operating point, and $\overline{\texttt{MaxUsers}} = 165$ and $\overline{\texttt{RIS}} = 135$ represent the nominal operating point of the system. The number of requests was averaged over a sampling period of 60 s. ∇

Another application of difference equation is in the implementation of control systems on computers. Early controllers were analog physical systems, which can

be modeled by differential equations. When implementing a controller described by a differential equation using a computer it is necessary to do approximations. A simple way is to approximate derivatives by finite differences, as illustrated by the following example.

Example 3.5 Difference approximation of a PI controller
Consider the PI controller

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau = k_p e(t) + x(t), \qquad x = k_i \int_0^t e(\tau) d\tau,$$

where the controller state is given by the differential equation

$$\frac{dx}{dt} = k_i e(t) \tag{3.15}$$

Assume that the error is measured at regular sampling intervals $t = h, 2h, 3h, \ldots$. Approximating the derivative in equation (3.15) by differences gives

$$\frac{x(kh+h) - x(kh)}{h} = k_i e(kh),$$

and the controller is then given by the difference equation

$$x[k+1] = x[k] + hk_i e[k], \qquad u[k] = k_p e[k] + x[k],$$

where $x[k] = x(kh)$, $e[k] = e(kh)$ and $u[k] = u(kh)$ represent the discrete-time state, error and input sampled at each time interval. This controller is easy to implement on a computer since it consists of just addition and multiplication. ∇

The approximation in the example works well provided that the sampling interval is so short that the variable $e(t)$ changes very little over a sampling interval.

Finite State Machines

A finite state machine is a model in which the states of the system are chosen from a finite list of "modes". The dynamics of a finite state machine are given by transitions between these modes, possibly in response to external signals. We illustrate this concept with a simple example.

Example 3.6 Traffic light controller
Consider a finite state machine model of a traffic light control system, as shown in Figure 3.6. We represent the state of the system in terms of the set of traffic lights that are turned on (either east–west or north–south). In addition, once a light is turned on it should stay that way for a certain minimum time, and then only change when a car comes up to the intersection in the opposite direction. This gives us two states for each direction of the lights: waiting for a car to arrive and waiting for the timer to expire. Thus, we have four states for the system, as shown in Figure 3.6.

The dynamics for the light describe how the system transitions from one state to another. Starting at the left most state, we assume that the lights are set to allow

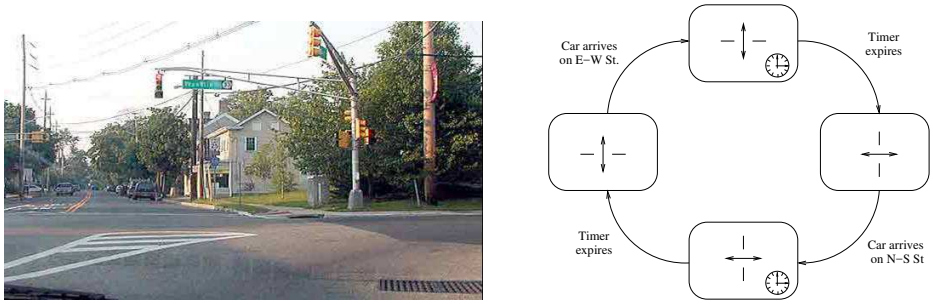


Figure 3.9: A simple model for a traffic light. The diagram on the right is a finite state machine model of the traffic light controller.

traffic in the north–south direction. When a car arrives on the east–west street, we transition to the state at the top of the diagram, where a timer is started. Once the timer reaches the designated amount of time, we transition to the state on the right side of the diagram and turn on the lights in the east–west direction. From here we wait until a car arrives on the north–south street and continue the cycle.

Viewed as a control system, this model has a state space consisting of four discrete states: north–south waiting, north–south countdown, east–west waiting, and east–west countdown. The inputs to the controller consist of the signals that indicate whether a car is present at the roads leading up to the intersection. The outputs from the controller are the signals that change the colors of the traffic light. Finally, the dynamics of the controller are the transition diagram that controls how the states (or modes) of the system change in time. ∇

More formally, a finite state machine can be represented as a finite set of discrete states $\alpha \in \mathbb{Q}_{\text{sys}}$, where $\mathbb{Q}_{\text{sys}}$ is a discrete set. The dynamics of the system are described by transitions between the discrete states, as in the finite state machine described in the previous example. These transitions can depend on external inputs or measurements and can generate output actions on transition into or out of a given state. If we let $\beta \in \mathbb{Q}_{\text{in}}$ represent (discrete) input events (button press, component failure, etc) and $\gamma \in \mathbb{Q}_{\text{out}}$ represent (discrete) output actions (such as turning off of device), then the dynamics of the finite state machine can be written as a guarded command system

$$g_i(\alpha,\beta) \implies \begin{array}{l} \alpha' = r_i(\alpha,\beta), \\ \gamma = a_i(\alpha,\beta), \end{array} \qquad i = 1,\ldots,N. \tag{3.16}$$

Here the function g is a Boolean (true/false) function that depends on the current system mode α and an external input β. If the guard g_i is true then the system transitions from the current state α to a new state α', determined by the rule (transition map) r_i and the external input. The output action γ is similarly dependent on the current state and external input. A guarded transition system can have many different rules, depending on the system state and external input.

The dynamics of a transition system is similar in many ways to the discrete

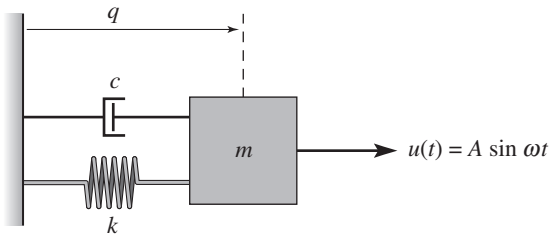


Figure 3.10: A driven spring–mass system with damping. Here we use a linear damping element with coefficient of viscous friction c. The mass is driven with a sinusoidal force of amplitude A.

time dynamics in equation (3.12). The major difference is that the transitions do not necessarily occur at regularly spaced intervals of time. Indeed, there is no strict notion of time in a transition system as we have described it here: it is only the sequence of events that is kept track of (through the evolution of the discrete state).

Specifications for finite transition systems are often written as logical functions describing the conditions that should be imposed on the system. For example, we might wish to say that if a specific sensor is not operating, then the system cannot transition to a mode that requires the use of that sensor. This could be written as the logical formula

$$\alpha \in \{\text{states with sensor } k \text{ not functioning}\} \implies \alpha' \notin \{\text{states requiring sensor } k\}.$$

The formula of the form $p \implies q$ where p and q are Boolean propositions can be written as the logical function $(!p)\,||\,(p\,\&\&\,q)$, which asserts that if proposition p is true then proposition q must be true. In the sensor example, p and q are represented by whether the system mode α is in some set of states.

Finite state machines are very useful for describing logical operations and are often combined with continuous state models (differential or different equations) to create a hybrid system model. The study of hybrid systems is beyond the scope of this text, but excellent references include Lee and Seshia [LS15] and Alur [Alu15].

Simulation and Analysis

State space models can be used to answer many questions. One of the most common, as we have seen in the previous examples, involves predicting the evolution of the system state from a given initial condition. While for simple models this can be done in closed form, more often it is accomplished through computer simulation.

Consider again the damped spring–mass system from Section 3.1, but this time with an external force applied, as shown in Figure 3.10. We wish to predict the motion of the system for a periodic forcing function, with a given initial condition, and determine the amplitude, frequency and decay rate of the resulting motion.

We choose to model the system with a linear ordinary differential equation.

Using Hooke's law to model the spring and assuming that the damper exerts a force that is proportional to the velocity of the system, we have

$$m\ddot{q} + c\dot{q} + kq = u, \tag{3.17}$$

where m is the mass, q is the displacement of the mass, c is the coefficient of viscous friction, k is the spring constant and u is the applied force. In state space form, using $x = (q, \dot{q})$ as the state and choosing $y = q$ as the output, we have

$$\frac{dx}{dt} = \begin{pmatrix} x_2 \\ -\dfrac{c}{m}x_2 - \dfrac{k}{m}x_1 + \dfrac{u}{m} \end{pmatrix}, \qquad y = x_1.$$

We see that this is a linear second-order differential equation with one input u and one output y.

We now wish to compute the response of the system to an input of the form $u = A\sin\omega t$. Although it is possible to solve for the response analytically, we instead make use of a computational approach that does not rely on the specific form of this system. Consider the general state space system

$$\frac{dx}{dt} = f(x,u).$$

Given the state x at time t, we can approximate the value of the state at a short time $h > 0$ later by assuming that the rate of change $f(x,u)$ is constant over the interval t to $t+h$. This gives

$$x(t+h) = x(t) + hf(x(t),u(t)). \tag{3.18}$$

Iterating this equation, we can thus solve for x as a function of time. This approximation is known as Euler integration and is in fact a difference equation if we let h represent the time increment and write $x[k] = x(kh)$, as we saw in Example 3.5. Although modern simulation tools such as MATLAB and Mathematica use more accurate methods than Euler integration, they still have some of the same basic trade-offs.

Returning to our specific example, Figure 3.11 shows the results of computing $x(t)$ using equation (3.18), along with the analytical computation. We see that as h gets smaller, the computed solution converges to the exact solution. The form of the solution is also worth noticing: after an initial transient, the system settles into a periodic motion. The portion of the response after the transient is called the *steady-state response* to the input.

In addition to generating simulations, models can also be used to answer other types of questions. Two that are central to the methods described in this text concern the stability of an equilibrium point and the input/output frequency response. We illustrate these two computations through the examples below and return to the general computations in later chapters.

Returning to the damped spring–mass system, the equations of motion with no

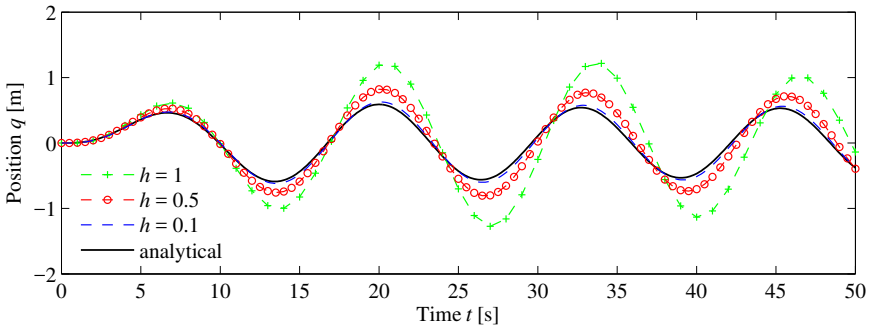


Figure 3.11: Simulation of the forced spring–mass system with different simulation time constants. The solid line represents the analytical solution. The dashed lines represent the approximate solution via the method of Euler integration, using decreasing step sizes.

input forcing are given by

$$\frac{dx}{dt} = \begin{pmatrix} x_2 \\ -\frac{c}{m}x_2 - \frac{k}{m}x_1 \end{pmatrix}, \tag{3.19}$$

where x_1 is the position of the mass (relative to the rest position) and x_2 is its velocity. We wish to show that if the initial state of the system is away from the rest position, the system will return to the rest position eventually (we will later define this situation to mean that the rest position is *asymptotically stable*). While we could heuristically show this by simulating many, many initial conditions, we seek instead to prove that this is true for *any* initial condition.

To do so, we construct a function $V : \mathbb{R}^n \to \mathbb{R}$ that maps the system state to a positive real number. For mechanical systems, a convenient choice is the energy of the system,

$$V(x) = \frac{1}{2}kx_1^2 + \frac{1}{2}mx_2^2. \tag{3.20}$$

If we look at the time derivative of the energy function, we see that

$$\frac{dV}{dt} = kx_1\dot{x}_1 + mx_2\dot{x}_2 = kx_1x_2 + mx_2(-\frac{c}{m}x_2 - \frac{k}{m}x_1) = -cx_2^2,$$

which is always either negative or zero. Hence $V(x(t))$ is never increasing and, using a bit of analysis that we will see formally later, the individual states must remain bounded.

If we wish to show that the states eventually return to the origin, we must use a slightly more detailed analysis. Intuitively, we can reason as follows: suppose that for some period of time, $V(x(t))$ stops decreasing. Then it must be true that $\dot{V}(x(t)) = 0$, which in turn implies that $x_2(t) = 0$ for that same period. In that case, $\dot{x}_2(t) = 0$, and we can substitute into the second line of equation (3.19) to obtain

$$0 = \dot{x}_2 = -\frac{c}{m}x_2 - \frac{k}{m}x_1 = -\frac{k}{m}x_1.$$

Thus we must have that x_1 also equals zero, and so the only time that $V(x(t))$ can stop decreasing is if the state is at the origin (and hence this system is at its rest position). Since we know that $V(x(t))$ is never increasing (because $\dot{V} \leq 0$), we therefore conclude that the origin is stable (for *any* initial condition).

This type of analysis, called Lyapunov stability analysis, is considered in detail in Chapter 5. It shows some of the power of using models for the analysis of system properties.

Another type of analysis that we can perform with models is to compute the output of a system to a sinusoidal input, known as the *frequency response*. We again consider the spring–mass system, but this time keeping the input and leaving the system in its original form:

$$m\ddot{q} + c\dot{q} + kq = u. \tag{3.21}$$

We wish to understand how the system responds to a sinusoidal input of the form

$$u(t) = A \sin \omega t.$$

We will see how to do this analytically in Chapter 7, but for now we make use of simulations to compute the answer.

We first begin with the observation that if $q(t)$ is the solution to equation (3.21) with input $u(t)$, then applying an input $2u(t)$ will give a solution $2q(t)$ (this is easily verified by substitution). Hence it suffices to look at an input with unit magnitude, $A = 1$. A second observation, which we will prove in Chapter 6, is that the long-term response of the system to a sinusoidal input is itself a sinusoid at the same frequency, and so the output has the form

$$q(t) = g(\omega) \sin(\omega t + \varphi(\omega)),$$

where $g(\omega)$ is called the *gain* of the system and $\varphi(\omega)$ is called the *phase* (or phase offset).

To compute the frequency response numerically, we can simulate the system at a set of frequencies $\omega_1, \ldots, \omega_N$ and plot the gain and phase at each of these frequencies. An example of this type of computation is shown in Figure 3.12. For linear systems the frequency response does not depend on the amplitude A of the input signal. Frequency response can also be applied to nonlinear systems but the gain and phase then depend on the A.

3.3 Modeling Methodology

To deal with large, complex systems, it is useful to have different representations of the system that capture the essential features and hide irrelevant details. In all branches of science and engineering it is common practice to use some graphical description of systems, called *schematic diagrams*. They can range from stylistic pictures to drastically simplified standard symbols. These pictures make it possible to get an overall view of the system and to identify the individual components.

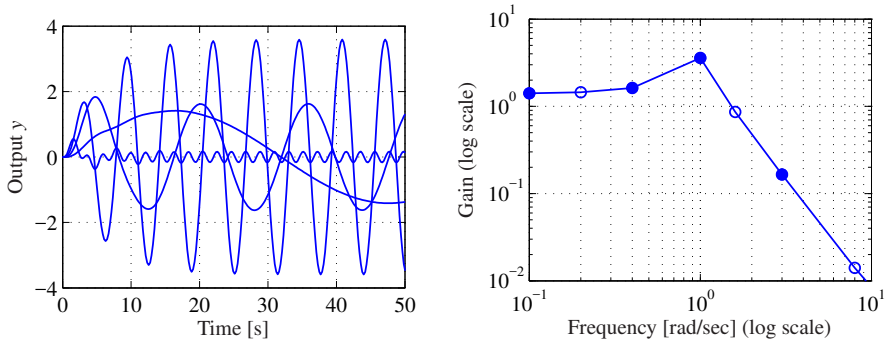
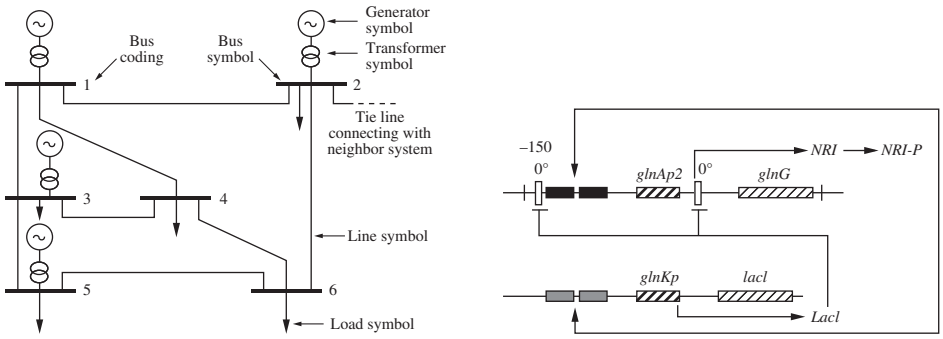
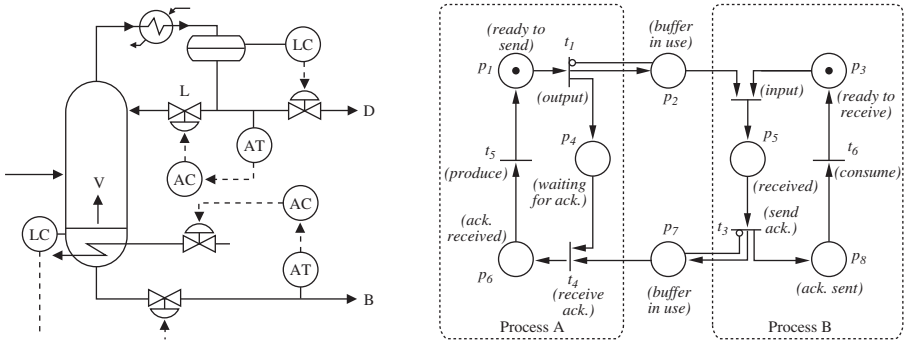


Figure 3.12: A frequency response (gain only) computed by measuring the response of individual sinusoids. The figure on the left shows the response of the system as a function of time to a number of different unit magnitude inputs (at different frequencies). The figure on the right shows this same data in a different way, with the magnitude of the response plotted as a function of the input frequency. The filled circles correspond to the particular frequencies shown in the time responses.



(a) Power electronics (b) Cell biology



(c) Process control (d) Networking

Figure 3.13: Schematic diagrams for different disciplines. Each diagram is used to illustrate the dynamics of a control system: (a) electrical schematics for a power system [Kun93], (b) a biological circuit diagram for a synthetic clock circuit [ASMN03], (c) a process diagram for a distillation column [SEM04] and (d) a Petri net description of a communication protocol.

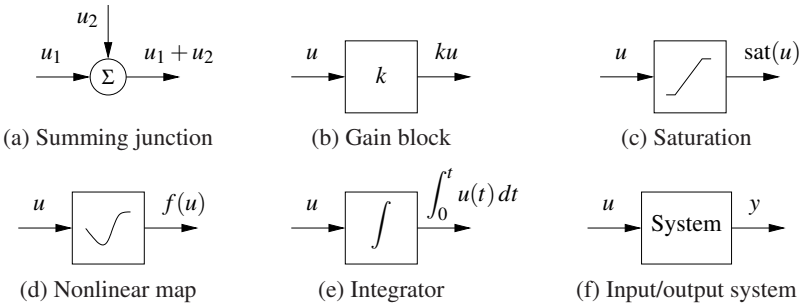


Figure 3.14: Standard block diagram elements. The arrows indicate the the inputs and outputs of each element, with the mathematical operation corresponding to the blocked labeled at the output. The system block (f) represents the full input/output response of a dynamical system.

Examples of such diagrams are shown in Figure 3.13. Schematic diagrams are useful because they give an overall picture of a system, showing different subprocesses and their interconnection and indicating variables that can be manipulated and signals that can be measured.

Block Diagrams

A special graphical representation called a *block diagram* has been developed in control engineering. The purpose of a block diagram is to emphasize the information flow and to hide details of the system. In a block diagram, different process elements are shown as boxes, and each box has inputs denoted by lines with arrows pointing toward the box and outputs denoted by lines with arrows going out of the box. The inputs denote the variables that influence a process, and the outputs denote the signals that we are interested in or signals that influence other subsystems. Block diagrams can also be organized in hierarchies, where individual blocks may themselves contain more detailed block diagrams.

Figure 3.14 shows some of the notation that we use for block diagrams. Signals are represented as lines, with arrows to indicate inputs and outputs. The first diagram is the representation for a summation of two signals. An input/output response is represented as a rectangle with the system name (or mathematical description) in the block. Two special cases are a proportional gain, which scales the input by a multiplicative factor, and an integrator, which outputs the integral of the input signal.

Figure 3.15 illustrates the use of a block diagram, in this case for modeling the flight response of a fly. The flight dynamics of an insect are incredibly intricate, involving careful coordination of the muscles within the fly to maintain stable flight in response to external stimuli. One known characteristic of flies is their ability to fly upwind by making use of the optical flow in their compound eyes as a feedback mechanism. Roughly speaking, the fly controls its orientation so that the point of contraction of the visual field is centered in its visual field.

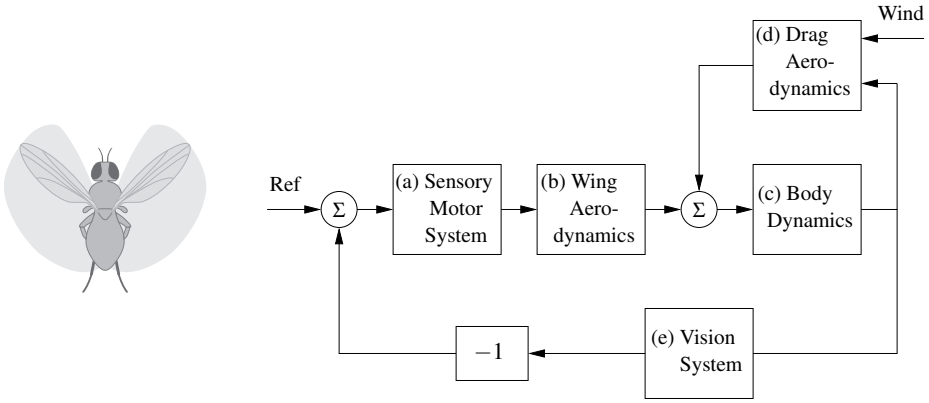


Figure 3.15: A block diagram representation of the flight control system for an insect flying against the wind. The mechanical portion of the model consists of the rigid-body dynamics of the fly, the drag due to flying through the air and the forces generated by the wings. The motion of the body causes the visual environment of the fly to change, and this information is then used to control the motion of the wings (through the sensory motor system), closing the loop.

To understand this complex behavior, we can decompose the overall dynamics of the system into a series of interconnected subsystems (or *blocks*). Referring to Figure 3.15, we can model the insect navigation system through an interconnection of five blocks. The sensory motor system (a) takes the information from the visual system (e) and generates muscle commands that attempt to steer the fly so that the point of contraction is centered. These muscle commands are converted into forces through the flapping of the wings (b) and the resulting aerodynamic forces that are produced. The forces from the wings are combined with the drag on the fly (d) to produce a net force on the body of the fly. The wind velocity enters through the drag aerodynamics. Finally, the body dynamics (c) describe how the fly translates and rotates as a function of the net forces that are applied to it. The insect position, speed and orientation are fed back to the drag aerodynamics and vision system blocks as inputs.

Each of the blocks in the diagram can itself be a complicated subsystem. For example, the visual system of a fruit fly consists of two complicated compound eyes (with about 700 elements per eye), and the sensory motor system has about 200,000 neurons that are used to process information. A more detailed block diagram of the insect flight control system would show the interconnections between these elements, but here we have used one block to represent how the motion of the fly affects the output of the visual system, and a second block to represent how the visual field is processed by the fly's brain to generate muscle commands. The choice of the level of detail of the blocks and what elements to separate into different blocks often depends on experience and on the questions that one wants to answer using the model. One of the powerful features of block diagrams is their ability to hide information about the details of a system that may not be needed to gain an understanding of the essential dynamics of the system.

Modeling from Experiments

Since control systems are provided with sensors and actuators, it is also possible to obtain models of system dynamics from experiments on the process. The models are restricted to input/output models since only these signals are accessible to experiments, but modeling from experiments can also be combined with modeling from physics through the use of feedback and interconnection.

A simple way to determine a system's dynamics is to observe the response to a step change in the control signal. Such an experiment begins by setting the control signal to a constant value; then when steady state is established, the control signal is changed quickly to a new level and the output is observed. The experiment gives the step response of the system, and the shape of the response gives useful information about the dynamics. It immediately gives an indication of the response time, and it tells if the system is oscillatory or if the response is monotone.

Example 3.7 Spring–mass system
The dynamics of the spring–mass system in Section 3.1 are given by

$$m\ddot{q} + c\dot{q} + kq = u. \tag{3.22}$$

We wish to determine the constants m, c and k by measuring the response of the system to a step input of magnitude F_0.

We will show in Chapter 7 that when $c^2 < 4km$, the step response for this system from the rest configuration is given by

$$q(t) = \frac{F_0}{k}\left(1 - \frac{1}{\omega_d}\sqrt{\frac{k}{m}}\exp\left(-\frac{ct}{2m}\right)\sin(\omega_d t + \varphi)\right),$$

$$\omega_d = \frac{\sqrt{4km - c^2}}{2m}, \qquad \varphi = \tan^{-1}\left(\frac{\sqrt{4km - c^2}}{c}\right).$$

From the form of the solution, we see that the shape of the step response is determined by the parameters of the system. Hence, by measuring certain features of the step response we can determine the parameter values.

Figure 3.16 shows the response of the system to a step of magnitude $F_0 = 20$ N, along with some measurements. We start by noting that the steady-state position of the mass (after the oscillations die down) is a function of the spring constant k:

$$q(\infty) = \frac{F_0}{k}, \tag{3.23}$$

where F_0 is the magnitude of the applied force ($F_0 = 1$ for a unit step input). The parameter $1/k$ is called the *gain* of the system. The period of the oscillation can be measured between two peaks and must satisfy

$$\frac{2\pi}{T} = \frac{\sqrt{4km - c^2}}{2m}. \tag{3.24}$$

Finally, the rate of decay of the oscillations is given by the exponential factor in

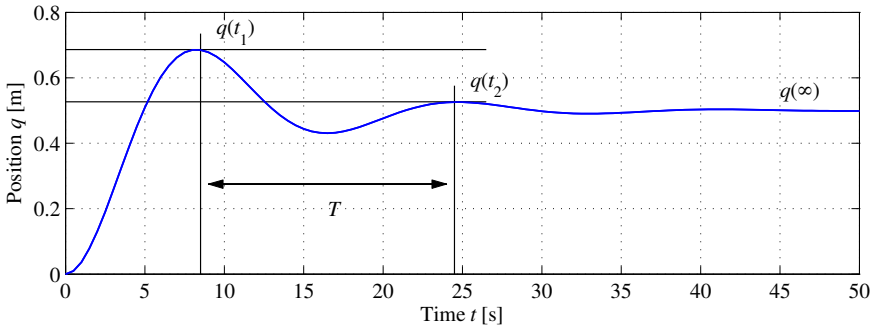


Figure 3.16: Step response for a spring–mass system. The magnitude of the step input is $F_0 = 20$ N. The period of oscillation T is determined by looking at the time between two subsequent local maxima in the response. The period combined with the steady-state value $q(\infty)$ and the relative decrease between local maxima can be used to estimate the parameters in a model of the system.

the solution. Measuring the amount of decay between two peaks, we have

$$\log\left(q(t_1) - \frac{F_0}{k}\right) - \log\left(q(t_2) - \frac{F_0}{k}\right) = \frac{c}{2m}(t_2 - t_1). \tag{3.25}$$

Using this set of three equations, we can solve for the parameters and determine that for the step response in Figure 3.16 we have $m \approx 250$ kg, $c \approx 60$ N s/m and $k \approx 40$ N/m. ∇

Modeling from experiments can also be done using many other signals. Sinusoidal signals are commonly used (particularly for systems with fast dynamics) and precise measurements can be obtained by exploiting correlation techniques. An indication of nonlinearities can be obtained by repeating experiments with input signals having different amplitudes. Modeling based on sinusoidal signals is very time consuming for systems with slow dynamics. In such situations it is advantageous to used signals that switch between two different levels. There is a whole subfield of control called *system identification* that deals with experimental determination of models. Questions like optimal inputs, experiments in open and closed loop, model accuracy and fundamental limitations are dealt with extensively.

Normalization and Scaling

When deriving a model, it is often useful to introduce dimension-free variables. Such a procedure can often simplify the equations for a system by reducing the number of parameters. It can also reveal interesting properties of the model. It is also useful to normalize variables by scaling to improve numerics and allow faster and more accurate simulations.

The procedure of scaling is straightforward in principle: choose units for each independent variable and introduce new variables by dividing the variables by the chosen normalization unit. We illustrate the procedure with two examples.

Example 3.8 Spring–mass system
Consider again the spring–mass system introduced earlier. Neglecting the damping, the system is described by

$$m\ddot{q} + kq = u.$$

The model has two parameters m and k. To normalize the model we introduce dimension-free variables $x = q/l$ and $\tau = \omega_0 t$, where $\omega_0 = \sqrt{k/m}$ and l is the chosen length scale. We scale force by $ml\omega_0^2$ and introduce $v = u/(ml\omega_0^2)$. The scaled equation then becomes

$$\frac{d^2x}{d\tau^2} = \frac{d^2q/l}{d(\omega_0 t)^2} = \frac{1}{ml\omega_0^2}(-kq+u) = -x+v,$$

which is the normalized undamped spring–mass system. Notice that the normalized model has no parameters, while the original model had two parameters m and k. Introducing the scaled, dimension-free state variables $z_1 = x = q/l$ and $z_2 = dx/d\tau = \dot{q}/(l\omega_0)$, the model can be written as

$$\frac{d}{d\tau}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0 \\ v \end{pmatrix}.$$

This simple linear equation describes the dynamics of any spring–mass system, independent of the particular parameters, and hence gives us insight into the fundamental dynamics of this oscillatory system. To recover the physical frequency of oscillation or its magnitude, we must invert the scaling we have applied. ∇

Example 3.9 Balance system
Consider the balance system described in Example 3.1. Neglecting damping by putting $c = 0$ and $\gamma = 0$ in equation (3.10), the model can be written as

$$(M+m)\frac{d^2p}{dt^2} - ml\cos\theta\frac{d^2\theta}{dt^2} + ml\sin\theta\left(\frac{d\theta}{dt}\right)^2 = F,$$
$$-ml\cos\theta\frac{d^2p}{dt^2} + (J+ml^2)\frac{d^2\theta}{dt^2} - mgl\sin\theta = 0.$$

Let $\omega_0 = \sqrt{mgl/(J+ml^2)}$, choose the length scale as l, let the time scale be $1/\omega_0$, choose the force scale as $(M+m)l\omega_0^2$ and introduce the scaled variables $\tau = \omega_0 t$, $x = p/l$ and $u = F/((M+m)l\omega_0^2)$. The equations then become

$$\frac{d^2x}{d\tau^2} - \alpha\cos\theta\frac{d^2\theta}{d\tau^2} + \alpha\sin\theta\left(\frac{d\theta}{d\tau}\right)^2 = u, \quad -\beta\cos\theta\frac{d^2x}{d\tau^2} + \frac{d^2\theta}{d\tau^2} - \sin\theta = 0,$$

where $\alpha = m/(M+m)$ and $\beta = ml^2/(J+ml^2)$. Notice that the original model has five parameters m, M, J, l and g but the normalized model has only two parameters α and β. If $M \gg m$ and $ml^2 \gg J$, we get $\alpha \approx 0$ and $\beta \approx 1$ and the model can be approximated by

$$\frac{d^2x}{d\tau^2} = u, \qquad \frac{d^2\theta}{d\tau^2} - \sin\theta = u\cos\theta.$$

The model can be interpreted as a mass combined with an inverted pendulum driven by the same input. ∇

For large systems scaling is not so easy: there are many choices and good selection of variables and normalization units require good understanding of the physics of the system and the numerical methods that will be used for analysis, scaling of large systems is therefor still an art.

3.4 Modeling Examples

In this section we introduce additional examples that illustrate some of the different types of systems for which one can develop differential equation and difference equation models. These examples are specifically chosen from a range of different fields to highlight the broad variety of systems to which feedback and control concepts can be applied. A more detailed set of applications that serve as running examples throughout the text are given in Chapter 4.

Motion Control Systems

Motion control systems involve the use of computation and feedback to control the movement of a mechanical system. Motion control systems range from nanopositioning systems (atomic force microscopes, adaptive optics), to control systems for the read/write heads in a disk drive of a DVD player, to manufacturing systems (transfer machines and industrial robots), to automotive control systems (antilock brakes, suspension control, traction control), to air and space flight control systems (airplanes, satellites, rockets and planetary rovers).

Example 3.10 Vehicle steering
A common problem in motion control is to control the trajectory of a vehicle through an actuator that causes a change in the orientation. A steering wheel on an automobile and the front wheel of a bicycle are two examples, but similar dynamics occur in the steering of ships or control of the pitch dynamics of an aircraft. In many cases, we can understand the basic behavior of these systems through the use of a simple model that captures the basic kinematics of the system.

Consider a vehicle with two wheels as shown in Figure 3.17. For the purpose of steering we are interested in a model that describes how the velocity of the vehicle depends on the steering angle δ. To be specific, let b be the wheel base and consider the velocity v at the center of mass, a distance a from the rear wheel, as shown in Figure 3.17. Let x and y be the coordinates of the center of mass, θ the heading angle and α the angle between the velocity vector v and the centerline of the vehicle. The point O is at the intersection of the normals to the front and rear wheels. The distance from O to the center of mass is r_a.

Assuming no slipping of the wheels the motion is a rotation around the point O in the figure. Since $b = r_a \tan\delta$ and $a = r_a \tan\alpha$, it follows that $\tan\alpha = (a/b)\tan\delta$

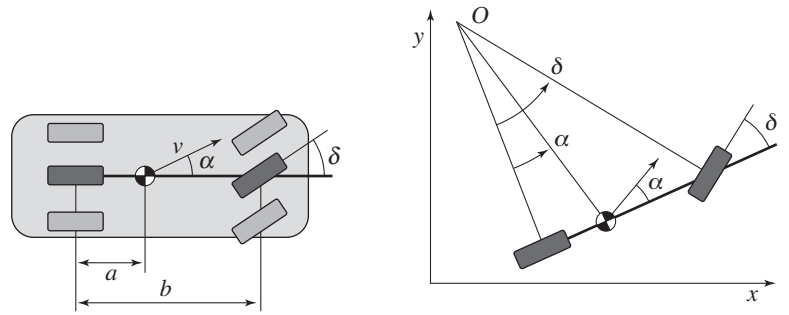


Figure 3.17: Vehicle steering dynamics. The left figure shows an overhead view of a vehicle with four wheels. The wheel base is b and the center of mass at a distance a forward of the rear wheels. By approximating the motion of the front and rear pairs of wheels by a single front wheel and a single rear wheel, we obtain an abstraction called the *bicycle model*, shown on the right. The steering angle is δ and the velocity at the center of mass has the angle α relative the length axis of the vehicle. The position of the vehicle is given by (x,y) and the orientation (heading) by θ.

and we get the following relation between α and the steering angle δ:

$$\alpha = \arctan\left(\frac{a\tan\delta}{b}\right). \tag{3.26}$$

Assume that the wheels are rolling without slip and that the velocity of the rear wheel is v_0. The vehicle speed at its center of mass is $v = v_0/\cos\alpha$, and we find that the motion of this point is given by

$$\begin{aligned}\frac{dx}{dt} &= v\cos(\alpha+\theta) = v_0\frac{\cos(\alpha+\theta)}{\cos\alpha},\\ \frac{dy}{dt} &= v\sin(\alpha+\theta) = v_0\frac{\sin(\alpha+\theta)}{\cos\alpha}.\end{aligned} \tag{3.27}$$

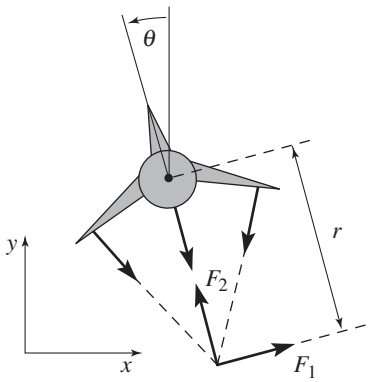
To see how the angle θ is influenced by the steering angle, we observe from Figure 3.17 that the vehicle rotates with the angular velocity v_0/r_a around the point O. Hence

$$\frac{d\theta}{dt} = \frac{v_0}{r_a} = \frac{v_0}{b}\tan\delta. \tag{3.28}$$

Equations (3.26)–(3.28) can be used to model an automobile under the assumptions that there is no slip between the wheels and the road and that the two front wheels can be approximated by a single wheel at the center of the car. This model is often called the *bicycle model*. The assumption of no slip can be relaxed by adding an extra state variable, giving a more realistic model. Such a model also describes the steering dynamics of ships as well as the pitch dynamics of aircraft and missiles. It is also possible to choose coordinates so that the reference point is at the rear wheels (corresponding to setting $\alpha = 0$), a model often referred to as the *Dubins car* [Dub57] or the bicycle model.

Figure 3.17 represents the situation when the vehicle moves forward and has

(a) Harrier "jump jet"



(b) Simplified model

Figure 3.18: Vectored thrust aircraft. The Harrier AV-8B military aircraft (a) redirects its engine thrust downward so that it can "hover" above the ground. Some air from the engine is diverted to the wing tips to be used for maneuvering. As shown in (b), the net thrust on the aircraft can be decomposed into a horizontal force F_1 and a vertical force F_2 acting at a distance r from the center of mass.

front-wheel steering. The figure shows that the model also applies to rear wheel steering if the sign of the velocity is reversed. ∇

Example 3.11 Vectored thrust aircraft

Consider the motion of vectored thrust aircraft, such as the Harrier "jump jet" shown Figure 3.18a. The Harrier is capable of vertical takeoff by redirecting its thrust downward and through the use of smaller maneuvering thrusters located on its wings. A simplified model of the Harrier is shown in Figure 3.18b, where we focus on the motion of the vehicle in a vertical plane through the wings of the aircraft. We resolve the forces generated by the main downward thruster and the maneuvering thrusters as a pair of forces F_1 and F_2 acting at a distance r below the aircraft (determined by the geometry of the thrusters).

Let (x, y, θ) denote the position and orientation of the center of mass of the aircraft. Let m be the mass of the vehicle, J the moment of inertia, g the gravitational constant and c the damping coefficient. Then the equations of motion for the vehicle are given by

$$\begin{aligned} m\ddot{x} &= F_1 \cos\theta - F_2 \sin\theta - c\dot{x}, \\ m\ddot{y} &= F_1 \sin\theta + F_2 \cos\theta - mg - c\dot{y}, \\ J\ddot{\theta} &= rF_1. \end{aligned} \tag{3.29}$$

It is convenient to redefine the inputs so that the origin is an equilibrium point of the system with zero input. Letting $u_1 = F_1$ and $u_2 = F_2 - mg$, the equations become

$$\begin{aligned} m\ddot{x} &= -mg\sin\theta - c\dot{x} + u_1\cos\theta - u_2\sin\theta, \\ m\ddot{y} &= mg(\cos\theta - 1) - c\dot{y} + u_1\sin\theta + u_2\cos\theta, \\ J\ddot{\theta} &= ru_1. \end{aligned} \tag{3.30}$$

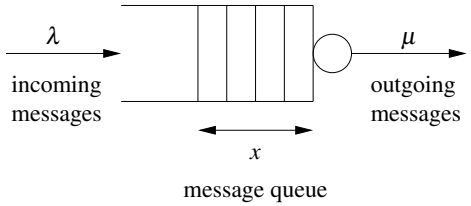


Figure 3.19: Schematic diagram of a queuing system. Messages arrive at rate λ and are stored in a queue. Messages are processed and removed from the queue at rate μ. The average length of the queue is given by $x \in \mathbb{R}$.

These equations describe the motion of the vehicle as a set of three coupled second-order differential equations. ∇

Information Systems

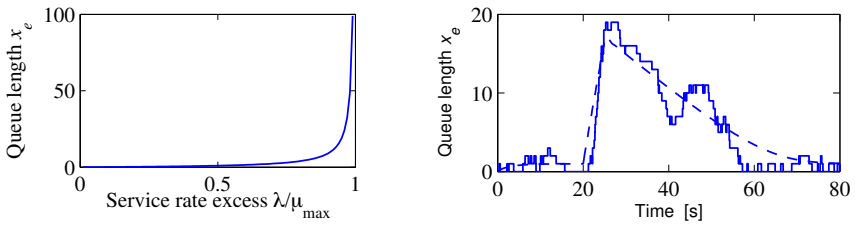
Information systems range from communication systems like the Internet to software systems that manipulate data or manage enterprise-wide resources. Feedback is present in all these systems, and designing strategies for routing, flow control and buffer management is a typical problem. Many results in queuing theory emerged from design of telecommunication systems and later from development of the Internet and computer communication systems [BG87, Kle75, Sch87]. Management of queues to avoid congestion is a central problem and we will therefore start by discussing the modeling of queuing systems.

Example 3.12 Queuing systems
A schematic picture of a simple queue is shown in Figure 3.19. Requests arrive and are then queued and processed. There can be large variations in arrival rates and service rates, and the queue length builds up when the arrival rate is larger than the service rate. When the queue becomes too large, service is denied using an admission control policy.

The system can be modeled in many different ways. One way is to model each incoming request, which leads to an event-based model where the state is an integer that represents the queue length. The queue changes when a request arrives or a request is serviced. The statistics of arrival and servicing are typically modeled as random processes. In many cases it is possible to determine statistics of quantities like queue length and service time, but the computations can be quite complicated.

A significant simplification can be obtained by approximating the discrete queue length by a continuous variable. Instead of keeping track of each request we instead view service and requests as continuous flows. The model obtained is called a *flow model* because of the analogy with fluid dynamics where motion of molecules are replace by continuous flows. Hence, if the queue length x is a continuous variable and the arrivals and services are flows with rates λ and μ, the system can be modeled by the first-order differential equation

$$\frac{dx}{dt} = \lambda - \mu = \lambda - \mu_{\max} f(x), \quad x \geq 0, \tag{3.31}$$



(a) Steady-state queue length (b) Overload condition

Figure 3.20: Queuing dynamics. (a) The steady-state queue length as a function of $\lambda/\mu_{\max}$. (b) The behavior of the queue length when there is a temporary overload in the system. The solid line shows a realization of an event-based simulation, and the dashed line shows the behavior of the flow model (3.32). The maximum service rate is $\mu_{\max} = 1$, and the arrival rate starts at $\lambda = 0.5$. The arrival rate is increased to $\lambda = 4$ at time 20, and it returns to $\lambda = 0.5$ at time 25.

proposed by Agnew [Agn76]. The service rate μ depends on the queue length; if there are no capacity restrictions we have $\mu = x/T$ where T is the time it takes to serve one customer. The service rate thus increases linearly with the queue length. In reality the growth will be slower because longer queues require more resources, and the service rate has an upper limit $\mu_{\max}$. These effects are captured by by modeling the service rate as $\mu_{\max} f(x)$ in equation (3.32). The function $f(x)$ is monotone, approximately linear for small x and $f(\infty) = 1$.

For a particular queue, the function can be determined empirically by measuring the queue length for different arrival and service rates. A simple choice is $f(x) = x/(1+x)$, which gives the model

$$\frac{dx}{dt} = \lambda - \mu_{\max}\frac{x}{x+1}. \tag{3.32}$$

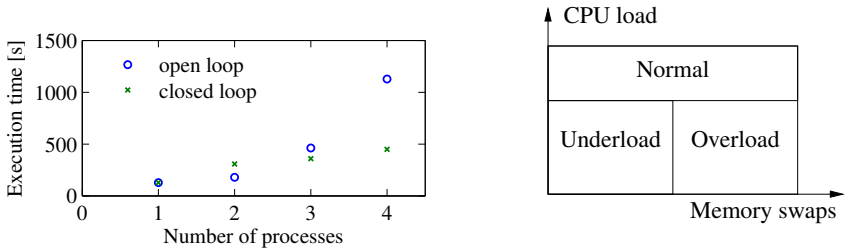
It was shown by Tipper [TS90], that if arrival and service processes are Poisson processes, then average queue length is given by equation (3.32). Furthermore the equation equation (3.32) is a good approximation even for short queue lengths.

To explore the properties of the model (3.32) we will first investigate the equilibrium value of the queue length when the arrival rate λ is constant. Setting the derivative dx/dt to zero in equation (3.32) and solving for x, we find that the queue length x approaches the steady-state value

$$x_e = \frac{\lambda}{\mu_{\max} - \lambda}. \tag{3.33}$$

Figure 3.20a shows the steady-state queue length as a function of $\lambda/\mu_{\max}$, the effective service rate excess. Notice that the queue length increases rapidly as λ approaches $\mu_{\max}$. To have a queue length less than 20 requires $\lambda/\mu_{\max} < 0.95$. The average time to service a request can be shown to be $T_s = (x+1)/\mu_{\max}$, and it increases dramatically as λ approaches $\mu_{\max}$.

Figure 3.20b illustrates the behavior of the server in a typical overload situation. The figure shows that the queue builds up quickly and clears very slowly. Since the



(a) System performance (b) System state

Figure 3.21: Illustration of feedback in the virtual memory system of the IBM/370. (a) The effect of feedback on execution times in a simulation, following [BG68]. Results with no feedback are shown with ○, and results with feedback with ×. Notice the dramatic decrease in execution time for the system with feedback. (b) How the three states are obtained based on process measurements.

response time is proportional to queue length, it means that the quality of service is poor for a long period after an overload. This behavior is called the *rush-hour effect* and has been observed in web servers and many other queuing systems such as automobile traffic.

The dashed line in Figure 3.20b shows the behavior of the flow model, which describes the average queue length. The simple model captures behavior qualitatively, but there are variations from sample to sample when the queue length is short. ∇

Many complex systems use discrete control actions. Such systems can be modeled by characterizing the situations that correspond to each control action, as illustrated in the following example.

Example 3.13 Virtual memory paging control

An early example of the use of feedback in computer systems was applied in the operating system OS/VS for the IBM 370 [BG68, Cro75]. The system used virtual memory, which allows programs to address more memory than is physically available as fast memory. Data in current fast memory (random access memory, RAM) is accessed directly, but data that resides in slower memory (disk) is automatically loaded into fast memory. The system is implemented in such a way that it appears to the programmer as a single large section of memory. The system performed very well in many situations, but very long execution times were encountered in overload situations, as shown by the open circles in Figure 3.21a. The difficulty was resolved with a simple discrete feedback system. The load of the central processing unit (CPU) was measured together with the number of page swaps between fast memory and slow memory. The operating region was classified as being in one of three states: normal, underload or overload. The normal state is characterized by high CPU activity, the underload state is characterized by low CPU activity and few page replacements, the overload state has moderate to low CPU load but many page replacements; see Figure 3.21b. The boundaries between the regions and the time for measuring the load were determined from simulations using typ-

ical loads. The control strategy was to do nothing in the normal load condition, to exclude a process from memory in the overload condition and to allow a new process or a previously excluded process in the underload condition. The crosses in Figure 3.21a show the effectiveness of the simple feedback system in simulated loads. Similar principles are used in many other situations, e.g., in fast, on-chip cache memory.

∇

Example 3.14 Consensus protocols in sensor networks

Sensor networks are used in a variety of applications where we want to collect and aggregate information over a region of space using multiple sensors that are connected together via a communications network. Examples include monitoring environmental conditions in a geographical area (or inside a building), monitoring the movement of animals or vehicles and monitoring the resource loading across a group of computers. In many sensor networks the computational resources are distributed along with the sensors, and it can be important for the set of distributed agents to reach a consensus about a certain property, such as the average temperature in a region or the average computational load among a set of computers.

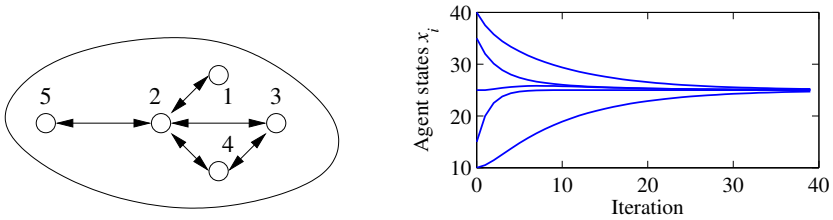
To illustrate how such a consensus might be achieved, we consider the problem of computing the average value of a set of numbers that are locally available to the individual agents. We wish to design a "protocol" (algorithm) such that all agents will agree on the average value. We consider the case in which all agents cannot necessarily communicate with each other directly, although we will assume that the communications network is connected (meaning that no two groups of agents are completely isolated from each other). Figure 3.22a shows a simple situation of this type.

We model the connectivity of the sensor network using a graph, with nodes corresponding to the sensors and edges corresponding to the existence of a direct communications link between two nodes. For any such graph, we can build an *adjacency matrix*, where each row and column of the matrix corresponds to a node and a 1 in the respective row and column indicates that the two nodes are connected. For the network shown in Figure 3.22a, the corresponding adjacency matrix is

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

We use the notation $\mathcal{N}_i$ to represent the set of neighbors of a node i. For example, in the network shown in Figure 3.22a $\mathcal{N}_2 = \{1,3,4,5\}$ and $\mathcal{N}_3 = \{2,4\}$.

To solve the consensus problem, let x_i be the state of the ith sensor, corresponding to that sensor's estimate of the average value that we are trying to compute. We initialize the state to the value of the quantity measured by the individual sensor.



(a) Sensor network (b) Consensus convergence

Figure 3.22: Consensus protocols for sensor networks. (a) A simple sensor network with five nodes. In this network, node 1 communicates with node 2 and node 2 communicates with nodes 1, 3, 4, 5, etc. (b) A simulation demonstrating the convergence of the consensus protocol (3.34) to the average value of the initial conditions.

The consensus protocol (algorithm) can now be realized as a local update law

$$x_i[k+1] = x_i[k] + \gamma \sum_{j \in \mathcal{N}_i} (x_j[k] - x_i[k]). \tag{3.34}$$

This protocol attempts to compute the average by updating the local state of each agent based on the value of its neighbors. The combined dynamics of all agents can be written in the form

$$x[k+1] = x[k] - \gamma(D-A)x[k], \tag{3.35}$$

where A is the adjacency matrix and D is a diagonal matrix with entries corresponding to the number of neighbors of each node. The constant γ describes the rate at which the estimate of the average is updated based on information from neighboring nodes. The matrix $L := D - A$ is called the *Laplacian* of the graph.

The equilibrium points of equation (3.35) are the set of states such that $x_e[k+1] = x_e[k]$. It can be shown that if the network is connected, $x_e = (\alpha, \alpha, \ldots, \alpha)$ is an equilibrium state for the system, corresponding to each sensor having an identical estimate α for the average. Furthermore, we can show that α is indeed the average value of the initial states. Since there can be cycles in the graph, it is possible that the state of the system could enter into an infinite loop and never converge to the desired consensus state. A formal analysis requires tools that will be introduced later in the text, but it can be shown that for any connected graph we can always find a γ such that the states of the individual agents converge to the average. A simulation demonstrating this property is shown in Figure 3.22b. Although we have focused here on consensus to the average value of a set of measurements, other consensus states can be achieved through choice of appropriate feedback laws. Examples include finding the maximum or minimum value in a network, counting the number of nodes in a network or computing higher-order statistical moments of a distributed quantity [OSFM07]. ∇

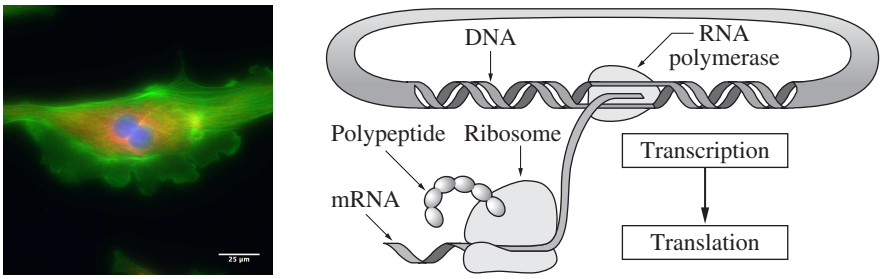


Figure 3.23: Biological circuitry. The cell on the left is a bovine pulmonary cell, stained so that the nucleus, actin and chromatin are visible. The figure on the right gives an overview of the process by which proteins in the cell are made. RNA is transcribed from DNA by an RNA polymerase enzyme. The RNA is then translated into a polypeptide chain by a molecular machine called a ribosome, and then the polypeptide chain folds into a protein molecule.

Biological Systems

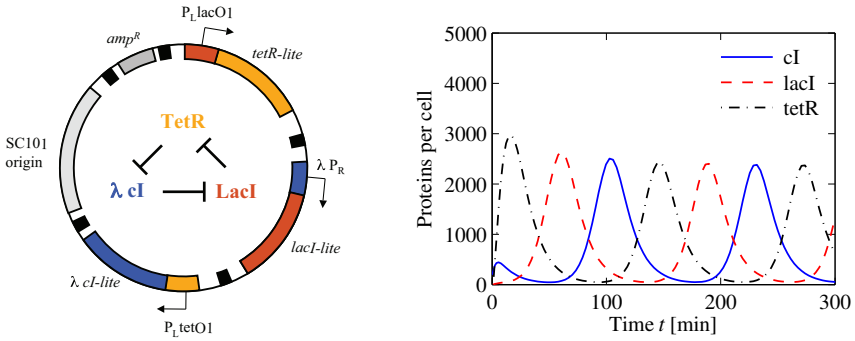
Biological systems provide perhaps the richest source of feedback and control examples. The basic problem of homeostasis, in which a quantity such as temperature or blood sugar level is regulated to a fixed value, is but one of the many types of complex feedback interactions that can occur in molecular machines, cells, organisms and ecosystems.

Example 3.15 Transcriptional regulation
Transcription is the process by which messenger RNA (mRNA) is generated from a segment of DNA. The promoter region of a gene allows transcription to be controlled by the presence of other proteins, called *transcription factors*, which bind to the promoter region and either repress or activate RNA polymerase, the enzyme that produces an mRNA transcript from DNA. The mRNA is then translated into a protein according to its nucleotide sequence. This process is illustrated in Figure 3.23.

A simple model of the transcriptional regulation process is through the use of a Hill function [dJ02, Mur04]. Consider the regulation of a protein A with a concentration given by p_a and a corresponding mRNA concentration m_a. Let B be a second protein with concentration p_b that represses the production of protein A through transcriptional regulation. The resulting dynamics of p_a and m_a can be written as

$$\frac{dm_a}{dt} = \frac{\alpha_{ab}}{1 + k_{ab} p_b^{n_{ab}}} + \alpha_{a0} - \gamma_a m_a, \qquad \frac{dp_a}{dt} = \beta_a m_a - \delta_a p_a, \tag{3.36}$$

where $\alpha_{ab} + \alpha_{a0}$ is the unregulated transcription rate, γ_a represents the rate of degradation of mRNA, α_{ab}, k_{ab} and n_{ab} are parameters that describe how B represses A, β_a represents the rate of production of the protein from its corresponding mRNA and δ_a represents the rate of degradation of the protein A. The parameter α_{a0} describes the "leakiness" of the promoter, and n_{ab} is called the Hill coefficient and relates to the cooperativity of the promoter.



(a) Repressilator plasmid (b) Repressilator simulation

Figure 3.24: The repressilator genetic regulatory network. (a) A schematic diagram of the repressilator, showing the layout of the genes in the plasmid that holds the circuit as well as the circuit diagram (center). (b) A simulation of a simple model for the repressilator, showing the oscillation of the individual protein concentrations. (Figure courtesy M. Elowitz.)

A similar model can be used when a protein activates the production of another protein rather than repressing it. In this case, the equations have the form

$$\frac{dm_a}{dt} = \frac{\alpha_{ab} k_{ab} p_b^{n_{ab}}}{1 + k_{ab} p_b^{n_{ab}}} + \alpha_{a0} - \gamma_a m_a, \qquad \frac{dp_a}{dt} = \beta_a m_a - \delta_a p_a, \tag{3.37}$$

where the variables are the same as described previously. Note that in the case of the activator, if p_b is zero, then the production rate is $\alpha_{a0} \ll \alpha_{ab}$ (versus $\alpha_{ab} + \alpha_{a0}$ for the repressor). As p_b gets large, the first term in the expression for $\dot{m}_a$ approaches 1 and the transcription rate becomes $\alpha_{ab} + \alpha_{a0}$ (versus α_{a0} for the repressor). Thus we see that the activator and repressor act in opposite fashion from each other.

As an example of how these models can be used, we consider the model of a "repressilator," originally due to Elowitz and Leibler [EL00]. The repressilator is a synthetic circuit in which three proteins each repress another in a cycle. This is shown schematically in Figure 3.24a, where the three proteins are TetR, λ cI and LacI. The basic idea of the repressilator is that if TetR is present, then it represses the production of λ cI. If λ cI is absent, then LacI is produced (at the unregulated transcription rate), which in turn represses TetR. Once TetR is repressed, then λ cI is no longer repressed, and so on. If the dynamics of the circuit are designed properly, the resulting protein concentrations will oscillate.

We can model this system using three copies of equation (3.36), with A and B replaced by the appropriate combination of TetR, cI and LacI. The state of the system is then given by $x = (m_{\text{TetR}}, p_{\text{TetR}}, m_{\text{cI}}, p_{\text{cI}}, m_{\text{LacI}}, p_{\text{LacI}})$. Figure 3.24b shows the traces of the three protein concentrations for parameters $n = 2$, $\alpha = 0.5$, $k = 6.25 \times 10^{-4}$, $\alpha_0 = 5 \times 10^{-4}$, $\gamma = 5.8 \times 10^{-3}$, $\beta = 0.12$ and $\delta = 1.2 \times 10^{-3}$ with initial conditions $x(0) = (1,200, 0, 0, 0, 0)$ (following [EL00]). ∇

Example 3.16 Wave propagation in neuronal networks

The dynamics of the membrane potential in a cell are a fundamental mechanism in understanding signaling in cells, particularly in neurons and muscle cells. The Hodgkin–Huxley equations give a simple model for studying propagation waves in networks of neurons. The model for a single neuron has the form

$$C\frac{dV}{dt} = -I_{\text{Na}} - I_{\text{K}} - I_{\text{leak}} + I_{\text{input}},$$

where V is the membrane potential, C is the capacitance, I_{Na} and I_{K} are the current caused by the transport of sodium and potassium across the cell membrane, I_{leak} is a leakage current and I_{input} is the external stimulation of the cell. Each current obeys Ohm's law, i.e.,

$$I = g(V - E),$$

where g is the conductance, which is different for different ions, and E is the equilibrium voltage. The equilibrium voltage is given by Nernst's law,

$$E = \frac{RT}{nF}\log\frac{c_e}{c_i},$$

where R is Boltzmann's constant, T is the absolute temperature, F is Faraday's constant, n is the charge (or valence) of the ion and c_i and c_e are the ion concentrations inside the cell and in the external fluid. At 20 °C we have $RT/F = 20$ mV.

The Hodgkin–Huxley model was originally developed as a means to predict the quantitative behavior of the squid giant axon [HH52]. Hodgkin and Huxley shared the 1963 Nobel Prize in Physiology (along with J. C. Eccles) for analysis of the electrical and chemical events in nerve cell discharges. The voltage clamp described in Section 1.4 was a key element in Hodgkin and Huxley's experiments. ∇

3.5 Further Reading

Modeling is ubiquitous in engineering and science and has a long history in applied mathematics. For example, the Fourier series was introduced by Fourier when he modeled heat conduction in solids [Fou07]. Models of dynamics have been developed in many different fields, including mechanics [Arn78, Gol53], heat conduction [CJ59], fluids [BRS60], vehicles [Abk69, Bla91, Ell94], robotics [MLS94, SV89], circuits [Gui63], power systems [Kun93], acoustics [Ber54] and micromechanical systems [Sen01]. Control theory requires modeling from many different domains, and most control theory texts contain several chapters on modeling using ordinary differential equations and difference equations (see, for example, [FPEN05]). A classic book on the modeling of physical systems, especially mechanical, electrical and thermofluid systems, is Cannon [Can03]. The book by Aris [Ari94] is highly original and has a detailed discussion of the use of dimension-free variables. Two of the authors' favorite books on modeling of biological systems are J. D. Murray [Mur04] and Wilson [Wil99]. A good source for system identification in Ljung [Lju99].

Exercises

3.1 (Chain of integrators form) Consider the linear ordinary differential equation (3.8). Show that by choosing a state space representation with $x_1 = y$, the dynamics can be written as

$$A = \begin{pmatrix} 0 & 1 & & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & & -a_1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & \dots & 0 & 0 \end{pmatrix}.$$

This canonical form is called the *chain of integrators* form.

3.2 (Inverted pendulum) Use the equations of motion for a balance system to derive a dynamic model for the inverted pendulum described in Example 3.2 and verify that the dynamics are given by equation (3.11).

3.3 (Discrete-time dynamics) Consider the following discrete-time system

$$x[k+1] = Ax[k] + Bu[k], \qquad y[k] = Cx[k],$$

where

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

In this problem, we will explore some of the properties of this discrete-time system as a function of the parameters, the initial conditions and the inputs.

(a) For the case when $a_{12} = 0$ and $u = 0$, give a closed form expression for the output of the system.

(b) A discrete system is in *equilibrium* when $x[k+1] = x[k]$ for all k. Let $u = r$ be a constant input and compute the resulting equilibrium point for the system. Show that if $|a_{ii}| < 1$ for all i, all initial conditions give solutions that converge to the equilibrium point.

(c) Write a computer program to plot the output of the system in response to a unit step input, $u[k] = 1$, $k \geq 0$. Plot the response of your system with $x[0] = 0$ and A given by $a_{11} = 0.5$, $a_{12} = 1$ and $a_{22} = 0.25$.

3.4 (Keynesian economics) Keynes' simple model for an economy is given by

$$Y[k] = C[k] + I[k] + G[k],$$

where Y, C, I and G are gross national product (GNP), consumption, investment and government expenditure for year k. Consumption and investment are modeled by difference equations of the form

$$C[k+1] = aY[k], \qquad I[k+1] = b(C[k+1] - C[k]),$$

where a and b are parameters. The first equation implies that consumption increases with GNP but that the effect is delayed. The second equation implies that investment is proportional to the rate of change of consumption.

Show that the equilibrium value of the GNP is given by

$$Y_e = \frac{1}{1-a} G_e,$$

where the parameter $1/(1-a)$ is the Keynes multiplier (the gain from G to Y). With $a = 0.75$ an increase of government expenditure will result in a fourfold increase of GNP. Also show that the model can be written as the following discrete-time state model:

$$\begin{pmatrix} C[k+1] \\ I[k+1] \end{pmatrix} = \begin{pmatrix} a & a \\ ab-b & ab \end{pmatrix} \begin{pmatrix} C[k] \\ I[k] \end{pmatrix} + \begin{pmatrix} a \\ ab \end{pmatrix} G[k],$$
$$Y[k] = C[k] + I[k] + G[k].$$

3.5 (Least squares system identification) Consider a nonlinear differential equation that can be written in the form

$$\frac{dx}{dt} = \sum_{i=1}^{M} \alpha_i f_i(x),$$

where $f_i(x)$ are known nonlinear functions and α_i are unknown, but constant, parameters. Suppose that we have measurements (or estimates) of the full state x at time instants $t_1, t_2, \ldots, t_N$, with $N > M$. Show that the parameters α_i can be estimated by finding the least squares solution to a linear equation of the form

$$H\alpha = b,$$

where $\alpha \in \mathbb{R}^M$ is the vector of all parameters and $H \in \mathbb{R}^{N \times M}$ and $b \in \mathbb{R}^N$ are appropriately defined.

3.6 (Normalized oscillator dynamics) Consider a damped spring–mass system with dynamics

$$m\ddot{q} + c\dot{q} + kq = F.$$

Let $\omega_0 = \sqrt{k/m}$ be the natural frequency and $\zeta = c/(2\sqrt{km})$ be the damping ratio.

(a) Show that by rescaling the equations, we can write the dynamics in the form

$$\ddot{q} + 2\zeta\omega_0\dot{q} + \omega_0^2 q = \omega_0^2 u, \tag{3.38}$$

where $u = F/k$. This form of the dynamics is that of a linear oscillator with natural frequency ω_0 and damping ratio ζ.

(b) Show that the system can be further normalized and written in the form

$$\frac{dz_1}{d\tau} = z_2, \qquad \frac{dz_2}{d\tau} = -z_1 - 2\zeta z_2 + v. \tag{3.39}$$

The essential dynamics of the system are governed by a single damping parameter ζ. The *Q-value* defined as $Q = 1/2\zeta$ is sometimes used instead of ζ.

3.7 (Electric generator) An electric generator connected to a strong power grid can be modeled by a momentum balance for the rotor of the generator:

$$J\frac{d^2\varphi}{dt^2} = P_m - P_e = P_m - \frac{EV}{X}\sin\varphi,$$

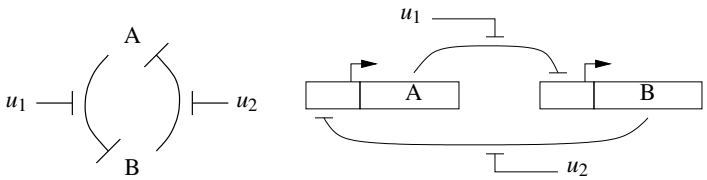
where J is the effective moment of inertia of the generator, φ the angle of rotation, P_m the mechanical power that drives the generator, P_e is the active electrical power, E the generator voltage, V the grid voltage and X the reactance of the line. Assuming that the line dynamics are much faster than the rotor dynamics, $P_e = VI = (EV/X)\sin\varphi$, where I is the current component in phase with the voltage E and φ is the phase angle between voltages E and V. Show that the dynamics of the electric generator has a normalized form that is similar to the dynamics of a pendulum with forcing at the pivot.

3.8 (Admission control for a queue) Consider the queuing system described in Example 3.12. The long delays created by temporary overloads can be reduced by rejecting requests when the queue gets large. This allows requests that are accepted to be serviced quickly and requests that cannot be accommodated to receive a rejection quickly so that they can try another server. Consider an admission control system described by

$$\frac{dx}{dt} = \lambda u - \mu_{\max}\frac{x}{x+1}, \qquad u = \mathrm{sat}_{(0,1)}(k(r-x)), \tag{3.40}$$

where the controller is a simple proportional control with saturation ($\mathrm{sat}_{(a,b)}$ is defined by equation (4.9)) and r is the desired (reference) queue length. Use a simulation to show that this controller reduces the rush-hour effect and explain how the choice of r affects the system dynamics.

3.9 (Biological switch) A genetic switch can be formed by connecting two repressors together in a cycle as shown below.

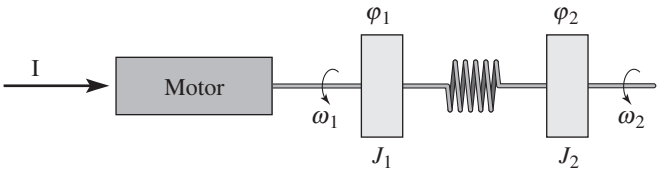


Using the models from Example 3.15—assuming that the parameters are the same for both genes and that the mRNA concentrations reach steady state quickly—show that the dynamics can be written in normalized coordinates as

$$\frac{dz_1}{d\tau} = \frac{\mu}{1+z_2^n} - z_1 - v_1, \qquad \frac{dz_2}{d\tau} = \frac{\mu}{1+z_1^n} - z_2 - v_2, \tag{3.41}$$

where z_1 and z_2 are scaled versions of the protein concentrations and the time scale has also been changed. Show that $\mu \approx 200$ using the parameters in Example 3.15, and use simulations to demonstrate the switch-like behavior of the system.

3.10 (Motor drive) Consider a system consisting of a motor driving two masses that are connected by a torsional spring, as shown in the diagram below.



This system can represent a motor with a flexible shaft that drives a load. Assuming that the motor delivers a torque that is proportional to the current I, the dynamics of the system can be described by the equations

$$\begin{aligned} J_1\frac{d^2\varphi_1}{dt^2} + c\left(\frac{d\varphi_1}{dt} - \frac{d\varphi_2}{dt}\right) + k(\varphi_1 - \varphi_2) &= k_I I, \\ J_2\frac{d^2\varphi_2}{dt^2} + c\left(\frac{d\varphi_2}{dt} - \frac{d\varphi_1}{dt}\right) + k(\varphi_2 - \varphi_1) &= T_d, \end{aligned} \tag{3.42}$$

where φ_1 and φ_2 are the angles of the two masses, $\omega_i = d\varphi_i/dt$ are their velocities, J_i represents moments of inertia, c is the damping coefficient, k represents the shaft stiffness, k_I is the torque constant for the motor, and T_d is the disturbance torque applied at the end of the shaft. Similar equations are obtained for a robot with flexible arms and for the arms of DVD and optical disk drives.

Derive a state space model for the system by introducing the (normalized) state variables $x_1 = \varphi_1, x_2 = \varphi_2, x_3 = \omega_1/\omega_0$, and $x_4 = \omega_2/\omega_0$, where $\omega_0 = \sqrt{k(J_1 + J_2)/(J_1 J_2)}$ is the undamped natural frequency of the system when the control signal is zero.