Chapter 5

Dynamical Systems

Controllers such as a thermostat for regulating the temperature in a room or a cruise controller for tracking and maintaining the speed of an automobile, are interacting with the physical world via sensors and actuators. The physical quantities involved such as temperature, pressure, and speed, evolve continuously obeying laws of physics. Design and analysis of control systems, as a result, requires construction of models of the physical system. In this chapter, we focus on continuous-time models of dynamical systems. This is a mathematically rich area that is explored in details in a course on control systems. The purpose of this chapter is to give a brief introduction to the core concepts.

5.1 Continuous-time Models

5.1.1 Continuously Evolving Inputs and Outputs

The typical architecture of a control system is depicted in Figure 5.1. The physical world that is to be controlled is modeled by a component called the *plant*. The evolution of the plant can be influenced by the controller using actuators, and the controller can make its decisions based on measurements provided by the sensors. For example, in a thermostat design, the plant is the room whose temperature is to be controlled. The sensor is a thermometer that can measure the current temperature. The task of the controller is to regulate the temperature so that it stays *close* to the temperature set by the occupant on the thermostat. The controller can influence the temperature by adjusting the heat flow from the furnace. The plant model, in this case, needs to capture how the temperature of the room changes as a function of the heat-flow from the furnace and the difference between the room temperature and the outside temperature.

Models of dynamical systems can also be conveniently described as components with inputs and outputs that are connected to one another using block diagrams. The underlying model of computation is *synchronous* as in Chapter 1 with one

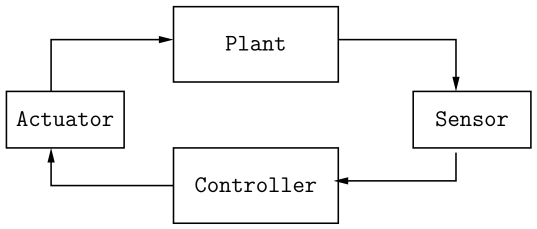


Figure 5.1: Block diagram of a control system

essential difference: while the values of the variables of a synchronous reactive component are updated in a sequence of *discrete* logical rounds, the values of the variables of a dynamical system are updated *continuously* with passage of time. We call such components *continuous-time components*. The variables of a continuous-time component typically range over a compact set such as a closed interval of the set of reals with specified units of measurement. For example, the velocity v of a car can be modeled by a variable that ranges over the real numbers in the interval from 0 to 150 miles-per-hour. In our specifications of dynamical systems, we assume that every variable has the type `real`, without explicitly mentioning the associated interval range. The values a variable takes over time can then be described as a function from the time domain to `real`. Such functions from the time domain to the set of reals are called *signals*. Throughout we will assume that the time domain consists of the set of non-negative real numbers, and denote this set by `time`. For a variable x, a signal over x is a function ρ that assigns a real value $\rho(t)$ to the variable x for every time t in `time`.

Given a set V of variables, a signal ρ over V gives values to all the variables in V as a function of time: the value of a variable x at time t is $\rho(t)(x)$. If the set V contains k variables, then a signal over V is a function from the time domain `time` to k-tuples or vectors over `real`. The number k of variables is called the *dimensionality* of the signal. A k-dimensional signal can, alternatively, be viewed as a k-tuple of single-dimensional signals, one for each of the variables in V.

Since the domain and the range of a signal consists of real numbers or vectors, we can use the standard notion of Euclidean distance over reals to measure how far apart two quantities are. For two vectors u and v, let $\|u-v\|$ denote the distance between u and v. Now, the standard mathematical notions of *continuity* and *differentiability* apply to signals. For example, a signal ρ is *continuous* if for all time values $t \in$ `time`, for all $\epsilon > 0$, there exists a $\delta > 0$ such that for all time values $t' \in$ `time`, if $\|t - t'\| < \epsilon$, then $\|f(t) - f(t')\| < \delta$ holds.

As a first example, Figure 5.2 shows a continuous-time component `NetHeat` that has two input variables h_{in} and h_{out}, and a single output variable h_{net}, which denote the heat inflow, heat outflow, and net heatflow. The component `NetHeat`

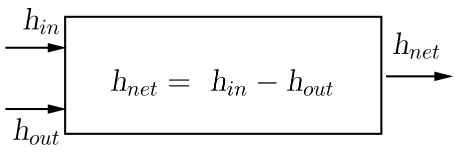


Figure 5.2: Continuous-time component `NetHeat`

is a mapping from two input signals to an output signal, and its dynamics is expressed by the equation:

$$h_{net} \;=\; h_{in} - h_{out},$$

which says that at every time t, the value of the output h_{net} equals the expression $h_{in} - h_{out}$. Given a signal ρ_{in} for the input variable h_{in}, and a signal ρ_{out} for the input variable h_{out}, the output signal ρ_{net} for the output variable h_{net} is defined by $\rho_{net}(t) = \rho_{in}(t) - \rho_{out}(t)$ for all times t in `time`. This unique output signal is called the *response* of the component to the input signals ρ_{in} and ρ_{out}. If the input signals are continuous, then so is the output signal.

Note that the computation of the output value based on the input values is expressed in a *declarative* style using *equations* instead in an *operational* style using *assignments*. Indeed such a declarative description is the norm in the modeling of control systems since models are primarily used to express relationships between various signals in a mathematically precise manner so that they can be subjected to analysis.

5.1.2 Models with State Variables

Example: Newtonian Motion

The component `NetHeat` is stateless. As an example of a stateful continuous-time component, let us build a model of how the speed of a car changes as a result of the force applied to it by the engine. For the purpose of designing a cruise controller, it typically suffices to make a number of simplifying assumptions. In particular, let us assume that the rotational inertia of the wheels is negligible and that the friction resisting the motion is proportional to the car's speed. If x denotes the position of the car (measured with respect to an inertial reference), and F denotes the force applied to the car, then using the classical Newton's laws for motion, we can capture the dynamics of the car by the differential equation:

$$F - k\,\dot{x} = m\,\ddot{x}.$$

Here k is the coefficient of the frictional force, and m denotes the mass of the car. The quantity $\dot{x}$ denotes the first-order time derivative of the signal assigning values to the variable x, and thus captures the velocity of the car. Similarly, $\ddot{x}$ denotes the second-order derivative of this signal, that is, the acceleration of the

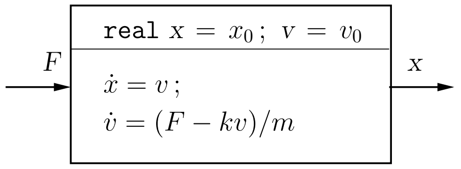


Figure 5.3: Continuous-time component modeling car motion

car. This equation of motion is modeled by the continuous-time component `Car` of Figure 5.3. It uses two state variables: the variable x modeling the position of the car and the variable v modeling the velocity of the car. For every state variable, the component needs to specify its initial value. In our example, the initial value of the position x is given by the constant x_0, and the initial value of the velocity v is given by the constant v_0. For every state variable, the dynamics is given by specifying the first-order time-derivative of the value of the state variable as a function of the input variables and state variables. The differential equation $\dot{x} = v$ says that the rate of change of the state variable x at each time t equals the value of the state variable v at time t; and the rate of change of the state variable v at each time t equals the value of the expression $(F - kv)/m$ at time t. The two equations together are equivalent to the original equation $F - k\,\dot{x} = m\,\ddot{x}$. The output of the car is its position. For every output variable, the component specifies the value of the output variable as a function of the input variables and the state variables. In this example the output simply equals the state variable x.

To illustrate the behaviors of this model, let us consider the case when the input force F equals the value kv_0 at all times. Then, the position x and the velocity v of the car at all times can be obtained by solving the system of differential equations

$$\dot{x} \;=\; v;\;\; \dot{v} \;=\; k(v_0 - v)/m$$

with the initial condition $x = x_0$ and $v = v_0$ at time 0. These equations have a unique solution: the velocity stays constant at the value v_0 at all times, and the distance x increases linearly with time t, and is given by the expression $x_0 + t \cdot v_0$. In other words, given the (constant) input signal $\rho_F(t) = kv_0$, the corresponding signal describing the dynamics of the state variable v is given by $\rho_v(t) = v_0$, and the signal describing the state/output variable x is given by $\rho_x(t) = x_0 + t \cdot v_0$.

Differential Equations for Specifying Dynamics

In general, the dynamics of the component is specified by (1) a real-valued expression h_y for every output variable y, and (2) a real-valued expression f_x for every state variable x. Each of these expressions is an expression over the input variables and the state variables. The value of the output variable y at time t is obtained by evaluating the expression h_y using the values of the state and input variables at time t, and the signal for a state variable x should be such that its rate of change at each time t equals the value of the expression f_x evaluated using the values of the state and input variables at time t. Thus, the execution of a continuous-time component is similar to the execution of a deterministic synchronous reactive component, except the notion of a round is now infinitesimal: at every time t, the outputs at time t are determined as a function of the inputs at time t and the state of the component at time t, and then the state is updated using the rate of change specified by the derivative evaluated using the inputs and state at time t.

Determining the signals for the state variables and the output variables corresponding to a given input signal using mathematical analysis amounts to solving an *initial value problem* for a system of *ordinary differential equations.* We need to impose restrictions on the expressions used to define the state and output responses to ensure uniqueness of such solutions since not all differential equations are well-behaved. For example, for the differential equation

$$\dot{x} \;=\; \texttt{if}\ x = 0\ \texttt{then}\ 1\ \texttt{else}\ 0,$$

there is no differentiable function ρ that can satisfy the given equation. If the right-hand-side of a differential equation is a *continuous* function, then a solution is guaranteed to exist. As another example, for the initial value $x_0 = 0$, the differential equation $\dot{x} = x^{1/3}$ has two solutions: the constant signal $\rho_1(t) = 0$ and the signal $\rho_2(t) = (2/3) \cdot t^{3/2}$. A classical way to avoid this problem and ensure *uniqueness* of the solution to a differential equation is to require the right-hand-side to be *Lipschitz continuous.* Recall that a function $f : \texttt{real}^n \mapsto \texttt{real}^n$ is said to be Lipschitz continuous if for every compact set S of $\texttt{real}^n$, there exists a constant K such that for all vectors u and v in $\texttt{real}^n$, $\|f(u) - f(v)\| \leq K \cdot \|u - v\|$.

We then require that the expression f_x defining the rate of change of the state variable x is a Lipschitz continuous function over the state and input variables. Now, assuming that the input signal is continuous, the state signal is uniquely defined. We would like the output signal to be continuous as outputs of one component can be connected to other components as inputs in a block diagram. For this purpose, we demand that each expression h_y defining the output variable y is a continuous function.

The definition of a continuous-time component is now summarized below:

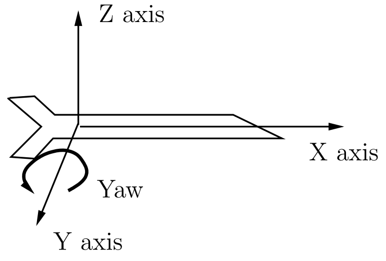


Figure 5.4: Simplified modeling of helicopter motion

CONTINUOUS-TIME COMPONENT

A continuous-time component C has a finite set I of real-valued input variables, a finite set O of real-valued output variables, a finite set S of real-valued state variables, an initial value $x_0 \in \texttt{real}$ for every state variable $x \in S$, a continuous real-valued expression h_y over $I \cup S$ for every output variable $y \in O$, and a Lipschitz-continuous real-valued expression f_x over $I \cup S$ for every state variable $x \in S$. Given a continuous signal ρ over the input variables I, the corresponding execution of the component C is the unique continuous signal ρ' over the state variables S and the output variables O such that (1) for every state variable x, $\rho'(0)(x) = x_0$; (2) for every output variable y and time t, $\rho'(t)(y)$ equals the value of h_y evaluated using the values $\rho(t)(I)$ for the input variables and $\rho'(t)(S)$ for the state variables; and (3) for every state variable x and time t, the time derivative $d/dt\, \rho'(t)$ equals the value of f_x evaluated using the values $\rho(t)(I)$ for the input variables and $\rho'(t)(S)$ for the state variables.

Example: Helicopter Spin

As another example, let us consider the classical problem of controlling a helicopter so as to keep it from spinning. A helicopter has 6 degrees of motion, 3 for position and 3 for rotation. In our simplified version, let us assume that the helicopter position is fixed and the helicopter remains vertical. Then the only freedom of motion is the angular rotation around the Y-axis, which is in the horizontal plane perpendicular to the body axis. This rotation is called the *yaw* (see Figure 5.4). The friction of the main rotor at the top of the helicopter causes the yaw to change. The tail rotor then needs to apply a torque to counteract this rotational force to keep the helicopter from spinning. In this setting, the helicopter model has a continuous-time input signal T, denoting the torque around the Y-axis. The moment of inertia of the helicopter in this simplified setting can be modeled by a single scalar I. The output signal of the model is the angular velocity around the vertical axis, and is modeled by the spin $s = \dot{\theta}$,

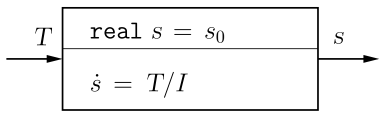


Figure 5.5: Continuous-time component modeling helicopter spin

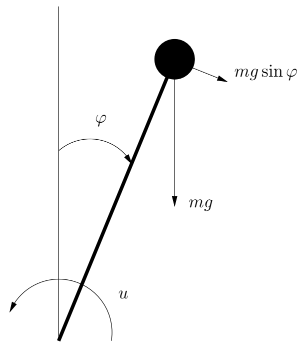


Figure 5.6: Inverted pendulum

where θ gives the yaw. The equation of motion is then given by

$$\ddot{\theta} \ = \ T/I$$

The corresponding continuous-time component is shown in Figure 5.5. There is a single state variable s modeling the spin, a single input variable T corresponding to the torque, and a single output variable, which equals the state s. The initial value of the state is s_0 and its rate of change is given by the expression T/I.

The models expressed as continuous-time components with state variables and differential equations are sometimes called the *state-space representation* of dynamical systems. The dynamics can alternatively be expressed using equations that specify the output signals by integrating over input signals. For instance, for our helicopter model, the value of the output spin at time t equals the sum of its initial value and the integral of the input torque upto time t. This is captured by the *integral equation*

$$s(t) \ = \ s_0 \ + \ (1/I) \int_0^t T(\tau) d\tau.$$

Note that the internal state is implicit in the integral model. Modeling tools such as Simulink allow modeling by both state-space representation and by integral equations.

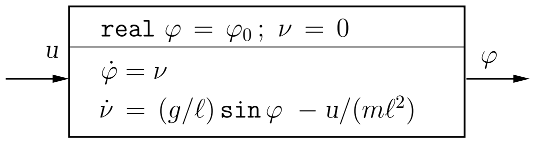

u
real $\varphi = \varphi_0$; $\nu = 0$
$\dot{\varphi} = \nu$
$\dot{\nu} = (g/\ell)\,\mathtt{sin}\,\varphi - u/(m\ell^2)$
φ

Figure 5.7: Continuous-time component modeling the inverted pendulum

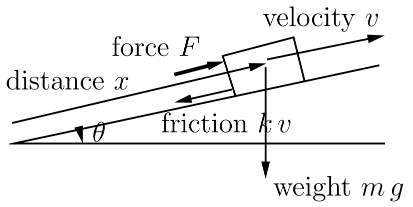


Figure 5.8: Car motion on a graded road

Example: Simple Pendulum

Consider a simple pendulum shown in Fig. 5.6. The pendulum is a rod with a rotational joint at one end and a mass at the other end. For simplicity, we assume that the friction and the weight of the rod are negligible. A motor is placed at the pivot to provide an external torque to control the pendulum. From Newton's law for rotating objects, the dynamics of the pendulum system is described by the following (nonlinear) differential equation:

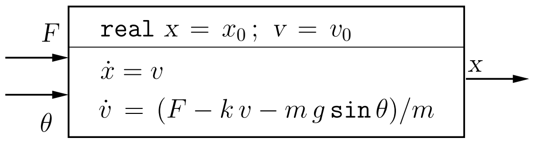
$$-\,m\,\ell^2\,\ddot{\varphi} + m\,g\,\ell\,\sin\varphi = u$$

where m is the weight of the mass, $g = 9.8\,m/s^2$ is the gravitational acceleration, ℓ is the length of the rod, φ is the angle of the pendulum measured clockwise from the upward vertical, and u is the external torque applied in the counterclockwise direction. Note that the range of values for the angle φ is the interval $[0, 2\pi)$.

The continuous component modeling the pendulum is shown in Figure 5.7. The second-order differential equation of motion is replaced by a pair of (first-order) differential equations by introducing the state variable ν that captures the angular velocity of the pendulum. The constant φ_0 gives the initial angular position of the pendulum.

5.1.3 Models with Disturbance

Let us revisit the model of the motion of a car. The model described in Figure 5.3 assumes that the car is moving in a single dimension on a flat road. Suppose we now want to account for the *grade* of the road: on an up-hill, the weight of the car works against the force applied by the engine, and on a down-hill,



F
θ
real x = x_0 ; v = v_0
$\dot{x} = v$
$\dot{v} = (F - k\,v - m\,g\,\mathtt{sin}\,\theta)/m$
x

Figure 5.9: Continuous-time component modeling car motion on a graded road

the weight of the car adds to this force (see Figure 5.8). The cruise controller needs to adjust the force F to keep the net velocity v in the direction along the road constant. We can model the grade of the road by an additional input, denoted θ, that captures the angle of the road with the horizontal: a positive angle indicates an up-hill slope and a negative angle indicates a down-hill slope. The weight of the car equals $m\,g$ in the vertically downwards direction, where $g = 9.8\,m/s^2$ is the gravitational acceleration. The modified dynamical system model is shown in Figure 5.9. The forces acting on the car in the direction along the road are: F in the forward direction controlled by the engine, $k\,v$ in the backward direction modeling the friction, and $m\,g\,\mathtt{sin}\,\theta$ capturing the gravitational force along the road in the backward direction.

The control design problem for the model of Figure 5.9 differs from the corresponding problem for the model of Figure 5.3 in a crucial way: the input signal θ modeling the grade of the road is not controlled by the controller and is not known in advance. Such an uncontrolled input is sometimes called a *disturbance*. The controller should be designed to produce the controlled input signal F so that it works no matter how the input θ varies within a reasonable range of values (for instance, all values in the range [-30,+30]).

5.1.4 Composing Components

Continuous-time components can be composed using block diagrams in a way similar to synchronous components. We can define the operations of variable renaming, output hiding, and parallel composition in the same way as in Chapter 1. To ensure determinism and well-formed composition, while composing components, we would like to establish absence of cyclic awaits dependencies. The awaits-dependency of an output variable on input variables is defined the same way as in synchronous components: an output variable y of a continuous-time component C awaits an input variable x if the value of the output y at time t depends on the value of the input x at time t.

For the component `NetHeat` of Figure 5.2, the output h_{net} awaits both the input variables h_{in} and h_{out}. For the components of Figure 5.3 and Figure 5.9 modeling the motion of a car, and the component of Figure 5.5 modeling the motion of a helicopter, the output variable at time t equals the value of one of the state variables at time t, and hence, does not await any of the corresponding input variables.

Since the evolution of each output variable y of a continuous-time component is described by an expression h_y over the state and input variables, the await-dependencies can be determined by a simple syntactic check: if the set of input variables that occur in the expression h_y is J, then the output y awaits the input variables in J, but does not await the remaining input variables.

5.1.5 Stability

In earlier chapters, we have explored safety and liveness requirements for synchronous as well as asynchronous systems. Control systems also have similar requirements. For example, a safety requirement for a cruise controller can demand that the speed of the car should always be below some maximum speed, and a liveness requirement can demand that the difference between the actual speed and the desired speed of the car should eventually be close to zero. A cruise controller, however, has a new kind of requirement, namely, that small perturbations in input values, such as the grade of the road, should not cause disproportionately large changes in the speed of the car. This requirement, which is relevant only for continuous-time systems, is called *stability*. We will first define the notion of Lyapunov stability for dynamical systems, and then consider the notion of bounded-input-bounded-output stability state for continuous-time components.

Equilibria

To define stability, we first need to understand the notion of an equilibrium state of operation of a dynamical system. For this purpose, we assume that there are no inputs: if the original system has inputs, then the stability is analyzed by setting the input signal to a fixed value, say 0, at all times.

Consider a closed continuous-time component C whose state x is an n-dimensional vector, with the dynamics given by the Lifschitz-continuous differential equation $\dot{x} = f(x)$. A state x_e of the system is said to be an *equilibrium* state if $f(x_e) = 0$. Clearly, if the initial state of the system is an equilibrium state x_e, then the state of the system stays at this equilibrium state at all times.

As an example, let us revisit the continuous-time component `Car` of Figure 5.3. First, let us set the input force F to be the constant signal with value 0. Then, the dynamics of the component is described by the equations

$$\dot{x} \;=\; v;\; \dot{v} \;=\; -k\,v/m.$$

A state (x_e, v_e) is an equilibrium state of this system exactly when $v_e = 0$. When the velocity v_e equals 0, the signal that sets the position x always equal to x_e, and sets the velocity v always equal to 0, satisfies the above differential equations. On the other hand, if the initial velocity v_e is non-zero, the velocity will keep changing at an exponential rate. Thus, for the input signal $F = 0$, the system has infinitely many equilibria of the form $(x_e, 0)$.

Now, suppose we set the input force F to be the constant signal with value F_0, for $F_0 \neq 0$. Then, the dynamics of the component is described by the equations

$$\dot{x} \;=\; v;\; \dot{v} \;=\; (F_0 - kv)/m.$$

Now the acceleration $\dot{v}$ is 0 exactly when the velocity v equal F_0/k, but since this quantity is non-zero, it is not possible to satisfy the equations $\dot{x} = 0$ and $\dot{v} = 0$ simultaneously. This implies that, when the car is applied a constant non-zero force, the resulting dynamical system has no equilibrium state.

As another example, consider the pendulum model of Figure 5.7, and suppose the input torque u is set to the constant value 0. Then, the dynamics of the component is described by the equations

$$\dot{\varphi} \;=\; \nu;\; \dot{\nu} \;=\; g\,\mathtt{sin}\,\varphi/\ell.$$

Recall that the angular displacement φ ranges over the interval $[0, 2\pi)$, and in this range, $\mathtt{sin}\,\varphi$ equals 0 for two values of φ, namely, $\varphi = 0$ and $\varphi = \pi$. Thus, the system has two equilibria: one where $\varphi = 0$ and $\nu = 0$ (and this corresponds to the pendulum in the vertically inverted position), and one where $\varphi = \pi$ and $\nu = 0$ (and this corresponds to the pendulum in the vertically downwards position).

Lyapunov Stability

Let us consider a continuous-time component whose state x is an n-dimensional vector, with the dynamics given by the Lifschitz-continuous differential equation $\dot{x} = f(x)$. Consider an equilibrium state x_e of the system. We know that if the initial state of the system is x_e, then the corresponding system evolution is described by the constant signal $\rho_e(t) = x_e$. Now suppose the initial state is perturbed slightly, that is, the initial state is chosen to be x_0 such that the distance $\|x_e - x_0\|$ is small. Consider the signal ρ_0 that is the unique response of the system starting from the initial state x_0. If this signal stays "close" to the constant signal ρ_e at all times, then we can conclude that a small perturbation from the equilibrium state causes the system state to stay close to the equilibrium. In such a case, the equilibrium x_e is said to be stable. If in addition, the state $\rho_0(t)$ *converges* to the equilibrium state x_e as time t advances, then we are guaranteed that after a small perturbation from the equilibrium state, the system state stays close to the equilibrium eventually returning to the equilibrium. When this additional convergence requirement holds, the equilibrium x_e is said to be *asymptotically* stable.

These notions of stability, usually referred to as Lyapunov stability in the literature on dynamical systems, are summarized below:

Lyapunov Stability

Consider a continuous-time component C with state $x \in \texttt{real}^n$ and dynamics given by the equation $\dot{x} = f$, where $f : \texttt{real}^n \mapsto \texttt{real}^n$ is Lipschitz continuous. A state x_e is said to be an equilibrium of the component C if $f(x_e) = 0$. Given an initial state $x_0 \in \texttt{real}^n$, let $\rho_0 : \texttt{time} \mapsto \texttt{real}^n$ be the unique response of the component C from the initial state x_0. An equilibrium x_e of C is said to be stable if for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all states x_0 with $\|x_e - x_0\| < \delta$, $\|\rho_o(t) - x_e\| \leq \epsilon$ holds at all times $t \geq 0$. An equilibrium x_e of C is said to be asymptotically stable if it is stable, and there exists a $\delta > 0$ such that for all states x_0 with $\|x_e - x_0\| < \delta$, the limit $\texttt{lim}_{t \to \infty} \rho_0(t)$ exists and equals x_e.

Let us revisit the continuous-time component `Car` of Figure 5.3 again, with the input force F set to the constant value 0. We have already noted that the state with the position $x = x_e$ and the velocity $v = 0$ is an equilibrium, for every choice of x_e. Suppose we perturb this equilibrium; that is, consider the behavior of the component starting at the initial state (x_0, v_0) such that the distance $\|(x_e, 0), (x_0, v_0)\|$ is small. The car will slow down according to the differential equation $\dot{v} = -k\,v/m$ starting from the initial velocity v_0, with the velocity converging to 0. The position of the car will converge to some value x_f that is a function of the initial position x_0 and the initial velocity v_0. Thus, for the resulting signal ρ_0, the value $\|\rho_o(t), (x_e, 0)\|$ is bounded (the exact bound depends on the values of v_0, x_e, x_0, and x_f). Thus, the equilibrium $(x_e, 0)$ is stable. However, it is not asymptotically stable, since along the signal ρ_0, as time advances, the position does not converge to x_e, but converges to a different value x_f.

Now, let us consider the pendulum of Figure 5.7 with the input signal u set to the constant value 0. We know that the model $\dot{\varphi} = \nu$; $\dot{\nu} = g\,\texttt{sin}\,\varphi/\ell$ has two equilibria, namely, $(\varphi = 0,\ \nu = 0)$ and $(\varphi = \pi,\ \nu = 0)$. The former corresponds to the vertically upwards position, and is not stable: if we displace the pendulum from this vertically upwards position setting the angular displacement φ to a small positive value, the angular velocity ν will keep increasing positively, thereby increasing the displacement φ, pushing the pendulum away from the vertical. On the other hand, the equilibrium $(\varphi = \pi, \nu = 0)$ corresponding to the vertically downwards position is stable. For example, if we set the initial angle to be slightly more than π, the angular velocity will be negative, and will cause the angle to decrease back towards the equilibrium value π. The pendulum will oscillate around the equilibrium position, and if the initial perturbation is (φ_0, ν_0), the value of $\|\rho_0(t), (\pi, 0)\|$ stays bounded, with the bound dependent on the values of φ_0 and ν_0. In this model, this equilibrium is not asymptotically stable: for example, if the initial starting position is $\pi + \epsilon$ with initial angular velocity 0, the pendulum will keep swinging forever in the arc from $\pi + \epsilon$ to $\pi - \epsilon$ around the vertical ($\varphi = \pi$). In reality, of course, such a pendulum asymptotically converges to the vertical coming to a halt, due to the damping effects not captured in our model.

Input-Output Stability

The notion of Lyuponov stability is based on the state-space representation of the dynamical system, and concerns the behavior of the system when its state is perturbed from the equilibrium state, with the input signal set to 0. An alternative notion of stability views the dynamical system as a tranformed mapping input signals to output signals, and demands that a small change to the input signal should cause only a small change to the output signal. This notion of *input-output stability* is formalized next.

A signal ρ assigning values to a real-valued variable as a function of time, is said to be *bounded* if there exists a constant Δ such that $\|\rho(t)\| \leq \Delta$ for all times t. Here are some typical signals analyzed for their boundedness:

- The constant signal defined by $\rho(t) = a$, for constant a, is bounded.
- The linearly increasing signal defined by $\rho(t) = a + bt$, for constants a and $b > 0$, is not bounded.
- The exponentially increasing signal defined by $\rho(t) = a + e^{bt}$, for constants a and $b > 0$, is not bounded.
- The exponentially decaying signal defined by $\rho(t) = a + e^{-bt}$, for constants a and $b > 0$, is bounded.
- The step-signal defined by $\rho(t) = a$ for $t < t_0$ and $\rho(t) = b$ for $t \geq t_0$, for constants t_0, a, b, is bounded.
- The sinusoidal signal defined by $\rho(t) = a \cos bt$, for constants a, b, is bounded.

A signal ρ over a set V of variables is bounded if the component of ρ corresponding to each variable x in V is bounded.

In a stable system, when started in the initial state $x_0 = 0$, whenever the input signal is bounded then so is the output signal produced by the component in response. The bound on the output signal can be different from the bound on the input signal. This particular formalization of stability is known as *Bounded Input Bounded Output* (BIBO) stability:

BIBO Stability

A continuous-time component C with input variables I and output variables O is *Bounded-Input-Bounded-Output stable* if for every bounded input signal ρ, the output signal ρ' produced by C starting in the initial state $x_0 = 0$, in response to the input signal ρ, is also bounded.

Let us consider the model of the helicopter from Figure 5.5:

$$\dot{s} = T/I$$

and assume that the initial spin is 0. This system is unstable. Suppose we apply a constant torque T_0 to the system, then the rate of change of the spin s is constant. The spin will keep increasing linearly: the input signal is the bounded constant function $\rho(t) = T_0$, and the corresponding output signal is described by the unbounded function $\rho'(t) = (T_0/I)\,t$.

5.2 Linear Systems

When the dynamics of a continuous-time component is specified using linear expressions, a number of questions regarding the behavior of the system can be answered using mathematical analysis.

5.2.1 Linearity

A *linear* expression over a set of variables is formed using the operations of addition and multiplication by a numerical constant. If the variables are $x_1, x_2, \ldots x_n$, then a linear expression has the form $a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n$, where $a_0, a_1, \ldots a_n$ are real-valued coefficients. A linear system is a continuous-time component where the expressions used to specify the dynamics of the state variables and the output variables are such linear expressions.

When the coefficient a_0 equals 0, the linear expression is said to be *homogeneous*. A homogeneous linear expression evaluates to 0 when all the variables are assigned the value 0. In models of dynamical systems, expressions used to specify the dynamics are typically homogeneous, and if not, the model can be tranformed by introducing additional state variables so that the dynamics uses only homogeneous expressions, while preserving linear dependence. Henceforth, we will use "linear expression" to mean "homogeneous linear expression."

Linear Component

A continuous-time component C with input variables I, output variables O, and state variables S, is said to be a *linear* component if (1) for every output variable y, the expression h_y is a (homogeneous) linear expression over $I \cup S$, and (2) for every state variable x, the expression f_x is a (homogeneous) linear expression over $I \cup S$.

In our examples, the components of Figure 5.2, Figure 5.3, and Figure 5.5 are linear, but the component of Figure 5.9 with the non-linear term $m\,g\,\texttt{sin}\,\theta$ in the dynamics is not linear. However, notice that the input θ is not controlled, and simply represents disturbance or noise that the controller must handle. As a result we can replace the input θ by another variable d with the meaning that d captures the value of $m\,g\,\texttt{sin}\,\theta$. Now the dynamics becomes $\dot{v} = (F - k\,v - d)/m$, and is linear. The range of values for input disturbance $\theta \in [-30, +30]$ must be replaced by the range $[m\,g\,\texttt{sin}\,(-30), m\,g\,\texttt{sin}\,(+30)]$ for the new variable d.

Matrix-based Representation

The conventional form for expressing the dynamics for linear systems uses matrices. Consider a linear component with m input variables $I = \{u_1, \dots u_m\}$, n state variables $S = \{x_1, \dots x_n\}$, and k output variables $O = \{y_1, \dots y_k\}$. In this case, we can view an input as a vector of dimension $m \times 1$, a state as a vector of dimension $n \times 1$, and an output as a vector of dimension $k \times 1$. The dynamics is expressed by four matrices each with real-valued coefficients: a matrix A of dimension $n \times n$, a matrix B of dimension $n \times m$, z matrix C of dimension $k \times n$, and a matrix D of dimension $k \times m$. The dynamics is given by

$$\dot{S} = AS + BI,\ O = CS + DI.$$

That is, for each state variable x_i, the differential equation modeling its rate of change as a function of the state variables and the input variables is given by the linear differential equation:

$$\dot{x_i} \ = \ A_{i1}\, x_1 + \ \cdots \ + A_{in}\, x_n \ + \ B_{i1}\, u_1 + \ \cdots \ + B_{im}\, u_m$$

For each output variable y_j, its value is defined in terms of the state variables and the input variables by the linear expression:

$$y_j \ = \ C_{j1}\, x_1 + \ \cdots \ + C_{jn}\, x_n \ + \ D_{j1}\, u_1 + \ \cdots \ + D_{jm}\, u_m.$$

In our example of the model of the car of Figure 5.3, $m = 2$ and $n = k = 1$. The matrices are given by:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -k/m \end{bmatrix};\quad B = [0 \ \ 1/m];\quad C = [1 \ \ 0];\quad D = [0]$$

Linear Response

We have defined linearity based on the state-space representation of the component. Linearity can, alternatively, be studied based on the properties of the transformation from the space of input signals to the space of output signals induced by the component.

Consider a continuous-time component C with an input variable x and an output variable y. Given a input signal ρ : `time` $\mapsto$ `real`, there is a unique output signal ρ' : `time` $\mapsto$ `real` corresponding to the execution of the component C corresponding to the input signal ρ. Thus, a continuous-time component is a function from the set of input signals to the set of output signals. This transformation is guaranteed to be *linear* for a linear component. Linearity of transformations means the following two properties:

- **Scaling:** If the input signal is scaled by a constant factor, then the output signal also gets scaled by the same factor. Given an input signal ρ and a constant α, let $\alpha\,\rho$ be the input signal whose value at each time t is $\alpha\,\rho(t)$. Then, for a linear component C, for all input signals ρ and all scaling factors α, if ρ' is the output signal corresponding to the input signal ρ, then $\alpha\,\rho'$ is the output signal corresponding to the input signal $\alpha\,\rho$.

- **Additivity:** If an input signal can be expressed as a sum of two input signals, then the corresponding output signal is also the sum of the output signals corresponding to the component input signals. That is, if the input signals ρ, ρ_1 and ρ_2 are such that $\rho(t) = \rho_1(t) + \rho_2(t)$ for all times t, and if ρ', ρ'_1, ρ'_2 are the output signals produced by the component corresponding to the input signals ρ, ρ_1, and ρ_2, respectively, then it must be the case that $\rho'(t) = \rho'_1(t) + \rho'_2(t)$ for all times t.

In general, the component has multiple input and multiple output variables, and we need to consider signals that are mappings from the time domain to the set of real-valued vectors. In this case, linearity is defined by considering sum of vectors and scaling of vectors. For two signals ρ and ρ' over a set V of variables, and constants $\alpha, \beta \in$ `real`, the signal $\alpha\,\rho + \beta\,\rho'$ is defined to be the signal that assigns, for every time t, the value $\alpha\,\rho(t)(x) + \beta\,\rho'(t)(x)$ to each variable $x \in V$. Linearity of transformations is now captured by the following theorem:

Theorem 5.1 [Linearity of Input-Output Transformation] *Let C be a linear component with input variables I and output variables O. For all input signals ρ_1 and ρ_2, and constants $\alpha, \beta \in$ `real`, if the output signals generated by the component C in response to the input signals ρ_1 and ρ_2 are ρ'_1 and ρ'_2, respectively, then the output signal generated by the component C in response to the input signal $\alpha\,\rho_1 + \beta\,\rho_2$ is $\alpha\,\rho'_1 + \beta\,\rho'_2$.*

Exercise 5.1 *:* Let us revisit the non-linear model of the pendulum of Figure 5.7. A classical approach to designing controllers for non-linear systems is to linearize the model about an operating point then design a controller for that linearized model and use it for the original system. For the pendulum example, the operating point of interest is $\varphi = 0$ corresponding to the vertical position of the pendulum. Using the fact that $\sin\varphi \approx \varphi$ for small values of φ, build a corresponding linear component model of the pendulum. ■

5.2.2 Solutions of Linear Differential Equations

For linear systems, a number of analysis techniques are available to understand how the output signal is related to the input signal. Let us first consider the linear differential equation $\dot{S} = AS$, and suppose the initial state is given by the vector x_0. To solve this equation, we can construct a sequence of signals $\rho^0, \rho^1, \ldots$ that approximate the desired solution in the following manner. Let ρ^0 be the constant signal defined by $\rho^0(t) \; = \; x_0$, and for each $m > 0$, define:

$$\rho^m(t) \;=\; x_0 \;+\; \int_0^t \; A \cdot \rho^{m-1}(\tau)\, d\tau.$$

After substitution, and simplification, we obtain

$$\rho^m(t) \;=\; [\mathbf{I} \;+\; \sum_{j=1}^{m} A^j \cdot t^j / j\,!] \cdot x_0,$$

where $\mathbf{I}$ is the identity matrix. The sequence of functions $\rho^0, \rho^1, \rho^2, \ldots$ converges to the unique solution of the differential equation, given by

$$\rho(t) \;=\; [\mathbf{I} \;+\; \sum_{j=1}^{\infty} A^j \cdot t^j / j\,!] \cdot x_0.$$

Recall that, for a real number a, the quantity e^a is defined by:

$$e^a \;=\; 1 + \sum_{j=1}^{\infty} a^j / j! \;.$$

Similarly, the *matrix exponential* e^A, for a matrix A, is defined by the equation:

$$e^A \;=\; \mathbf{I} \;+\; \sum_{j=1}^{\infty} A^j / j\,!$$

With this notation, the solution of the differential equation $\dot{S} = AS$, with the initial state x_0, is given by

$$\rho(t) \;=\; e^{At} \cdot x_0.$$

A similar analysis can be performed for the model of a linear component with inputs. Consider the dynamics $\dot{S} = AS + BI$. Suppose the initial state is given by the vector x_0. Given an input signal ρ, the resulting state signal ρ' is given by the equation:

$$\rho'(t) \;=\; e^{At} x_0 \;+\; \int_0^t e^{A(t-\tau)}\, B\, \rho(\tau)\, d\tau.$$

The response of the system to a given input signal can be computed using this equation: the output value at time t equals $C\rho'(t) + D\rho(t)$.

A number of mathematical tools exist to compute the quantity e^{At}, depending on the structural properties of the matrix A. We consider only a simple, but common, case: the matrix A is a diagonal matrix (that is, each entry A_{ij}, for $i \neq j$, equals 0). Let us denote the i-th diagonal entry of the diagonal matrix A by a_i. In this case, observe that, for every j, the matrix A^j is also a diagonal matrix, whose i-th diagonal entry is given by a_i^j. It is easy to verify that the desired quantity e^{At} is also a diagonal matrix, and its i-th diagonal entry is the scalar $e^{a_i t}$.

5.2.3 Stability

For linear systems, stability can be analyzed using standard mathematical tools. Consider the system given by

$$\dot{S} \;=\; AS \;+\; BI; \; O \;=\; CS \;+\; DI.$$

To check whether the system is BIBO-stable, we set the initial state $x_0 = 0$, and consider the behavior of the system for a bounded input signal $\rho(t)$. We note a sufficient condition for checking BIBO-stability in terms of Lyapunov-stability criterion:

Theorem 5.2 [Reducing BIBO-Stability to Lyapunov Stability] *If the equilibrium $x_0 = 0$ is asymptotically stable for the dynamical system $\dot{S} = AS$, then the continuous-time component with the dynamics $\dot{S} = AS + BI$ and $O = CS + DI$, is BIBO-stable.*

This suggests that stability can be analyzed by setting the input signal to the constant value 0, and the stability depends only on the properties of the matrix A. Asymptotic stability of the equilibrium 0 means that, irrespective of the initial state x_0, the state of the system goes to 0 as time advances.

Note that for the helicopter model, if we set the input torque to 0, dynamics is given by $\dot{s} = 0$. In this dynamics, if the initial spin is s_0, it will remain constant at the value s_0, and thus the system is unstable.

Consider a system with one state variable x and one input variable u, and suppose the dynamics is given by the linear equation

$$\dot{x} \; = \; ax + bu.$$

When the input signal for u is always 0, the dynamics becomes $\dot{x} = ax$. If x_0 denotes the initial state, then the state at time t equals $e^{at}x_0$. If the coefficient a is negative, then no matter what the initial value of x is, the value of x decays exponentially, and will become 0 in the limit. In this case, the equilibrium $x_0 = 0$ is asymptotically, and the system is BIBO stable. If the coefficient a is 0, then the value of x stays equal to its initial value x_0, and the system is unstable. If the coefficient a is positive, then the value of x increases exponentially and grows in an unbounded manner, and the system is unstable. Thus, the stability of one-dimensional linear system depends on the sign of the coefficient of the term capturing dependence of the rate of change on the state.

In the general case of linear systems, the eigenvalues of the matrix A in the dynamics can be examined to determine the stability. For a $(n \times n)$-matrix A, if the equation $A\,x \; = \; \lambda\,x$, for a constant $\lambda \in \texttt{real}$ and a non-zero vector $x \in \texttt{real}^n$, then the value λ is called an *eigenvalue* of the matrix A, and the vector x is called an eigenvector of A corresponding to the eigenvalue λ. A $(n \times n)$-matrix A has at most n distinct eigenvalues, and these correspond to the *characteristic equation* of A, given by

$$\texttt{det}(\,A \; - \; \lambda\,\mathbf{I}) \; = \; 0,$$

where $\texttt{det}$ represents the *determinant* of a matrix. Note that if λ_1 is an eigenvalue, then the term $(\lambda - \lambda_1)$ is a factor of the characteristic polynomial $\texttt{det}(\,A \; - \; \lambda\,\mathbf{I})$. If $\lambda_1, \ldots \lambda_p$ are all the eigenvalues of the matrix A then

$$\texttt{det}(\,A \; - \; \lambda\,\mathbf{I}) \; = \; (\lambda - \lambda_1)^{n_1} \cdot \cdots \cdot (\lambda - \lambda_p)^{n_p}$$

where n_j is the multiplicity of the eigenvalue λ_j, and $n_1 + \cdots + n_p$ equals n.

An eigenvalue of a matrix is, in general, a complex number. Stability is guaranteed if the real part of such an eigenvalue is either negative, or is 0 with multiplicity 1. If an eigenvalue has a positive real part, or is 0 with multiplicity 2, then the system is unstable. If the matrix A is a diagonal matrix (all entries A_{ij} with $i \neq j$, are zeros) then the eigenvalues are simply the coefficients A_{ii} along the diagonal, and if they are all non-positive, with at most one zero, stability is assured.

Theorem 5.3 [Stability Test for Linear Dynamics] *For the dynamical system given by $\dot{S} = AS$, the equilibrium $x_0 = 0$ is asymptotically stable if and only if for every eigenvalue λ of the matrix A, either the real part of λ is negative, or the real part of λ is 0 and the multiplicity of λ is 1.*

5.3 Designing Controllers

Given a dynamical system model of the plant, the controller is designed to provide the controlled input signals to maintain the output of the system close to the desired output irrespective of changes in uncontrolled input signals corresponding to disturbances. Designing controllers in a principled manner is a well-developed discipline. We will review some basic terminology in control design, and get familiar with the most commonly used class of controllers in industrial practice.

5.3.1 Open-loop vs. Feedback Controller

Given a dynamical system model of the plant, the controller is designed to provide the controlled input signals to maintain the output of the system close to the desired output irrespective of changes in uncontrolled input signals corresponding to disturbances. Let us review a basic classification for control design.

An *open-loop* controller does not use measurements of the state or outputs of the plant to make its decisions. Such a controller relies on the model of the plant to decide on the controlled input for the plant, and its implementation does not require sensors. For example, consider the first model of the car from Figure 5.3. Suppose the controller's objective is to maintain a constant velocity (that is, we want $\dot{v} = 0$ at all times). Then, if the initial velocity is v_0, the desired value of the input force F equals $k\,v_0$: the controller can simply apply this constant force to the car to maintain the velocity constant at the value v_0. Such a controller is called *open-loop*: when compared to the architecture shown in Figure 5.1, the block for sensors and the flow of information from the plant to the controller is missing. Obviously, such a controller would be less expensive to implement than one that requires sensors to estimate the state of the plant. However, its design heavily relies on the assumption that the behavior of the plant is entirely predictable and accurately captured by the idealized

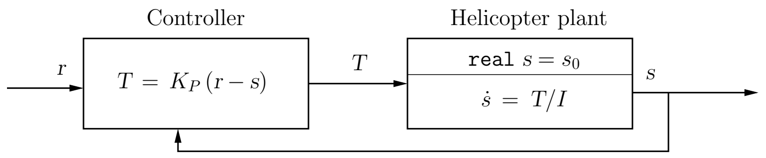


Figure 5.10: Stabilizing controller for the helicopter model

mathematical model. In practice, operation of such a controller is acceptable, provided there is a possibility of manual intervention. If the driver finds the speed of the car unacceptable, she would simply increment or decrement the desired speed triggering a recalculation of the force applied by the open-loop controller.

A *feedback* controller uses sensors to measure the output, and thus, indirectly the current state of the plant, to update the values of the controlled input variables. For example, in the revised model of the car in Figure 5.9, the model accounts for the change in the grade of the road. Suppose that the controller is applying the correct amount of force to maintain the velocity of the car at the desired cruising speed. A positive change in the grade θ causes the car to slow down, while a negative change in θ causes the car to speed up. The speed of the car, as measured by the sensors, is an input to the controller. It notices the change in speed and adjusts the force to make the velocity again equal to the desired cruising speed. A feedback controller not only can cope with disturbances (such as the grade) whose variation with time is not predictable in advance, but can work well even when the mathematical model of the plant is only a rough approximation of the real-world dynamics. Implementation of a feedback controller requires sensors, and its performance is related to the accuracy of measurements by these sensors.

5.3.2 Stabilizing Controller

We now describe a simple and typical pattern for designing a controller using the helicopter example. The controller is shown in Figure 5.10. It takes two input signals: the input variable r represents the *reference signal* that captures the desired spin, and the signal s is the plant output, namely, the (measured) spin of the helicopter. Given two such input variables corresponding to the desired and actual values, we can define the *error signal* e by the equation $e = r - s$. The goal of the controller is to keep the magnitude of the error as small as possible, and also ensure that the closed-loop system obtained by composing the helicopter model and the controller is stable. Note that a positive value of e means that the controller should try to increase the actual spin s by applying a positive torque, and a negative value of e means that the controller should try to decrease the actual spin s by applying a negative torque. The controller of Figure 5.10 computes the torque by simply scaling the error signal by a positive

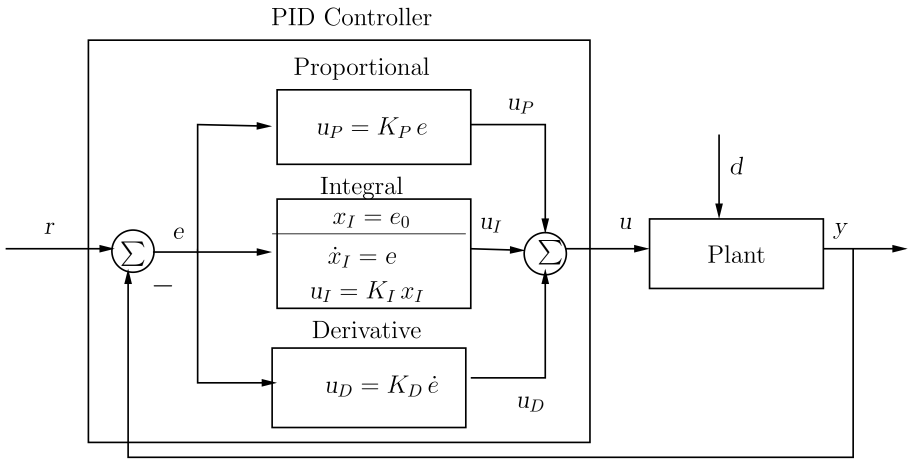


Figure 5.11: A generic PID controller

constant factor K_P. Such a controller is called a *proportional controller* and the constant K_P is called the *gain* of the controller.

For the closed-loop system consisting of the composition of the controller and the helicopter, the input signal is the reference value r and the output is the spin s. The dynamics of the composite system is given by the equation

$$\dot{s} \;=\; K_P\,(r - s)/I.$$

This is a one-dimensional linear system, and the coefficient capturing the dependence of the rate of change of the state variable s on itself is $-K_P/I$. We know that such a system is stable exactly when this coefficient is negative. Thus the composite system is stable as long as the gain K_P is positive.

When the reference input is 0, that is, when the objective of the controller is to keep the helicopter from spinning, the controller applies the torque equal to $-K_P\, s/I$. No matter what the initial spin s_0 is, this causes the spin to decay to 0 exponentially. Higher the value of K_P, faster is the rate of convergence.

5.3.3 PID Controllers

In industrial control systems, the most commonly used design of a controller to correct the discrepancy between the desired reference signal and the measured output signal uses a combination of three terms: a *proportional* term capturing the reaction to the current error, an *integral* term capturing the reaction to the cumulative error, and a *derivative* term capturing the response to the rate of change of error. Such a controller is called a PID controller.

The controller is shown in Figure 5.11. The controller takes two signals as inputs: the reference signal r and the measured output y of the plant. Note that both

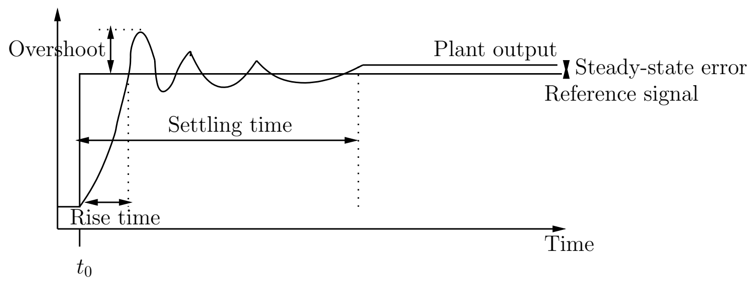


Figure 5.12: Typical output response to a step reference signal

r and y can be vectors. For example, in the general setting for controlling the motion of a helicopter, the reference signal includes a three-dimensional motion plan from the source to the destination, along with constraints on the angular rotations. Let the variable e denote the *error signal* capturing the difference between the reference signal r and the measured plant output y. Then, the controller's output u, which is fed to the plant, is the sum of three terms.

- **Proportional term** $K_P\, e$: The contribution of this term is directly proportional to the current error. The constant K_P is called the *proportional gain*, and the controller scales the error by this factor.

- **Integral term** $K_I \int_0^t e(\tau)\, d\tau$: Note that the integral of the error signal upto time t gives the cumulative error upto time t, and thus, the contribution of this term accounts for the cumulative error so far. The constant K_I is called the *integral gain*, and the controller scales the accumulated error by this factor.

- **Derivative term** $K_D\, \dot{e}$: The contribution of this term is corelated to the rate at which the error changes. The constant K_D is called the *derivative gain*, and the controller scales the rate of change of the error by this factor.

Note that the proportional component of the PID controller is stateless. The integral component maintains a state variable x_I that captures the accumulated error, and the rate of change of this state variable equals the error e. The derivative component has a state variable x_D that stores the error e, and the output of the derivative component is the first-order derivative of this state variable. In Figure 5.11, using the standard convention, the Sum block is illustrated as a circle labeled with the symbol Σ. Such a component simply outputs the sum of its input signals, where the negative sign on an input signal indicates that the corresponding input should be subtracted. Some of the components may be missing in a specific control design. For instance, a P controller has only the proportional block, and a PI controller has only the proportional and the integral blocks. Both of these are also common in practice.

To understand how the control performance changes with the contributions of various terms, let us consider how the output of a linear plant typically changes in response to a step signal given as a reference signal. Figure 5.12 shows a typical change in the output signal y. At the time instance t_0 when the reference signal jumps from the initial value to the set-point, the magnitude of the error variable e is highest. As e changes, so does the output u of the controller, which impacts the state and the output of the plant. The output approaches 1, but overshoots the desired level. The smoothness of the dynamics of the physical world makes such an overshoot unavoidable. The overshoot makes the error negative causing the controller to change the direction of the derivative of its output. The same phenomenon repeats causing the output to oscillate for a while before it settles into a steady-state value (with no change in absence of further external disturbances). For such a response in the plant output, the following metrics capture the performance of the controller:

1. *Overshoot:* The difference between the maximum value of the plant output and the desired reference value. Ideally, the overshoot should be as small as possible. In particular, a safety requirement can assert that the overshoot should be below some threshold value.

2. *Rise time:* The time difference between the instance t_0 when the reference signal changes and the time at which the output signal crosses the desired reference value. Ideally, smaller rise time means better responsiveness of the system, and typically, an attempt to reduce the rise time will increase the overshoot.

3. *Steady state error:* The difference between the steady-state value of the output signal and the set-point value of the reference signal. Ideally, steady-state error should be 0, but a small error may be acceptable.

4. *Settling time:* The time difference between the instance t_0 when the reference signal changes and the time at which the output signal reaches its steady-state value. Ideally settling time should also be small, and the system should reach the desired output value with few oscillations.

The values of the three parameters K_P, K_I, and K_D of the PID controller are adjusted so that the controller's performance is acceptable on all the above metrics. Higher values of the proportional gain K_P mean that the rise time will be smaller, but with higher overshoot. Too high a value of K_P can cause large oscillations delaying settling time. A purely proportional controller also has a steady-state error. The effect of the integral term gets rid of the steady-state error. Higher values of the integral gain K_I lead to better responsiveness, but also contribute to overshoot. The overshoot is reduced by using the derivative component. Higher values of the derivative gain K_D contribute to reducing the overshoot and the settling time. A number of tools are available for automatic tuning of these three parameters for desired performance.

Exercise 5.2 *:* Recall the pendulum from Figure 5.7, and the corresponding linear model from Exercise 5.1. We will use a *Proportional-Derivative* (PD) controller for the linearized model. The feedback control is given as $u = K_P\,\varphi + K_D\,\dot{\varphi}$, where we need to suitably choose the gain parameters K_P and K_D. Write the equations of the closed-loop system which consists of the composition of the linearized model of the pendulum and the PD controller. For what values of the parameters K_P and K_D is the closed-loop system stable? (Hint: all eigenvalues of the system matrix should have negative real parts.)

Suppose the mass m and the length ℓ are such that $m\ell^2 = 1\,(kg.m^2)$ and $mg\ell = 1\,(N.m)$. Simulate the closed-loop system using MATLAB with different values of the gain parameters K_P and K_D. You can choose any initial position $\varphi(0)$ and initial angular velocity $\dot{\varphi}(0)$. Experiment with different parameter values. Plot and discuss your results. ■

5.4 Analysis Techniques

Given a model of a continuous-time component, which may include the plant model, the feedback controller, and constraints on the initial values and disturbances, we want to analyze the behavior of the system. We first discuss the traditional approach based on *numerical simulation*, and then some emerging *symbolic* techniques for verifying safety requirements.

5.4.1 Numerical Simulation

Consider a continuous-time component whose state is captured by the variable x which may be a vector of variables, and whose input is described by a variable u which also can be a vector of variables. The function f gives the rate of change of the value of x as a function of state x and input u. Given an input signal that assigns values to u as a function of time and an initial state x_0, the evolution of the state of the system, then, can be computed by solving the differential equation $\dot{x} = f(x, u)$ with $x(0) = x_0$. While for some specific forms of the function f, a closed-form solution for the state response $x(t)$ at time t can be computed, a general method for computing this signal is to employ numerical simulation. For numerical simulation, the user provides a discretization step parameter Δ, and the simulator attempts to compute the values of the state at times $\Delta, 2\Delta, 3\Delta, \cdots$ that approximate the values of the desired response signal $x(t)$ as closely as possible. The simulation algorithm samples the input signal $u(t)$ only at times $0, \Delta, 2\Delta, 3\Delta, \cdots$ and the result of the simulation thus depends on the discrete sequence $u_0, u_1, u_2, \ldots$ of input values, where u_i is the value of the input signal $u(t)$ at time $t = i\,\Delta$ for each i.

Euler's Method

Euler's method relies on the observation that the rate of change of x, that is, dx/dt, at time t, is simply the limit of the quantity $(x(t + \Delta) - x(t))/\Delta$ as Δ

goes to 0. As a result, for small values of Δ, it is natural to approximate the value of $x(t+\Delta)$ by the following equation

$$x(t+\Delta) \;=\; x(t) + \Delta\, f(x(t), u(t)).$$

This approximation assumes that the rate of change of x is constant during the interval $[t, t+\Delta)$ and equals the rate of change at the beginning of the interval. The rate of change at the beginning of the interval is obtained by evaluating the function f using the values of x and u at time t. The change in the value of x is obtained by multiplying this rate by the size Δ of the interval. Thus, given an initial value x_0 for the state, and a sequence of values $u_0, u_1, \ldots$ for the input, the Euler's method for simulation of differential equation $\dot{x} = f(x, u)$ computes the values, for every $i \geq 0$,

$$x_{i+1} \;=\; x_i + \Delta\, f(x_i, u_i).$$

This sequence of values is linearly-extrapolated to give the response signal that defines the state $x(t)$ at every time $t \in$ `time`: for every $i \geq 0$, and time value $t \in [i\,\Delta, i\,\Delta + \Delta)$, $x(t) = x_i + (t - i\,\Delta)\, f(x_i, u_i)$.

Runge-Kutta Methods

Euler's method estimates the state at the end of interval by assuming that the rate of change of state stays constant during the interval, and this rate is based only on the state at the beginning of the interval. A better approximation can be obtained if estimated change in the state at the end of the interval is used to estimate a change in the derivative, and use this to readjust the state estimate. Runge-Kutta methods comprise a popular class of numerical integration methods based on this idea. In particular, the *second-order Runge-Kutta method* computes the state x_{i+1} by the following calculation:

$$\begin{aligned} k_1 &= f(x_i, u_i) \\ k_2 &= f(x_i + \Delta\, k_1, u_{i+1}) \\ x_{i+1} &= x_i + \Delta\,(k_1 + k_2)/2 \end{aligned}$$

Given the current state x_i and input u_i, the first step computes k_1 to be the current rate of change. However, instead of setting the state x_{i+1} at the end of the interval to be $x_i + \Delta\, k_1$ as in Euler's method, it uses this estimated state to calculate the estimated rate of change k_2 at the end of the interval (the input value u_{i+1} at the end of the interval is used for this estimate). The third step calculates x_{i+1} assuming that the rate of change is constant during the interval, but equals the average of the two values k_1 and k_2.

The higher-order Runge-Kutta methods use the same basic idea, but use estimates of derivatives at the midpoint as well as at the end-points to compute a weighted average. The most commonly used method in practice is the Fourth-order Runge-Kutta method.

The quality of approximation afforded by numerical simulation depends on the step size Δ. Most simulation tools automatically tune this parameter to keep the error small. In particular, adaptive techniques are used to change the value of Δ dynamically, possibly at every step of simulation, to adjust to the current rate of change.

5.4.2 Computing Reachable States

Numerical simulation is an effective technique to understand the behavior of a dynamical system starting from a specific initial state. When we know that the initial state belongs to a set, which can be described by constraints on initial values, the simulation tool needs to choose different values for the initial state from the given set, and run multiple simulations. Such an approach cannot be exhaustive, and thus, we need some alternative techniques.

In this section, let us focus on the following safety verification problem. We are given a dynamical system C with state variable x and dynamics described by $\dot{x} = f(x)$. Given a set *Init* of initial states and a state property φ, we want to know if for every state signal $x(t)$ with $x(0) \in \mathit{Init}$, is it the case that the state $x(t)$ satisfies φ for every t. That is, if the system is initialized to start in a state satisfying the constraint described by *Init*, we want to check if the state always stays within the region described by φ. The set *Init* describes the set of possible initial states, and a violation of φ denotes an error. For example, the property φ can assert that the magnitude of the error e between the reference signal and the plant output is less than some constant δ; a violation of this property would indicate an unacceptable overshoot.

A natural strategy is to develop a *symbolic simulation* algorithm in the style of the symbolic reachability algorithm of Section 2.4. To implement such a symbolic reachability analysis, we need a way to represent regions (that is, sets of states) that supports operations such as intersection, emptiness test, and subset test. For continuous-time systems, a natural candidate for such a representation is *polyhedra*. An n-dimensional polyhedron is specified by the inequality $R\,x \leq b$, where R is $(r \times n)$-matrix and b is $(n \times 1)$-vector. Each row R_j of the matrix corresponds to a linear constraint over the state variables: $R_j\,x \leq b_j$, and the number r of rows corresponds to the number of constraints. In one dimension, a polyhedron is an interval, and in two dimensions, a polyhedron is a polygon where the number of constraints needed to represent the polygon corresponds to the number of sides. An alternative to such a constraint-based representation uses the extremal vertices of the polyhedron: each such vertex corresponds to an n-dimensional vector, and the polyhedron is simply the convex-hull of all the vertices. Algorithms are available to transform one representation to another.

Let us review some operations on polyhedra:

- *Emptiness test:* given a polyhedron represented by $R\,x \leq b$, is it empty? This amounts to finding a vector x that satisfies the set of linear con-

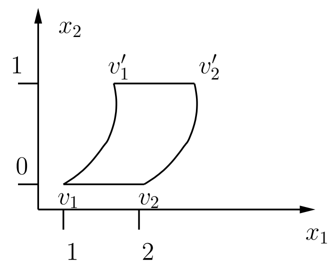


Figure 5.13: Post-image of a line segment under linear dynamics

straints given by R, and can be solved using standard algorithms for linear programming.

- *Intersection:* given two polyhedra $R_1\,x \leq b_1$ and $R_2\,x \leq b_2$, compute their intersection. First note that intersection of two polyhedra is indeed a polyhedron. Algorithms are available for computing intersection of polyhedra. However, intersection adds to the complexity of representation. For instance, intersection of two triangles can be a hexagon.

- *Union:* Union of two polyhedra need not be a polyhedron. As a result, we can either switch to a representation in which a set of states is described by a list of polyhedra, or approximate the union operation by *convex-hull* of polyhedra. Convex hull of two polyhedra is the smallest polyhedron that contains the union of the two polyhedra, and can be computed effectively from the representation of the input polyhedra.

- *Subset test:* given two polyhedra $R_1\,x \leq b_1$ and $R_2\,x \leq b_2$, to check whether the first polyhedron is entirely contained within the second, we can compute the vertices of the first, and check that each such vertex v lies within the second (that is, $R_2\,v \leq b_2$ holds).

For our symbolic analysis, we can assume that the initial region *Init*, and the desired invariant property φ are represented using polyhedra. We are ready to use the symbolic reachability algorithm, provided we know how to implement the image computation step `Post`. More specifically, given a time step Δ, a polyhedron P, and dynamics $\dot{x} = f(x)$, we want to compute the set $\texttt{Post}_\Delta(P, f)$ of states that can appear on state signals of the system before time Δ when started in a state belonging to the set P. This is thus the set of reachable states from the set P upto the time horizon Δ. Formally, the set $\texttt{Post}_\Delta(P, f)$ is

$$\{x \mid \exists t \leq \Delta\,.\,\exists \rho : [0,t] \mapsto \texttt{real}^n\,.\,\rho(0) \in P \wedge \rho(t) = x \wedge \forall\,\tau \in [0,t)\,.\,\dot{\rho}(\tau) = f(\rho(\tau))\}$$

Unfortunately, even when the system dynamics is linear, the post-image of a polyhedron need not be a polyhedron. For example, consider the 2-dimensional

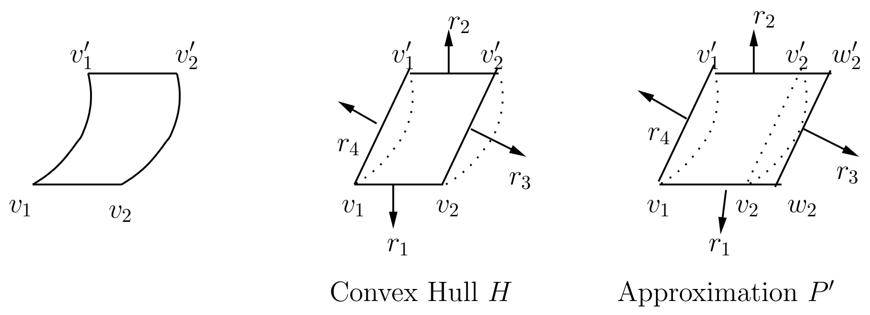


Figure 5.14: Post-image computation by flow-pipe approximation

system with the dynamics $\dot{x}_1 = x_1$ and $\dot{x}_2 = 1$. For the initial set given by the polyhedron $1 \leq x_1 \leq 2 \wedge x_2 = 0$, Figure 5.13 shows the set of reachable states upto time 1. The initial set is the line segment joining the vertices $v_1 = (1, 0)$ and $v_2 = (2, 0)$. For the given dynamics and the initial region, note that the value of x_2 at time t is t, and the value of x_1 at time t is $e^t\, x_{10}$, where $1 \leq x_{10} \leq 2$ is the starting value. Thus, the set of reachable states upto time 1 is the region with vertices v_1, v_2, $v_1' = (e, 1)$, and $v_2' = (2e, 1)$ shown in Figure 5.13. This is not a polyhedral set, and is not even convex.

The best we can hope for is to *over-approximate* the post-image: we compute a polyhedron P' that is a superset of $\texttt{Post}_\Delta(P, f)$. If we can establish that the over-approximation of the reachable set is contained in the desired invariant φ, we have established that the system satisfies the desired safety requirement. If states violating φ are found reachable, this may not be a real bug: bugs introduced due to the over-approximation of the set of reachable states by the analysis tool are called *false negatives*.

We describe a procedure, called *flowpipe computation*, for over-approximating the set $\texttt{Post}_\Delta(P, f)$ with a polyhedron. This computation proceeds according to the following steps:

1. For every vertex v of the initial polyhedron P, compute the state $v' = \texttt{Post}_\Delta(v, f)$ of the system at time Δ assuming the state is v at time 0. This can be done by numerical simulation for a non-linear system, or from the closed form solution for a linear system. Ixf A is the state matrix for a linear system, then $v' = e^{A\Delta}\, v$.

2. Compute the convex-hull H of all the vertices v of P and the vertices $v' = \texttt{Post}_\Delta(v, f)$ obtained in step 1. The region H is a polyhedron, but may not contain all of $\texttt{Post}_\Delta(P, f)$.

3. Let the representation of the convex-hull H be $R\,x \leq d$, where R is $r \times n$ matrix. For each $1 \leq j \leq r$, we compute the smallest d_j' such that the

constraint $R_j\, x \leq d'_j$ holds at all times upto time Δ for all choices of initial states $x_0 \in P$. The values d'_j can be obtained by integrating optimization in numerical simulation, or for linear systems, by a direct calculation. The desired polyhedral over-approximation is $Rx \leq d'$.

Intuitively, in step 3, we keep the number and normal vectors for facets of the desired polyhedron same as those for the convex-hull H. The facets are moved outwards so as to include all executions starting in P by adjusting the coefficients d_j.

The computation of a flowpipe approximation is illustrated in Figure 5.14. The vertices of the initial polyhedron P are v_1 and v_2. Their post-images at time $\Delta = 1$ are vertices v'_1 and v'_2 as computed in the step 1. The convex hull H is the parallelogram connecting these four vertices, and the vectors r_1, r_2, r_3, and r_4 are the normal vectors to the sides of this polyhedron H. In step 3, the sides are pushed outside to include all reachable states. The resulting over-approximation P' is the parallelogram with vertices v_1, w_2, v'_1, and w'_2, and has the same normal vectors as the convex hull H.

For the symbolic simulation method, we assumed that the dynamical system has no inputs. If the system has only controlled inputs, and the controller has already been designed to supply these controlled inputs based on the plant outputs, then the closed-loop system consisting of the parallel composition of the plant and the controller has no external inputs, and we can apply the analysis method. If the system has uncontrolled inputs, and we have a deterministic model for evolution of such inputs, then also our analysis technique can be applied to the composition of the environment model supplying the inputs and the plant, together with the controller. If only constraints on the uncontrolled inputs are known, then we would like to verify the system for all input signals that satisfy these constraints. Such models have nondeterminism initially as well as at every simulation step, and the flowpipe computation step needs to be adapted to account for the resulting nondeterminism.

5.4.3 Barrier Certificates

In Chapter 2, we studied the principle of *inductive invariants* to prove safety requirements of (discrete) transition systems. To show that a property φ is an invariant of a transition system T, we find a property ψ such that (1) every state satisfying ψ satisfies φ, (2) the initial states of T satisfy ψ, and (3) if a state s satisfies ψ and (s,t) is a transition of T, then the state t is guaranteed to satisfy ψ. We describe an analogous method for establishing that a given state property is an invariant of a continuous-time system.

Let us reconsider the safety verification problem discussed in Section 5.4.2: given a dynamical system C with state variable x, dynamics described by $\dot{x} = f(x)$, a set *Init* of initial states, and a state property φ, we want to know if for every state signal $x(t)$ with $x(0) \in \mathit{Init}$, is it the case that the state $x(t)$ satisfies φ for every t. As usual, the state x can be an n-dimensional vector.

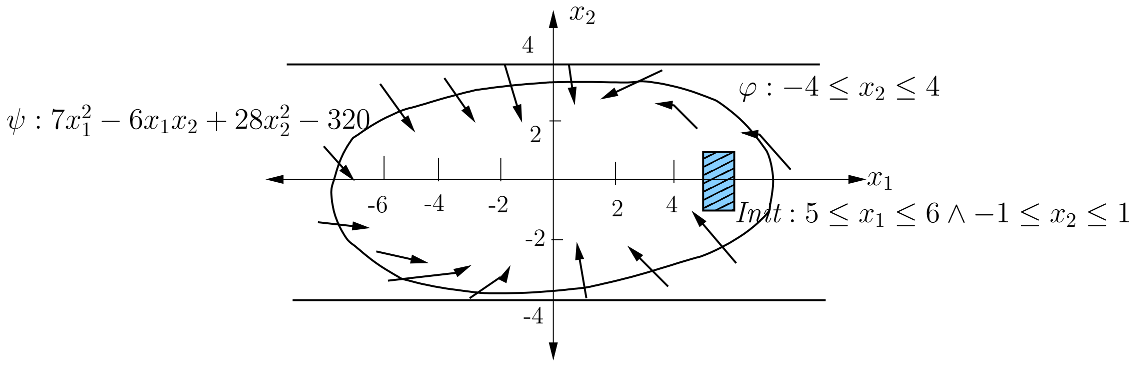

x_2
4
2
-2
-4
x_1
-6
-4
-2
2
4
$\psi : 7x_1^2 - 6x_1x_2 + 28x_2^2 - 320$
$\varphi : -4 \leq x_2 \leq 4$
$Init : 5 \leq x_1 \leq 6 \wedge -1 \leq x_2 \leq 1$

Figure 5.15: Illustrating invariant verification using barrier certificate

We will explain the method using an example. Consider a two-dimensional linear system whose dynamics is given by

$$\dot{x_1} \;=\; -7\,x_1 \,+\, 3\,x_2\,; \quad \dot{x_2} \;=\; -3\,x_1.$$

Suppose the initial set is described by the formula

$$Init \;=\; 5 \,\leq\, x_1 \,\leq\, 6 \;\wedge\; -1 \,\leq\, x_2 \,\leq\, 1.$$

In Figure 5.15, the initial set corresponds to the shaded rectangle. We want to establish the property φ described by $-4 \leq x_2 \leq 4$ is an invariant for the system executions starting in the initial set. In Figure 5.15, we want to show that if the initial state belongs to the rectangle, the execution stays between the two horizontal lines $x_2 = 4$ and $x_2 = -4$.

As in case of proofs using inductive invariants, the first step is to identify the "strengthening" ψ. In this case, the desired strengthening is a function ψ : $\texttt{real}^n \mapsto \texttt{real}$ mapping states to real values, that is, a real-valued expression over the state vector x. The set of states satisfying the equation $\psi(x) = 0$ is called the *barrier*. The set of states satisfying the formula $\psi(x) \leq 0$, that is, the set of states inside the barrier, is the analog of the inductive invariant. The barrier is chosen to satisfy three obligations. First, we need to establish that every state satisfying $\psi(x) \leq 0$ also satisfies φ. Equivalently, if a state x violates φ, then $\psi(x)$ must be positive. The second obligation is to show that every initial state x satisfies $\psi(x) \leq 0$. These two obligations imply that the barrier should be chosen so that it separates initial states from states violating the desired invariant property: all "bad" states are outside the barrier, and all initial states are inside the barrier.

For the example of Figure 5.15, the barrier is described by the function

$$\psi(x) \;=\; 7\,x_1^2 \,-\, 6\,x_1\,x_2 \,+\, 28\,x_2^2 \,-\, 320.$$

The equation $\psi(x) = 0$ describes the ellipse shown the figure. Initial states lie inside the ellipse, while "bad" states (that is, states above the line $x_2 = 4$ and below the line $x_2 = -4$) are outside the ellipse.

The third obligation in the discrete case corresponds to showing that the inductive invariant is preserved by system transitions. In case of continuous-time components, discrete transitions are replaced by evolution of state over time as described by the differential equation $\dot{x} = f(x)$. Now we can exploit properties of continuous and differentiable functions. A system execution $x(t)$ demonstrating a violation of the invariant must start in an initial state, which lies inside the barrier, and end in a bad state, which lies outside the barrier. The system execution is a continuous and differentiable function, and hence, it cannot "jump" across the barrier, and must cross the barrier, that is, there must be a time τ such that the state $x(\tau)$ lies on the barrier, the state $x(\tau - dt)$ just before time τ is inside the barrier, and the state $x(\tau + dt)$ just after time τ is outside the barrier. It suffices to establish that the choice of the barrier with respect to the system dynamics f is such that crossing the barrier is impossible.

To understand the required condition, consider the example of Figure 5.15. At a given state (x_1, x_2), the value of x_1 changes at the rate $-7\,x_1 + 3\,x_2$, and the value of x_2 changes at the rate $-3\,x_1$. Thus, the state (x_1, x_2) flows, or evolves, along the vector $(-7\,x_1 + 3\,x_2, -3\,x_1)$ given by the dynamics f. These flow directions at some selected points are depicted by little arrows in Figure 5.15. A key observation is that the value of ψ always decreases along these flow directions. In particular, if the state lies on the barrier (that is, the boundary of the ellipse), this vector field points inwards. This implies that if the state of the system at time t lies on the barrier, then the state of the system at time $t + dt$ must lie in the interior. It is clear that system executions cannot cross the barrier from inside to outside.

The appropriate mathematical concept to formalize the requirement described above concerns *Lie derivatives*. The Lie derivative of a function described by the expression ψ with respect to the vector field described by the dynamics f is denoted $L_f\,\psi$, and gives the rate of change of the value of ψ as a function of the state x as the state evolves along the direction $f(x)$. In other words, the Lie derivative $L_f\,\psi$ is the *directional derivative* of the function ψ along the vector field f. The third obligation for ψ to be a barrier certificate is that the value of the function $(L_f\,\psi)(x)$ should always be non-positive. This ensures that for every value of x, $\psi(x)$ decreases as x evolves according to f: for sufficiently small values of time increments $dt > 0$, $\psi(x + f(x)\,dt)$ does not exceed $\psi(x)$.

A technical requirement for this argument is that the function described by the expression ψ should be a *smooth* function. A smooth function is a function for which all derivatives, higher order derivatives as well as partial derivatives, exist. This ensures that the Lie derivative is well defined. In the example of Figure 5.15, the function ψ is a quadratic function and is smooth. Every polynomial function is a smooth function. A discontinuous function is not smooth, and this means that the barrier separating the initial states and the bad states cannot be a rectangle (or a polyhedron).

The proof technique for establishing invariants using barrier certificates is summarized below.

Barrier Certificate

Let C be a continuous-time component with state x and dynamics given by $\dot{x} = f(x)$. A state property φ is invariant along all executions of C starting in initial states given by another property *Init*, if there exists a smooth real-valued expression ψ over state variables, called a *barrier certificate*, such that (1) if *Init*(x) holds then $\psi(x) \leq 0$; (2) if $\varphi(x)$ does not hold then $\psi(x) > 0$; and (3) the Lie derivative of ψ with respect to the vector field f is never positive: $(L_f\,\psi)(x) \leq 0$.

Exercise 5.3 *:* This project concerns implementing a tool for symbolic reachability analysis of one-dimensional dynamical systems in Matlab.

A closed interval of the set of real numbers is denoted $[a, b]$ where $a, b \in$ `real` and $a \leq b$, and corresponds the set of all real numbers between a and b. State of a one-dimensional dynamical system is a real number. A set of such states can be naturally represented as a *union* of closed intervals. For example, $[0, 1] \cup [4, 5]$ is the set $\{x \in \texttt{real} \mid 0 \leq x \leq 1 \text{ or } 4 \leq x \leq 5\}$. For the purpose of this project, let us fix the data type `reg` for regions to be such unions of closed intervals, each such region is of the form $A = \bigcup_i [a_i, b_i]$.

1. Implement a programming library for computing with regions. The first step is to choose a data structure to represent regions. Make sure the empty set is also represented correctly. Explain succinctly and rigorously your implementation of the following operations.
   - Union (disjunction) of regions: $A \cup B$.
   - Difference of regions: $A \setminus B$.
   - Check if region A is a subset of region B: $A \subseteq B$.
   - Check for emptiness: is the region A empty.
   - Sum of a region and a scalar or another region: $A + B = \{x + y \mid x \in A, y \in B\}$.
   - Product of a region and a scalar: $\alpha * A = \{\alpha * x \mid x \in A\}$.
   - Square of a set: $A^2 = \{x^2 \mid x \in A\}$.
2. Consider a discretized representation of the dynamical system of the form $x_{k+1} = f(x_k, u_k)$, $k \geq 0$, where $x_k \in$ `real` is the state, u_k is the control and is constrained in a set $U \subseteq$ `real`, f expresses the dynamics of the system using only the operations in part (a). The set of initial states is *Init*. Let *Reach*$_k$ be the set of reachable states at step k, that is, *Reach*$_k$ is the set

of all states x such that there exists an execution of the system starting from some $x_0 \in \mathit{Init}$ under some control inputs $u_0, u_1, ..., u_{k-1}$ in U and ending at states $x_k = x$. The set *Reach* of all reachable states is $\mathit{Reach} = \bigcup_{k=0,1,...} \mathit{Reach}_k$. Considera breadth-first search algorithm to compute successive values of Reach_k upto a maximum number of iterations N. The algorithm should terminate as soon as there is no new state that can be added to *Reach*, or when it iterates for N steps. In the latter case, *Reach* will be the set of reachable states for the first N iterations. Implement the algorithm using the library you developed in part (a), provided that *Init* and U are also regions, that is, unions of closed intervals. Experiment your implementation using the following examples. In each case, report how the algorithm terminates and at which step, Output *Reach* (in its minimal form as a union of disjoint closed intervals).

- $x_{k+1} = -0.95x_k + u_k$, $\mathit{Init} = [1, 2]$, $U = [-0.1, 0.1]$, $N = 100$;
- $x_{k+1} = -0.96x_k + u_k$, $\mathit{Init} = [1, 2]$, $U = [-0.1, 0.1]$, $N = 100$;
- $x_{k+1} = -0.95x_k + u_k$, $\mathit{Init} = [1, 2]$, $U = [-0.2, 0.2]$, $N = 100$;
- $x_{k+1} = 0.5x_k^2 + u_k$, $\mathit{Init} = [1.8, 1.89]$, $U = [0, 0.1]$, $N = 40$;
- $x_{k+1} = 0.5x_k^2 + u_k$, $\mathit{Init} = [1.8, 1.9]$, $U = [0, 0.1]$, $N = 40$ (do not plot Reach_k).

■