Chapter 11

Robot Control

A robot arm can exhibit a number of different behaviors, depending on the task and its environment. It can act as a source of programmed motions for tasks such as moving an object from one place to another or tracing a trajectory for a spray paint gun. It can act as a source of forces, as when applying a polishing wheel to a workpiece. In tasks such as writing on a chalkboard, it must control forces in some directions (the force must press the chalk against the board) and motions in others (the motion must be in the plane of the board). When the purpose of the robot is to act as a haptic display, rendering a virtual environment, we may want it to act like a spring, damper, or mass, yielding in response to forces applied to it.

In each of these cases, it is the job of the robot controller to convert the task specification to forces and torques at the actuators. Control strategies that achieve the behaviors described above are known as **motion control**, **force control**, **hybrid motion–force control**, or **impedance control**. Which of these behaviors is appropriate depends on both the task and the environment. For example, a force-control goal makes sense when the end-effector is in contact with something but not when it is moving in free space. We also have a fundamental constraint imposed by the mechanics, irrespective of the environment: the robot cannot independently control the motion and force in the same direction. If the robot imposes a motion then the environment will determine the force, and if the robot imposes a force then the environment will determine the motion.

Once we have chosen a control goal consistent with the task and environment, we can use feedback control to achieve it. Feedback control uses position, velocity, and force sensors to measure the actual behavior of the robot, compares it with the desired behavior, and modulates the control signals sent to the actuators. Feedback is used in nearly all robot systems.

In this chapter we focus on: feedback control for motion control, both in the joint space and in the task space; force control; hybrid motion–force control; and impedance control.

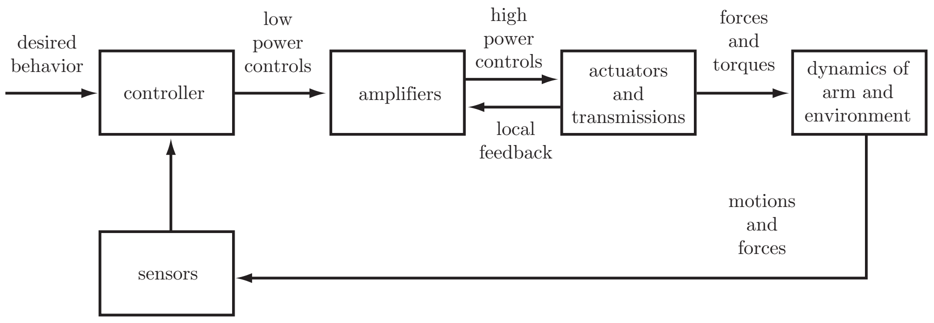
11.1 Control System Overview

A typical control block diagram is shown in Figure 11.1(a). The sensors are typically: potentiometers, encoders, or resolvers for joint position and angle sensing; tachometers for joint velocity sensing; joint force–torque sensors; and/or multi-axis force–torque sensors at the "wrist" between the end of the arm and the end-effector. The controller samples the sensors and updates its control signals to the actuators at a rate of hundreds to a few thousands of Hz. In most robotic applications control update rates higher than this are of limited benefit, given the time constants associated with the dynamics of the robot and environment. In our analysis we will ignore the fact that the sampling time is nonzero and treat controllers as if they were implemented in continuous time.

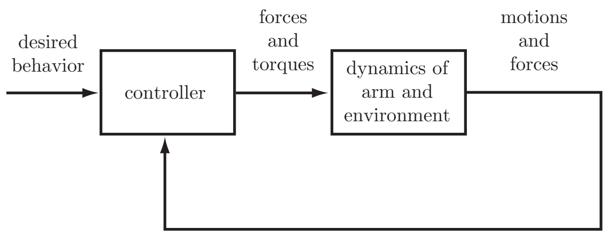
While tachometers can be used for direct velocity sensing, a common approach is to use a digital filter to numerically-difference the position signals at successive timesteps. A low-pass filter is often used in combination with the differencing filter to reduce the high-frequency signal content due to quantization of the differenced position signals.

As discussed in Section 8.9, there are a number of different technologies for creating mechanical power, transforming the speeds and forces, and transmitting to the robot joints. In this chapter we lump each joint's amplifier, actuator, and transmission together and treat them as a transformer from low-power control signals to forces and torques. This assumption, along with the assumption of perfect sensors, allows us to simplify the block diagram of Figure 11.1(a) to the one shown in Figure 11.1(b), where the controller produces the forces and torques directly. The rest of this chapter deals with the control algorithms that go inside the "controller" box in Figure 11.1(b).

Real robot systems are subject to flexibility and vibrations in the joints and links, backlash at the gears and transmissions, actuator saturation limits, and limited resolution of the sensors. These raise significant issues in design and control but they are beyond the scope of this chapter.



(a)



(b)

Figure 11.1: (a) A typical robot control system. An inner control loop is used to help the amplifier and actuator to achieve the desired force or torque. For example, a DC motor amplifier in torque control mode may sense the current actually flowing through the motor and implement a local controller to better match the desired current, since the current is proportional to the torque produced by the motor. Alternatively the motor controller may directly sense the torque by using a strain gauge on the motor's output gearing, and close a local torque-control loop using that feedback. (b) A simplified model with ideal sensors and a controller block that directly produces forces and torques. This assumes ideal behavior of the amplifier and actuator blocks in part (a). Not shown are the disturbance forces that can be injected before the dynamics block, or disturbance forces or motions injected after the dynamics block.

11.2 Error Dynamics

In this section we focus on the controlled dynamics of a single joint, as the concepts generalize easily to the case of a multi-joint robot.

If the desired joint position is $\theta_d(t)$ and the actual joint position is $\theta(t)$ then we define the joint error to be

$$\theta_e(t) = \theta_d(t) - \theta(t).$$

The differential equation governing the evolution of the joint error $\theta_e(t)$ of the controlled system is called the **error dynamics**. The purpose of the feedback controller is to create an error dynamics such that $\theta_e(t)$ tends to zero, or a small value, as t increases.

11.2.1 Error Response

A common way to test how well a controller works is to specify a nonzero initial error $\theta_e(0)$ and see how quickly, and how completely, the controller reduces the initial error. We define the (unit) **error response** to be the response $\theta_e(t), t > 0$, of the controlled system for the initial conditions $\theta_e(0) = 1$ and $\dot{\theta}_e(0) = \ddot{\theta}_e(0) = \cdots = 0$.

An ideal controller would drive the error to zero instantly and keep the error at zero for all time. In practice it takes time to reduce the error, and the error may never be completely eliminated. As illustrated in Figure 11.2, a typical error response $\theta_e(t)$ can be described by a **transient response** and a **steady-state response**. The steady-state response is characterized by the **steady-state error** e_{ss}, which is the asymptotic error $\theta_e(t)$ as $t \to \infty$. The transient response is characterized by the **overshoot** and (2%) **settling time**. The 2% settling time is the first time T such that $|\theta_e(t) - e_{\text{ss}}| \leq 0.02(\theta_e(0) - e_{\text{ss}})$ for all $t \geq T$ (see the pair of long-dashed lines). Overshoot occurs if the error response initially overshoots the final steady-state error, and in this case the overshoot is defined as

$$\text{overshoot} = \left| \frac{\theta_{e,\min} - e_{\text{ss}}}{\theta_e(0) - e_{\text{ss}}} \right| \times 100\%,$$

where $\theta_{e,\min}$ is the least positive value achieved by the error.

A good error response is characterized by

- little or no steady-state error,
- little or no overshoot, and
- a short 2% settling time.

11.2.2 Linear Error Dynamics

In this chapter we work primarily with **linear** systems with error dynamics described by linear ordinary differential equations of the form

$$a_p \theta_e^{(p)} + a_{p-1} \theta_e^{(p-1)} + \cdots + a_2 \ddot{\theta}_e + a_1 \dot{\theta}_e + a_0 \theta_e = c. \tag{11.1}$$

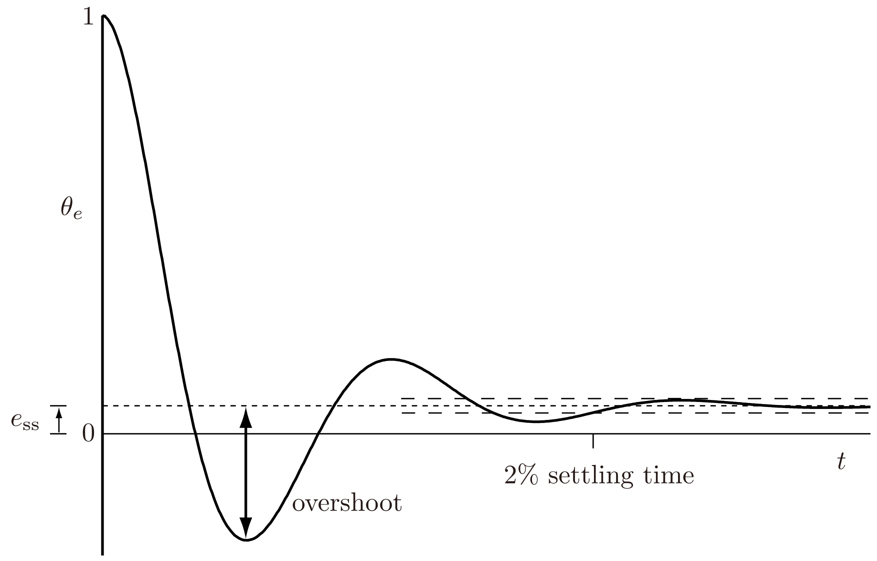


Figure 11.2: An example error response showing steady-state error e_{ss}, the overshoot, and the 2% settling time.

This is a pth-order differential equation, because p time derivatives of θ_e are present. The differential equation (11.1) is **homogeneous** if the constant c is zero and **nonhomogeneous** if $c \neq 0$.

For homogeneous ($c = 0$) linear error dynamics, the pth-order differential equation (11.1) can be rewritten as

$$\begin{aligned}\theta_e^{(p)} &= -\frac{1}{a_p}(a_{p-1}\theta_e^{(p-1)} + \cdots + a_2\ddot{\theta}_e + a_1\dot{\theta}_e + a_0\theta_e) \\ &= -a'_{p-1}\theta_e^{(p-1)} - \cdots - a'_2\ddot{\theta}_e - a'_1\dot{\theta}_e - a'_0\theta_e. \end{aligned} \tag{11.2}$$

This pth-order differential equation can be expressed as p coupled first-order differential equations by defining the vector $x = (x_1, \ldots, x_p)$, where

$$\begin{aligned} x_1 &= \theta_e, \\ x_2 &= \dot{x}_1 = \dot{\theta}_e, \\ &\vdots \quad \vdots \\ x_p &= \dot{x}_{p-1} = \theta_e^{(p-1)}, \end{aligned}$$

and writing Equation (11.2) as

$$\dot{x}_p = -a'_0 x_1 - a'_1 x_2 - \cdots - a'_{p-1} x_p.$$

Then $\dot{x}(t) = Ax(t)$, where

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a'_0 & -a'_1 & -a'_2 & \cdots & -a'_{p-2} & -a'_{p-1} \end{bmatrix} \in \mathbb{R}^{p \times p}.$$

By analogy with the scalar first-order differential equation $\dot{x}(t) = ax(t)$, which has solution $x(t) = e^{at}x(0)$, the vector differential equation $\dot{x}(t) = Ax(t)$ has solution $x(t) = e^{At}x(0)$ using the matrix exponential, as we saw in Section 3.2.3.1. Also analogous to the scalar differential equation, whose solution converges to the equilibrium $x = 0$ from any initial condition if a is negative, the differential equation $\dot{x}(t) = Ax(t)$ converges to $x = 0$ if the matrix A is negative definite, i.e., all eigenvalues of A (which may be complex) have negative real components.

The eigenvalues of A are given by the roots of the characteristic polynomial of A, i.e., the complex values s satisfying

$$\det(sI - A) = s^p + a'_{p-1}s^{p-1} + \cdots + a'_2 s^2 + a'_1 s + a'_0 = 0. \tag{11.3}$$

Equation (11.3) is also the characteristic equation associated with the pth-order differential equation (11.1).

A necessary condition for each root of Equation (11.3) to have a negative real component is that all coefficients $a'_0, \ldots, a'_{p-1}$ must be positive. This condition is also sufficient for $p = 1$ and 2. For $p = 3$, the condition $a'_2 a'_1 > a'_0$ must also hold. For higher-order systems, other conditions must hold.

If each root of Equation (11.3) has a negative real component, we call the error dynamics **stable**. If any of the roots has a positive real component, the error dynamics are **unstable**, and the error $\|\theta_e(t)\|$ can grow without bound as $t \to \infty$.

For second-order error dynamics, a good mechanical analogy to keep in mind is the linear mass–spring–damper (Figure 11.3). The position of the mass $\mathfrak{m}$ is θ_e and an external force f is applied to the mass. The damper applies a force $-b\dot{\theta}_e$ to the mass, where b is the damping constant, and the spring applies a force $-k\theta_e$ to the mass, where k is the spring constant. Therefore the equation of motion of the mass can be written as

$$\mathfrak{m}\ddot{\theta}_e + b\dot{\theta}_e + k\theta_e = f. \tag{11.4}$$

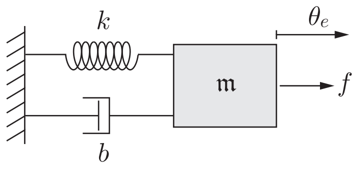


Figure 11.3: A linear mass–spring–damper.

In the limit as the mass $\mathfrak{m}$ approaches zero, the second-order dynamics (11.4) reduces to the first-order dynamics

$$b\dot{\theta}_e + k\theta_e = f. \tag{11.5}$$

By the first-order dynamics, an external force generates a velocity rather than an acceleration.

In the following subsections we consider the first- and second-order error responses for the homogeneous case ($f = 0$) with $b, k > 0$, ensuring that the error dynamics are stable and that the error converges to zero ($e_{\text{ss}} = 0$).

11.2.2.1 First-Order Error Dynamics

The first-order error dynamics (11.5) with $f = 0$ can be written in the form

$$\dot{\theta}_e(t) + \frac{k}{b}\theta_e(t) = 0$$

or

$$\dot{\theta}_e(t) + \frac{1}{\mathfrak{t}}\theta_e(t) = 0, \tag{11.6}$$

where $\mathfrak{t} = b/k$ is called the **time constant** of the first-order differential equation. The solution to the differential equation (11.6) is

$$\theta_e(t) = e^{-t/\mathfrak{t}}\theta_e(0). \tag{11.7}$$

The time constant $\mathfrak{t}$ is the time at which the first-order exponential decay has decayed to approximately 37% of its initial value. The error response $\theta_e(t)$ is defined by the initial condition $\theta_e(0) = 1$. Plots of the error response are shown in Figure 11.4 for different time constants. The steady-state error is zero, there is no overshoot in the decaying exponential error response, and the 2% settling time is determined by solving

$$\frac{\theta_e(t)}{\theta_e(0)} = 0.02 = e^{-t/\mathfrak{t}}$$

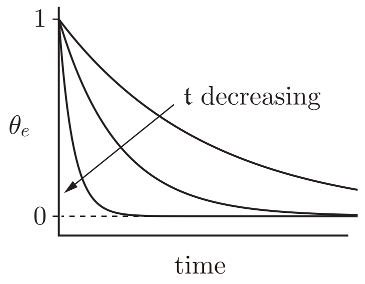


Figure 11.4: The first-order error response for three different time constants $\mathfrak{t}$.

for t. Solving, we get

$$\ln 0.02 = -t/\mathfrak{t} \quad \rightarrow \quad t = 3.91\mathfrak{t},$$

a 2% settling time of approximately $4\mathfrak{t}$. The response gets faster as the spring constant k increases or the damping constant b decreases.

11.2.2.2 Second-Order Error Dynamics

The second-order error dynamics

$$\ddot{\theta}_e(t) + \frac{b}{\mathfrak{m}}\dot{\theta}_e(t) + \frac{k}{\mathfrak{m}}\theta_e(t) = 0$$

can be written in the **standard second-order form**

$$\ddot{\theta}_e(t) + 2\zeta\omega_n\dot{\theta}_e(t) + \omega_n^2\theta_e(t) = 0, \tag{11.8}$$

where ω_n is called the **natural frequency** and ζ is called the **damping ratio**. For the mass–spring–damper, $\omega_n = \sqrt{k/\mathfrak{m}}$ and $\zeta = b/(2\sqrt{k\mathfrak{m}})$. The two roots of the characteristic polynomial

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{11.9}$$

are

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1} \qquad \text{and} \qquad s_2 = -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1}. \tag{11.10}$$

The second-order error dynamics (11.8) is stable if and only if $\zeta\omega_n > 0$ and $\omega_n^2 > 0$.

If the error dynamics is stable, then there are three types of solutions $\theta_e(t)$ to the differential equation, depending on whether the roots $s_{1,2}$ are real and unequal ($\zeta > 1$), real and equal ($\zeta = 1$), or complex conjugates ($\zeta < 1$).

- **Overdamped**: $\zeta > 1$. The roots $s_{1,2}$ are real and distinct, and the solution to the differential equation (11.8) is

  $$\theta_e(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t},$$

  where c_1 and c_2 can be calculated from the initial conditions. The response is the sum of two decaying exponentials, with time constants $\mathfrak{t}_1 = -1/s_1$ and $\mathfrak{t}_2 = -1/s_2$. The "slower" time constant in the solution is given by the less negative root, $s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1}$.

  The initial conditions for the (unit) error response are $\theta_e(0) = 1$ and $\dot{\theta}_e(0) = 0$, and the constants c_1 and c_2 can be calculated as

  $$c_1 = \frac{1}{2} + \frac{\zeta}{2\sqrt{\zeta^2 - 1}} \qquad \text{and} \qquad c_2 = \frac{1}{2} - \frac{\zeta}{2\sqrt{\zeta^2 - 1}}.$$

- **Critically damped**: $\zeta = 1$. The roots $s_{1,2} = -\omega_n$ are equal and real, and the solution is

  $$\theta_e(t) = (c_1 + c_2 t)e^{-\omega_n t},$$

  i.e., a decaying exponential multiplied by a linear function of time. The time constant of the decaying exponential is $\mathfrak{t} = 1/\omega_n$. For the error response with $\theta_e(0) = 1$ and $\dot{\theta}_e(0) = 0$,

  $$c_1 = 1 \qquad \text{and} \qquad c_2 = \omega_n.$$

- **Underdamped**: $\zeta < 1$. The roots $s_{1,2}$ are complex conjugates at $s_{1,2} = -\zeta\omega_n \pm j\omega_d$, where $\omega_d = \omega_n\sqrt{1 - \zeta^2}$ is the **damped natural frequency**. The solution is

  $$\theta_e(t) = (c_1 \cos \omega_d t + c_2 \sin \omega_d t)\, e^{-\zeta\omega_n t},$$

  i.e., a decaying exponential (time constant $\mathfrak{t} = 1/(\zeta\omega_n)$) multiplied by a sinusoid. For the error response with $\theta_e(0) = 1$ and $\dot{\theta}_e(0) = 0$,

  $$c_1 = 1 \qquad \text{and} \qquad c_2 = \frac{\zeta}{\sqrt{1 - \zeta^2}}.$$

Example root locations for the overdamped, critically damped, and underdamped cases, as well as their error responses $\theta_e(t)$, are shown in Figure 11.5. This figure also shows the relationship between the root locations and properties of the transient response: roots further to the left in the complex plane

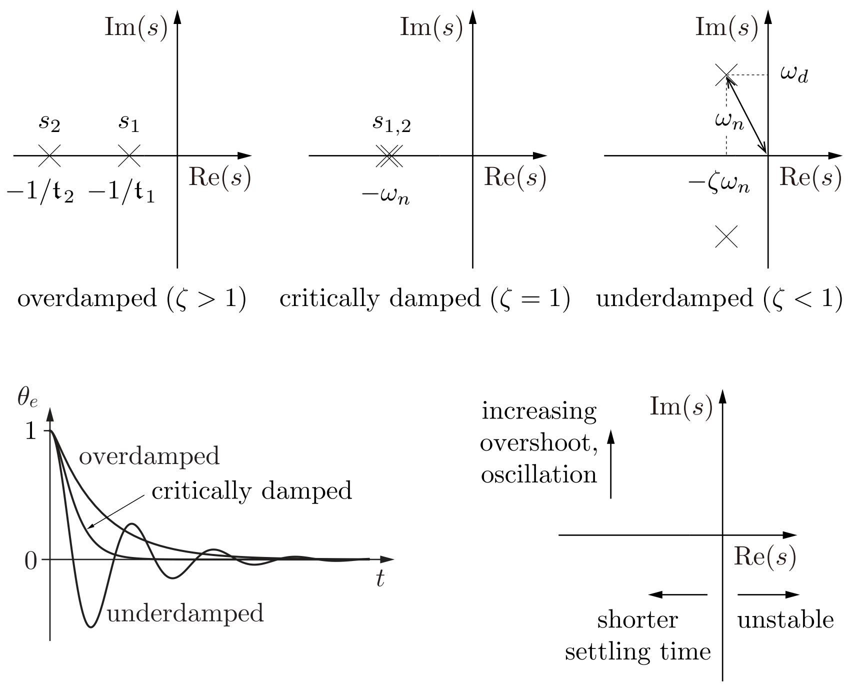


Figure 11.5: (Top) Example root locations for overdamped, critically damped, and underdamped second-order systems. (Bottom left) Error responses for overdamped, critically damped, and underdamped second-order systems. (Bottom right) Relationship of the root locations to properties of the transient response.

correspond to shorter settling times, and roots further away from the real axis correspond to greater overshoot and oscillation. These general relationships between root locations and transient response properties also hold for higher-order systems with more than two roots.

If the second-order error dynamics (11.8) is stable, the steady-state error e_{ss} is zero regardless of whether the error dynamics is overdamped, underdamped, or critically damped. The 2% settling time is approximately $4\mathfrak{t}$, where $\mathfrak{t}$ corresponds to the "slower" root s_1 if the error dynamics is overdamped. The overshoot is zero for overdamped and critically damped error dynamics and, for underdamped error dynamics, the overshoot can be calculated by finding the first time (after $t = 0$) where the error response satisfies $\dot{\theta}_e = 0$. This is the peak of the overshoot, and it occurs at

$$t_p = \pi/\omega_d.$$

Substituting this into the underdamped error response, we get

$$\begin{aligned}\theta_e(t_p) = \theta_e\left(\frac{\pi}{\omega_d}\right) &= \left(\cos\left(\omega_d \frac{\pi}{\omega_d}\right) + \frac{\zeta}{\sqrt{1-\zeta^2}} \sin\left(\omega_d \frac{\pi}{\omega_d}\right)\right) e^{-\zeta\omega_n \pi/\omega_d} \\ &= -e^{-\pi\zeta/\sqrt{1-\zeta^2}}.\end{aligned}$$

Therefore, by our definition of overshoot, the overshoot is $e^{-\pi\zeta/\sqrt{1-\zeta^2}} \times 100\%$. Thus $\zeta = 0.1$ gives an overshoot of 73%, $\zeta = 0.5$ gives an overshoot of 16%, and $\zeta = 0.8$ gives an overshoot of 1.5%.

11.3 Motion Control with Velocity Inputs

As discussed in Chapter 8, we typically assume that there is direct control of the forces or torques at robot joints, and the robot's dynamics transforms those controls to joint accelerations. In some cases, however, we can assume that there is direct control of the joint velocities, for example when the actuators are stepper motors. In this case the velocity of a joint is determined directly by the frequency of the pulse train sent to the stepper.[1] Another example occurs when the amplifier for an electric motor is placed in velocity control mode – the amplifier attempts to achieve the joint velocity requested by the user, rather than a joint force or torque.

In this section we will assume that the control inputs are joint velocities. In Section 11.4, and indeed in the rest of the chapter, the control inputs are assumed to be joint forces and torques.

The motion control task can be expressed in joint space or task space. When the trajectory is expressed in task space, the controller is fed a steady stream of end-effector configurations $X_d(t)$, and the goal is to command joint velocities that cause the robot to track this trajectory. In joint space, the controller is fed a steady stream of desired joint positions $\theta_d(t)$.

The main ideas are well illustrated by a robot with a single joint, so we begin there and then generalize to a multi-joint robot.

[1]This assumes that the torque requirements are low enough that the stepper motor can keep up with the pulse train.

11.3.1 Motion Control of a Single Joint

11.3.1.1 Feedforward Control

Given a desired joint trajectory $\theta_d(t)$, the simplest type of control would be to choose the commanded velocity $\dot{\theta}(t)$ as

$$\dot{\theta}(t) = \dot{\theta}_d(t), \tag{11.11}$$

where $\dot{\theta}_d(t)$ comes from the desired trajectory. This is called a **feedforward** or **open-loop controller**, since no feedback (sensor data) is needed to implement it.

11.3.1.2 Feedback Control

In practice, position errors will accumulate over time under the feedforward control law (11.11). An alternative strategy is to measure the actual position of each joint continually and implement a **feedback controller**.

P Control and First-Order Error Dynamics The simplest feedback controller is

$$\dot{\theta}(t) = K_p(\theta_d(t) - \theta(t)) = K_p\theta_e(t), \tag{11.12}$$

where $K_p > 0$. This controller is called a proportional controller, or **P controller**, because it creates a corrective control proportional to the position error $\theta_e(t) = \theta_d(t) - \theta(t)$. In other words, the constant **control gain** K_p acts somewhat like a virtual spring that tries to pull the actual joint position to the desired joint position.

The P controller is an example of a **linear controller**, as it creates a control signal that is a linear combination of the error $\theta_e(t)$ and possibly its time derivatives and time integrals.

The case where $\theta_d(t)$ is constant, i.e., $\dot{\theta}_d(t) = 0$, is called **setpoint control**. In setpoint control, the error dynamics

$$\dot{\theta}_e(t) = \cancelto{0}{\dot{\theta}_d(t)} - \dot{\theta}(t)$$

is written as follows after substituting in the P controller $\dot{\theta}(t) = K_p\theta_e(t)$:

$$\dot{\theta}_e(t) = -K_p\theta_e(t) \quad \rightarrow \quad \dot{\theta}_e(t) + K_p\theta_e(t) = 0.$$

This is a first-order error dynamic equation (11.6) with time constant $\mathfrak{t} = 1/K_p$. The decaying exponential error response is illustrated in Figure 11.4. The steady-state error is zero, there is no overshoot, and the 2% settling time is $4/K_p$. A larger K_p means a faster response.

Now consider the case where $\theta_d(t)$ is not constant but $\dot{\theta}_d(t)$ is constant, i.e., $\dot{\theta}_d(t) = c$. Then the error dynamics under the P controller can be written

$$\dot{\theta}_e(t) = \dot{\theta}_d(t) - \dot{\theta}(t) = c - K_p\theta_e(t),$$

which we rewrite as

$$\dot{\theta}_e(t) + K_p\theta_e(t) = c.$$

This is a first-order nonhomogeneous linear differential equation with solution

$$\theta_e(t) = \frac{c}{K_p} + \left(\theta_e(0) - \frac{c}{K_p}\right) e^{-K_p t},$$

which converges to the nonzero value c/K_p as time goes to infinity. Unlike the case of setpoint control, the steady-state error e_{ss} is nonzero; the joint position always lags behind the moving reference. The steady-state error c/K_p can be made small by choosing the control gain K_p large, but there are practical limits on how large K_p can be. For one thing, real joints have velocity limits that may prevent the realization of the large commanded velocities associated with a large K_p. For another, large values of K_p may cause instability when implemented by a discrete-time digital controller – the large gain may result in a large change in θ_e during a single servo cycle, meaning that the control action late in the servo cycle is in response to sensor data that is no longer relevant.

PI Control and Second-Order Error Dynamics An alternative to using a large gain K_p is to introduce another term in the control law. A proportional-integral controller, or **PI controller**, adds a term that is proportional to the time-integral of the error:

$$\dot{\theta}(t) = K_p\theta_e(t) + K_i \int_0^t \theta_e(\mathrm{t})\, d\mathrm{t}, \tag{11.13}$$

where t is the current time and t is the variable of integration. The PI controller block diagram is illustrated in Figure 11.6.

With this controller, the error dynamics for a constant $\dot{\theta}_d(t)$ becomes

$$\dot{\theta}_e(t) + K_p\theta_e(t) + K_i \int_0^t \theta_e(\mathrm{t})\, d\mathrm{t} = c.$$

Taking the time derivative of this dynamics, we get

$$\ddot{\theta}_e(t) + K_p\dot{\theta}_e(t) + K_i\theta_e(t) = 0. \tag{11.14}$$

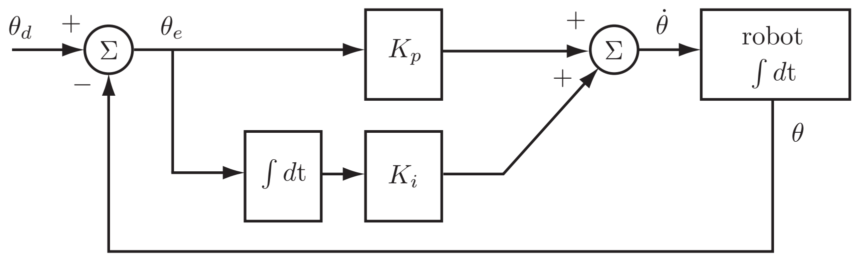


Figure 11.6: The block diagram of a PI controller that produces a commanded velocity $\dot{\theta}$ as input to the robot.

We can rewrite this equation in the standard second-order form (11.8), with natural frequency $\omega_n = \sqrt{K_i}$ and damping ratio $\zeta = K_p/(2\sqrt{K_i})$.

Relating the PI controller of Equation (11.14) to the mass–spring–damper of Figure 11.3, the gain K_p plays the role of $b/\mathfrak{m}$ for the mass–spring–damper (a larger K_p means a larger damping constant b), and the gain K_i plays the role of $k/\mathfrak{m}$ (a larger K_i means a larger spring constant k).

The PI-controlled error dynamics equation is stable if $K_i > 0$ and $K_p > 0$, and the roots of the characteristic equation are

$$s_{1,2} = -\frac{K_p}{2} \pm \sqrt{\frac{K_p^2}{4} - K_i}.$$

Let's hold K_p equal to 20 and plot the roots in the complex plane as K_i grows from zero (Figure 11.7). This plot, or any plot of the roots as one parameter is varied, is called a **root locus**.

For $K_i = 0$, the characteristic equation $s^2+K_p s+K_i = s^2+20s = s(s+20) = 0$ has roots at $s_1 = 0$ and $s_2 = -20$. As K_i increases, the roots move toward each other on the real axis of the s-plane as shown in the left-hand panel in Figure 11.7. Because the roots are real and unequal, the error dynamics equation is overdamped ($\zeta = K_p/(2\sqrt{K_i}) > 1$, case I) and the error response is sluggish due to the time constant $\mathfrak{t}_1 = -1/s_1$ of the exponential corresponding to the "slow" root. As K_i increases, the damping ratio decreases, the "slow" root moves left (while the "fast" root moves right), and the response gets faster. When K_i reaches 100, the two roots meet at $s_{1,2} = -10 = -\omega_n = K_p/2$, and the error dynamics equation is critically damped ($\zeta = 1$, case II). The error response has a short 2% settling time of $4\mathfrak{t} = 4/(\zeta\omega_n) = 0.4$ s and no overshoot or oscillation. As K_i continues to grow, the damping ratio ζ falls below 1 and the roots move vertically off the real axis, becoming complex conjugates at $s_{1,2} = -10 \pm j\sqrt{K_i - 100}$ (case III). The error dynamics is underdamped, and

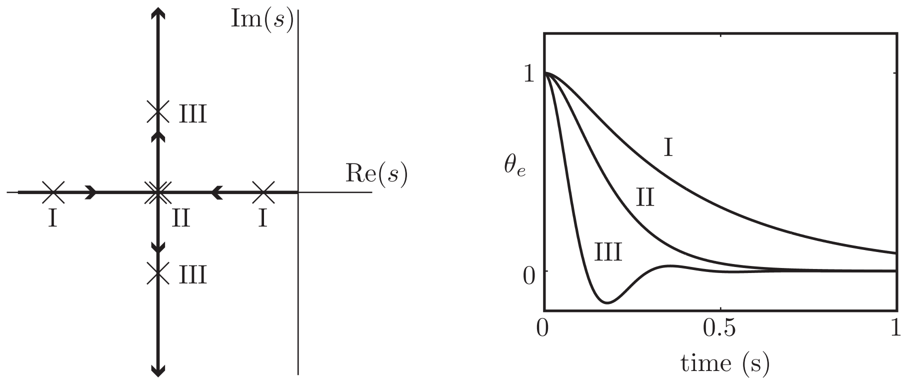


Figure 11.7: (Left) The complex roots of the characteristic equation of the error dynamics of the PI velocity-controlled joint for a fixed $K_p = 20$ as K_i increases from zero. This is known as a root locus plot. (Right) The error response to an initial error $\theta_e = 1$, $\dot{\theta}_e = 0$, is shown for overdamped ($\zeta = 1.5$, $K_i = 44.4$, case I), critically damped ($\zeta = 1$, $K_i = 100$, case II), and underdamped ($\zeta = 0.5$, $K_i = 400$, case III) cases.

the response begins to exhibit overshoot and oscillation as K_i increases. The settling time is unaffected as the time constant $\mathfrak{t} = 1/(\zeta\omega_n)$ remains constant.

According to our simple model of the PI controller, we could always choose K_p and K_i for critical damping ($K_i = K_p^2/4$) and increase K_p and K_i without bound to make the error response arbitrarily fast. As described above, however, there are practical limits. Within these practical limits, K_p and K_i should be chosen to yield critical damping.

Figure 11.8 shows for comparison the performances of a P controller and a PI controller attempting to track a constant-velocity trajectory. The proportional gain K_p is the same in both cases, while $K_i = 0$ for the P controller. From the shape of the response, it appears that K_i in the PI controller was chosen to be a bit too large, making the system underdamped. It is also clear that $e_{\text{ss}} = 0$ for the PI controller but $e_{\text{ss}} \neq 0$ for the P controller, agreeing with our analysis above.

If the desired velocity $\dot{\theta}_d(t)$ is anything other than constant, the PI controller cannot be expected to eliminate steady-state error completely. If it changes slowly, however, then a well-designed PI controller can be expected to provide better tracking performance than a P controller.

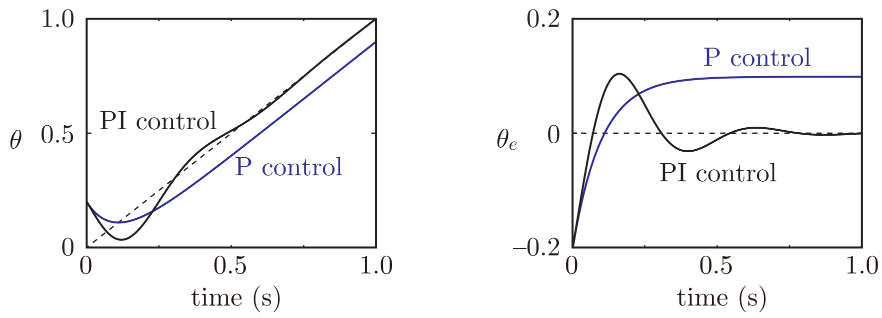


Figure 11.8: The motion of P-controlled and PI-controlled joints, with initial position error, tracking a reference trajectory (dashed) where $\dot{\theta}_d(t)$ is constant. (Left) The responses $\theta(t)$. (Right) The error responses $\theta_e(t) = \theta_d(t) - \theta(t)$.

11.3.1.3 Feedforward Plus Feedback Control

A drawback of feedback control is that an error is required before the joint begins to move. It would be preferable to use our knowledge of the desired trajectory $\theta_d(t)$ to initiate motion before any error accumulates.

We can combine the advantages of feedforward control, which commands motion even when there is no error, with the advantages of feedback control, which limits the accumulation of error, as follows:

$$\boxed{\dot{\theta}(t) = \dot{\theta}_d(t) + K_p\theta_e(t) + K_i \int_0^t \theta_e(\mathrm{t})\, d\mathrm{t}.} \tag{11.15}$$

This feedforward–feedback controller, illustrated in Figure 11.9, is our preferred control law for producing a commanded velocity to the joint.

11.3.2 Motion Control of a Multi-joint Robot

The single-joint PI feedback plus feedforward controller (11.15) generalizes immediately to robots with n joints. The reference position $\theta_d(t)$ and actual position $\theta(t)$ are now n-vectors, and the gains K_p and K_i are diagonal $n \times n$ matrices of the form k_pI and k_iI, where the scalars k_p and k_i are positive and I is the $n \times n$ identity matrix. Each joint is subject to the same stability and performance analysis as the single joint in Section 11.3.1.

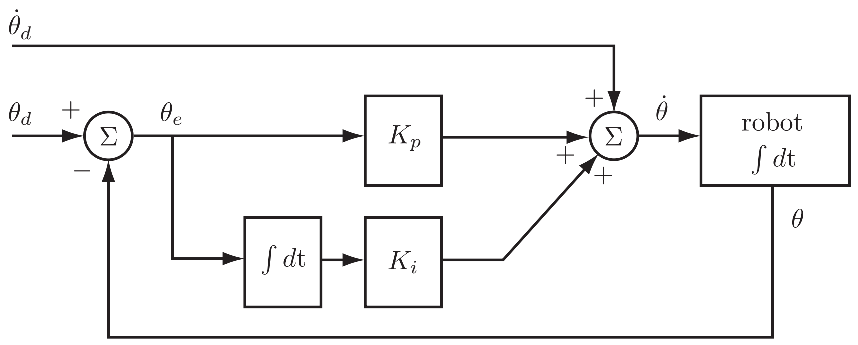


Figure 11.9: The block diagram of feedforward plus PI feedback control that produces a commanded velocity $\dot{\theta}$ as input to the robot.

11.3.3 Task-Space Motion Control

We can express the feedforward plus feedback control law in task space. Let $X(t) \in SE(3)$ be the configuration of the end-effector as a function of time and $\mathcal{V}_b(t)$ be the end-effector twist expressed in the end-effector frame $\{\mathrm{b}\}$, i.e., $[\mathcal{V}_b] = X^{-1}\dot{X}$. The desired motion is given by $X_d(t)$ and $[\mathcal{V}_d] = X_d^{-1}\dot{X}_d$. A task-space version of the control law (11.15) is

$$\boxed{\mathcal{V}_b(t) = [\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d(t) + K_p X_e(t) + K_i \int_0^t X_e(\mathrm{t})\, dt.} \tag{11.16}$$

The term $[\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d$ expresses the feedforward twist $\mathcal{V}_d$ in the actual end-effector frame at X (which could also be written X_{sb}) rather than the desired end-effector frame X_d (which could also be written X_{sd}). When the end-effector is at the desired configuration ($X = X_d$), this term reduces to $\mathcal{V}_d$. Also, the configuration error $X_e(t)$ is not simply $X_d(t) - X(t)$, since it does not make sense to subtract elements of $SE(3)$. Instead, as we saw in Section 6.2, X_e should refer to the twist which, if followed for unit time, takes X to X_d. The $se(3)$ representation of this twist, expressed in the end-effector frame, is $[X_e] = \log(X^{-1}X_d)$.

As in Section 11.3.2, the diagonal gain matrices $K_p, K_i \in \mathbb{R}^{6\times 6}$ take the form $k_p I$ and $k_i I$, respectively, where $k_p, k_i > 0$.

The commanded joint velocities $\dot{\theta}$ realizing $\mathcal{V}_b$ from the control law (11.16) can be calculated using the inverse velocity kinematics from Section 6.3,

$$\dot{\theta} = J_b^{\dagger}(\theta)\mathcal{V}_b,$$

where $J_b^{\dagger}(\theta)$ is the pseudoinverse of the body Jacobian.

Motion control in task space can be defined using other representations of the end-effector configuration and velocity. For example, for a minimal coordinate representation of the end-effector configuration $x \in \mathbb{R}^m$, the control law can be written

$$\dot{x}(t) = \dot{x}_d(t) + K_p(x_d(t) - x(t)) + K_i \int_0^t (x_d(\mathrm{t}) - x(\mathrm{t}))\, dt. \tag{11.17}$$

For a hybrid configuration representation $X = (R, p)$, with velocities represented by $(\omega_b, \dot{p})$:

$$\begin{bmatrix} \omega_b(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} R^{\mathrm{T}}(t)R_d(t) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \omega_d(t) \\ \dot{p}_d(t) \end{bmatrix} + K_p X_e(t) + K_i \int_0^t X_e(\mathrm{t})\, d\mathrm{t}, \tag{11.18}$$

where

$$X_e(t) = \begin{bmatrix} \log(R^{\mathrm{T}}(d)R_d(t)) \\ p_d(t) - p(t) \end{bmatrix}.$$

Figure 11.10 shows the performance of the control law (11.16), where the end-effector velocity is the body twist $\mathcal{V}_b$, and the performance of the control law (11.18), where the end-effector velocity is $(\omega_b, \dot{p})$. The control task is to stabilize X_d at the origin from the initial configuration

$$R_0 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad p_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The feedforward velocity is zero and $K_i = 0$. Figure 11.10 shows the different paths followed by the end-effector. The decoupling of linear and angular control in the control law (11.18) is visible in the straight-line motion of the origin of the end-effector frame.

An application of the control law (11.16) to mobile manipulation can be found in Section 13.5.

11.4 Motion Control with Torque or Force Inputs

Stepper-motor-controlled robots are generally limited to applications with low or predictable force–torque requirements. Also, robot-control engineers do not rely on the velocity-control modes of off-the-shelf amplifiers for electric motors, because these velocity-control algorithms do not make use of a dynamic model of the robot. Instead, robot-control engineers use amplifiers in torque-control

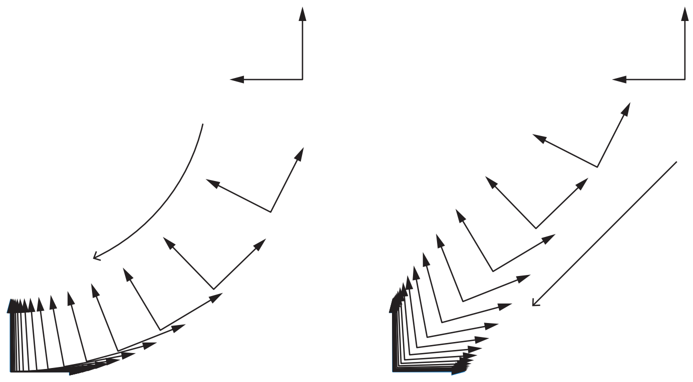


Figure 11.10: (Left) The end-effector configuration converging to the origin under the control law (11.16), where the end-effector velocity is represented as the body twist $\mathcal{V}_b$. (Right) The end-effector configuration converging to the origin under the control law (11.18), where the end-effector velocity is represented as $(\omega_b, \dot{p})$.

mode: the input to the amplifier is the desired torque (or force). This allows the robot-control engineer to use a dynamic model of the robot in the design of the control law.

In this section, the controller generates joint torques and forces to try to track a desired trajectory in joint space or task space. Once again, the main ideas are well illustrated by a robot with a single joint, so we begin there and then generalize to a multi-joint robot.

11.4.1 Motion Control of a Single Joint

Consider a single motor attached to a single link, as shown in Figure 11.11. Let τ be the motor's torque and θ be the angle of the link. The dynamics can be written as

$$\tau = M\ddot{\theta} + \mathfrak{m}gr\cos\theta, \tag{11.19}$$

where M is the scalar inertia of the link about the axis of rotation, $\mathfrak{m}$ is the mass of the link, r is the distance from the axis to the center of mass of the link, and $g \geq 0$ is the gravitational acceleration.

According to the model (11.19) there is no dissipation: if the link were made to move and τ were then set to zero, the link would move forever. This is unrealistic, of course; there is bound to be friction at the various bearings, gears, and transmissions. Friction modeling is an active research area, but in a

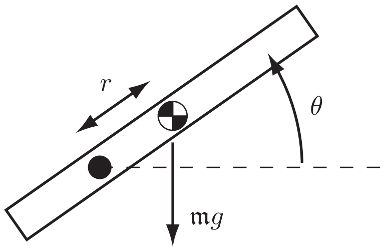


Figure 11.11: A single-joint robot rotating under gravity. The center of mass is indicated by the checkered disk.

simple model, rotational friction is due to viscous friction forces, so that

$$\tau_{\text{fric}} = b\dot{\theta}, \tag{11.20}$$

where $b > 0$. Adding the friction torque, our final model is

$$\tau = M\ddot{\theta} + \mathfrak{m}gr\cos\theta + b\dot{\theta}, \tag{11.21}$$

which we may write more compactly as

$$\tau = M\ddot{\theta} + h(\theta, \dot{\theta}), \tag{11.22}$$

where h contains all terms that depend only on the state, not the acceleration.

For concreteness in the following simulations, we set $M = 0.5$ kg m^2, $\mathfrak{m} = 1$ kg, $r = 0.1$ m, and $b = 0.1$ N m s/rad. In some examples the link moves in a horizontal plane, so $g = 0$. In other examples, the link moves in a vertical plane, so $g = 9.81$ m/s^2.

11.4.1.1 Feedback Control: PID Control

A common feedback controller is linear proportional-integral-derivative control, or **PID control**. The PID controller is simply the PI controller (Equation (11.13)) with an added term proportional to the time derivative of the error,

$$\boxed{\tau = K_p\theta_e + K_i\int \theta_e(\mathrm{t})dt + K_d\dot{\theta}_e,} \tag{11.23}$$

where the control gains K_p, K_i, and K_d are positive. The proportional gain K_p acts as a virtual spring that tries to reduce the position error $\theta_e = \theta_d - \theta$. The

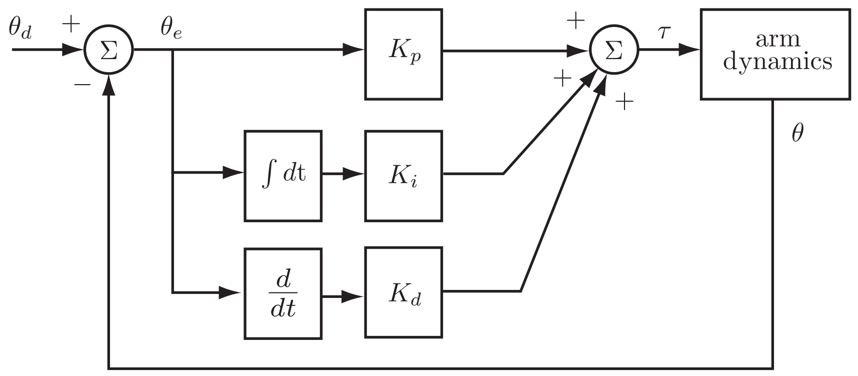


Figure 11.12: Block diagram of a PID controller.

derivative gain K_d acts as a virtual damper that tries to reduce the velocity error $\dot{\theta}_e = \dot{\theta}_d - \dot{\theta}$. The integral gain can be used to reduce or eliminate steady-state errors. The PID controller block diagram is given in Figure 11.12.

PD Control and Second-Order Error Dynamics For now let's consider the case where $K_i = 0$. This is known as PD control. Let's also assume the robot moves in a horizontal plane ($g = 0$). Substituting the PD control law into the dynamics (11.21), we get

$$M\ddot{\theta} + b\dot{\theta} = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}). \tag{11.24}$$

If the control objective is setpoint control at a constant θ_d with $\dot{\theta}_d = \ddot{\theta}_d = 0$, then $\theta_e = \theta_d - \theta$, $\dot{\theta}_e = -\dot{\theta}$, and $\ddot{\theta}_e = -\ddot{\theta}$. Equation (11.24) can be rewritten as

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e = 0, \tag{11.25}$$

or, in the standard second-order form (11.8), as

$$\ddot{\theta}_e + \frac{b + K_d}{M}\dot{\theta}_e + \frac{K_p}{M}\theta_e = 0 \quad \rightarrow \quad \ddot{\theta}_e + 2\zeta\omega_n\dot{\theta}_e + \omega_n^2\theta_e = 0, \tag{11.26}$$

where the damping ratio ζ and the natural frequency ω_n are

$$\zeta = \frac{b + K_d}{2\sqrt{K_p M}} \qquad \text{and} \qquad \omega_n = \sqrt{\frac{K_p}{M}}.$$

For stability, $b + K_d$ and K_p must be positive. If the error dynamics equation is stable then the steady-state error is zero. For no overshoot and a fast response, the gains K_d and K_p should be chosen to satisfy critical damping ($\zeta = 1$). For a fast response, K_p should be chosen to be as high as possible, subject to practical issues such as actuator saturation, undesired rapid torque changes (chattering), vibrations of the structure due to unmodeled flexibility in the joints and links, and possibly even instability due to the finite servo rate frequency.

PID Control and Third-Order Error Dynamics Now consider the case of setpoint control where the link moves in a vertical plane ($g > 0$). With the PD control law above, the error dynamics can now be written

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e = \mathfrak{m}gr\cos\theta. \tag{11.27}$$

This implies that the joint comes to rest at a configuration θ satisfying $K_p\theta_e = \mathfrak{m}gr\cos\theta$, i.e., the final error θ_e is nonzero when $\theta_d \neq \pm\pi/2$. The reason is that the robot must provide a nonzero torque to hold the link at rest at $\theta \neq \pm\pi/2$, but the PD control law creates a nonzero torque at rest only if $\theta_e \neq 0$. We can make this steady-state error small by increasing the gain K_p but, as discussed above, there are practical limits.

To eliminate the steady-state error, we return to the PID controller by setting $K_i > 0$. This allows a nonzero steady-state torque even with zero position error; only the *integrated* error must be nonzero. Figure 11.13 demonstrates the effect of adding the integral term to the controller.

To see how this works, write down the setpoint error dynamics

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e + K_i\int\theta_e(\mathrm{t})d\mathrm{t} = \tau_{\text{dist}}, \tag{11.28}$$

where τ_{dist} is a disturbance torque substituted for the gravity term $\mathfrak{m}gr\cos\theta$. Taking derivatives of both sides, we get the third-order error dynamics

$$M\theta_e^{(3)} + (b + K_d)\ddot{\theta}_e + K_p\dot{\theta}_e + K_i\theta_e = \dot{\tau}_{\text{dist}}. \tag{11.29}$$

If τ_{dist} is constant then the right-hand side of Equation (11.29) is zero, and its characteristic equation is

$$s^3 + \frac{b + K_d}{M}s^2 + \frac{K_p}{M}s + \frac{K_i}{M} = 0. \tag{11.30}$$

If all roots of Equation (11.30) have a negative real part then the error dynamics is stable, and θ_e converges to zero. (While the disturbance torque due to gravity is not constant as the link rotates, it approaches a constant as $\dot{\theta}$ goes to zero, and therefore similar reasoning holds close to the equilibrium $\theta_e = 0$.)

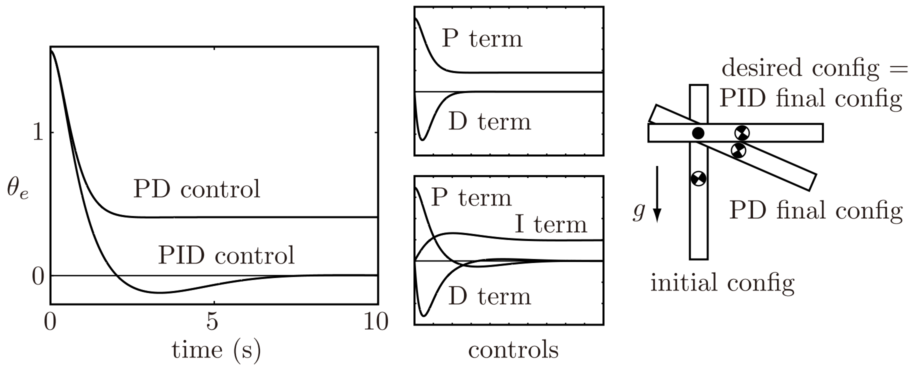


Figure 11.13: (Left) The tracking errors for a PD controller with $K_d = 2$ N m s/rad and $K_p = 2.205$ N m/rad for critical damping, and a PID controller with the same PD gains and $K_i = 1$ N m/(rad s). The arm starts at $\theta(0) = -\pi/2, \dot{\theta}(0) = 0$, with a goal state $\theta_d = 0, \dot{\theta}_d = 0$. (Middle) The individual contributions of the terms in the PD and PID control laws. Note that the nonzero I (integral) term for the PID controller allows the P (proportional) term to drop to zero. (Right) The initial and final configurations, with the center of mass indicated by checkered disks.

For all the roots of Equation (11.30) to have a negative real component, the following conditions on the control gains must be satisfied for stability (Section 11.2.2.2):

$$\begin{aligned} K_d &> -b \\ K_p &> 0 \\ \frac{(b+K_d)K_p}{M} > K_i &> 0. \end{aligned}$$

Thus the new gain K_i must satisfy both a lower *and* an upper bound (Figure 11.14). A reasonable design strategy is to choose K_p and K_d for a good transient response and then choose K_i large enough that it is helpful in reducing or eliminating steady-state errors but small enough that it does not significantly impact stability. In the example of Figure 11.13, the relatively large K_i worsens the transient response, giving significant overshoot, but steady-state error is eliminated.

In practice, $K_i = 0$ for many robot controllers, since stability is paramount. Other techniques can be employed to limit the adverse stability effects of integral control, such as **integrator anti-windup**, which places a limit on how large the error integral is allowed to grow.

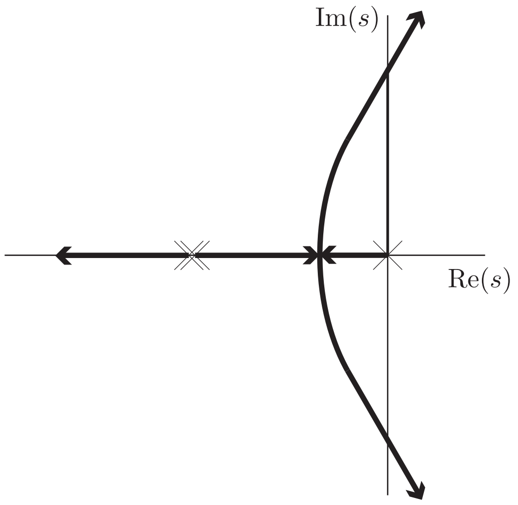


Figure 11.14: The movement of the three roots of Equation (11.30) as K_i increases from zero. First a PD controller is chosen with K_p and K_d yielding critical damping, giving rise to two collocated roots on the negative real axis. Adding an infinitesimal gain $K_i > 0$ creates a third root at the origin. As we increase K_i, one of the two collocated roots moves to the left on the negative real axis while the other two roots move toward each other, meet, break away from the real axis, begin curving to the right, and finally move into the right half-plane when $K_i = (b + K_d)K_p/M$. The system is unstable for larger values of K_i.

Pseudocode for the PID control algorithm is given in Figure 11.15.

While our analysis has focused on setpoint control, the PID controller applies perfectly well to trajectory following, where $\dot{\theta}_d(t) \neq 0$. Integral control will not eliminate tracking error along arbitrary trajectories, however.

11.4.1.2 Feedforward Control

Another strategy for trajectory following is to use a model of the robot's dynamics to proactively generate torques instead of waiting for errors. Let the controller's model of the dynamics be

$$\tau = \tilde{M}(\theta)\ddot{\theta} + \tilde{h}(\theta, \dot{\theta}), \tag{11.31}$$

where the model is perfect if $\tilde{M}(\theta) = M(\theta)$ and $\tilde{h}(\theta, \dot{\theta}) = h(\theta, \dot{\theta})$. Note that the inertia model $\tilde{M}(\theta)$ is written as a function of the configuration θ. While the inertia of our simple one-joint robot is not a function of configuration, writing

```
time = 0                          // dt = servo cycle time
eint = 0                          // error integral
qprev = senseAngle                // initial joint angle q
loop
  [qd,qdotd] = trajectory(time) // from trajectory generator

  q = senseAngle                // sense actual joint angle
  qdot = (q - qprev)/dt         // simple velocity calculation
  qprev = q

  e = qd - q
  edot = qdotd - qdot
  eint = eint + e*dt

  tau = Kp*e + Kd*edot + Ki*eint
  commandTorque(tau)

  time = time + dt
end loop
```

Figure 11.15: Pseudocode for PID control.

the equations in this way allows us to re-use Equation (11.31) for multi-joint systems in Section 11.4.2.

Given θ_d, $\dot{\theta}_d$, and $\ddot{\theta}_d$ from a trajectory generator, the feedforward torque is calculated as

$$\boxed{\tau(t) = \tilde{M}(\theta_d(t))\ddot{\theta}_d(t) + \tilde{h}(\theta_d(t), \dot{\theta}_d(t)).} \tag{11.32}$$

If the model of the robot dynamics is exact, and there are no initial state errors, then the robot follows the desired trajectory exactly.

A pseudocode implementation of feedforward control is given in Figure 11.16.

Figure 11.17 shows two examples of feedforward-trajectory following for the link under gravity. Here, the controller's dynamic model is correct except that it has $\tilde{r} = 0.08$ m, when actually $r = 0.1$ m. In Task 1 the error stays small, as unmodeled gravity effects provide a spring-like force to $\theta = -\pi/2$, accelerating the robot at the beginning and decelerating it at the end. In Task 2, unmodeled gravity effects act against the desired motion, resulting in a larger tracking error.

Because there are always modeling errors, feedforward control is always used in conjunction with feedback, as discussed next.

```
time = 0                                          // dt = servo cycle time
loop
  [qd,qdotd,qdotdotd] = trajectory(time)          // trajectory generator
  tau = Mtilde(qd)*qdotdotd + htilde(qd,qdotd)    // calculate dynamics
  commandTorque(tau)
  time = time + dt
end loop
```

Figure 11.16: Pseudocode for feedforward control.

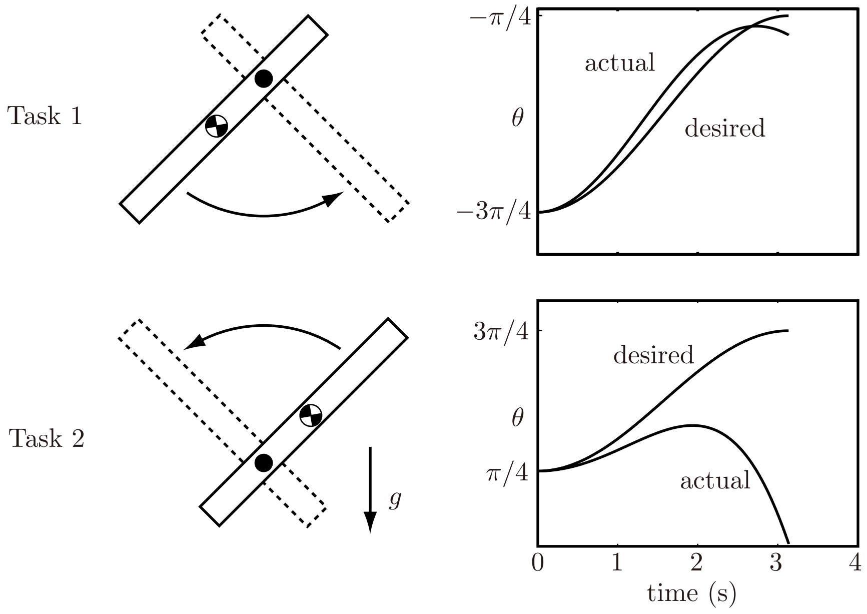


Figure 11.17: Results of feedforward control with an incorrect model: $\tilde{r} = 0.08$ m, but $r = 0.1$ m. The desired trajectory in Task 1 is $\theta_d(t) = -\pi/2 - (\pi/4)\cos(t)$ for $0 \leq t \leq \pi$. The desired trajectory for Task 2 is $\theta_d(t) = \pi/2 - (\pi/4)\cos(t), 0 \leq t \leq \pi$.

11.4.1.3 Feedforward Plus Feedback Linearization

All practical controllers use feedback, as no model of robot and environment dynamics will be perfect. Nonetheless, a good model can be used to improve performance and simplify analysis.

Let's combine PID control with a model of the robot dynamics $\{\tilde{M}, \tilde{h}\}$ to achieve the error dynamics

$$\ddot{\theta}_e + K_d\dot{\theta}_e + K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t} = c \tag{11.33}$$

along arbitrary trajectories, not just to a setpoint. The error dynamics (11.33) and a proper choice of PID gains ensure exponential decay of the trajectory error.

Since $\ddot{\theta}_e = \ddot{\theta}_d - \ddot{\theta}$, to achieve the error dynamics (11.33) we choose the robot's commanded acceleration to be

$$\ddot{\theta} = \ddot{\theta}_d - \ddot{\theta}_e,$$

then combine this with Equation (11.33) to get

$$\ddot{\theta} = \ddot{\theta}_d + K_d\dot{\theta}_e + K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t}. \tag{11.34}$$

Substituting $\ddot{\theta}$ from Equation (11.34) into a model of the robot dynamics $\{\tilde{M}, \tilde{h}\}$, we get the **feedforward plus feedback linearizing controller**, also called the **inverse dynamics controller** or the **computed torque controller**:

$$\boxed{\tau = \tilde{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e\right) + \tilde{h}(\theta,\dot{\theta}).} \tag{11.35}$$

This controller includes a feedforward component due to the use of the planned acceleration $\ddot{\theta}_d$ and is called feedback linearizing because feedback of θ and $\dot{\theta}$ is used to generate the linear error dynamics. The $\tilde{h}(\theta,\dot{\theta})$ term cancels the dynamics that depends nonlinearly on the state, and the inertia model $\tilde{M}(\theta)$ converts the desired joint accelerations into joint torques, realizing the simple linear error dynamics (11.33).

A block diagram of the computed torque controller is shown in Figure 11.18. The gains K_p, K_i, and K_d are chosen to place the roots of the characteristic equation so as to achieve good transient response. In practice K_i is often chosen to be zero.

Figure 11.19 shows the typical behavior of computed torque control relative to feedforward and feedback only. Pseudocode is given in Figure 11.20.

11.4.2 Motion Control of a Multi-joint Robot

The methods applied above for a single-joint robot carry over directly to n-joint robots. The difference is that the dynamics (11.22) now takes the more general,

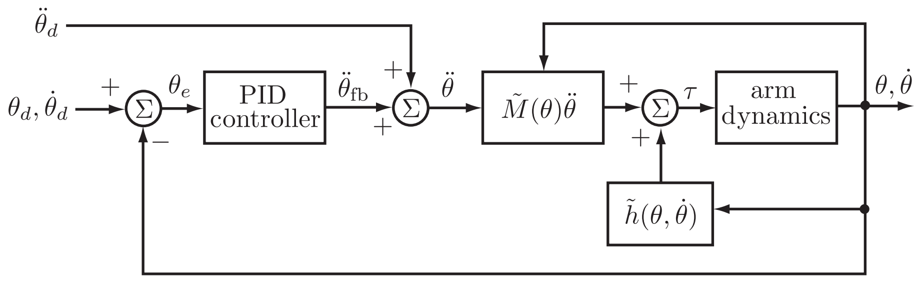

$\ddot{\theta}_d$
$\theta_d, \dot{\theta}_d$
+
−
Σ
θ_e
PID controller
$\ddot{\theta}_{\text{fb}}$
+
+
$\ddot{\theta}$
$\tilde{M}(\theta)\ddot{\theta}$
+
+
τ
arm dynamics
$\theta, \dot{\theta}$
$\tilde{h}(\theta, \dot{\theta})$

Figure 11.18: Computed torque control. The feedforward acceleration $\ddot{\theta}_d$ is added to the acceleration $\ddot{\theta}_{\text{fb}}$ computed by the PID feedback controller to create the commanded acceleration $\ddot{\theta}$.

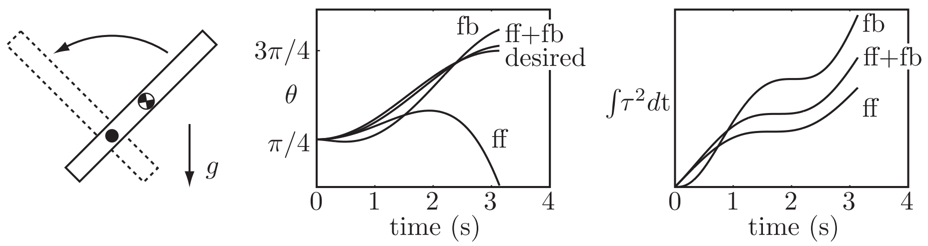


Figure 11.19: Performance of feedforward only (ff), feedback only (fb), and computed torque control (ff+fb). The PID gains are taken from Figure 11.13, and the feedforward modeling error is taken from Figure 11.17. The desired motion is Task 2 from Figure 11.17 (left-hand plot). The center plot shows the tracking performance of the three controllers. The right-hand plot shows $\int \tau^2(\mathrm{t})dt$, a standard measure of the control effort, for each of the three controllers. These plots show typical behavior: the computed torque controller yields better tracking than either feedforward or feedback alone, with less control effort than feedback alone.

vector-valued, form

$$\tau = M(\theta)\ddot{\theta} + h(\theta, \dot{\theta}), \tag{11.36}$$

where the $n \times n$ positive-definite mass matrix M is now a function of the configuration θ. In general, the components of the dynamics (11.36) are coupled – the acceleration of a joint is a function of the positions, velocities, and torques at other joints.

We distinguish between two types of control for multi-joint robots: **decentralized** control, where each joint is controlled separately with no sharing of information between joints, and **centralized** control, where full state information for each of the n joints is available to calculate the controls for each joint.

```
time = 0                                   // dt = cycle time
eint = 0                                   // error integral
qprev = senseAngle                         // initial joint angle q
loop
  [qd,qdotd,qdotdotd] = trajectory(time) // from trajectory generator

  q = senseAngle                           // sense actual joint angle
  qdot = (q - qprev)/dt                    // simple velocity calculation
  qprev = q

  e = qd - q
  edot = qdotd - qdot
  eint = eint + e*dt

  tau = Mtilde(q)*(qdotdotd+Kp*e+Kd*edot+Ki*eint) + htilde(q,qdot)
  commandTorque(tau)

  time = time + dt
end loop
```

Figure 11.20: Pseudocode for the computed torque controller.

11.4.2.1 Decentralized Multi-joint Control

The simplest method for controlling a multi-joint robot is to apply at each joint an independent controller, such as the single-joint controllers discussed in Section 11.4.1. Decentralized control is appropriate when the dynamics are decoupled, at least approximately. The dynamics are decoupled when the acceleration of each joint depends only on the torque, position, and velocity of that joint. This requires that the mass matrix be diagonal, as in Cartesian or **gantry** robots, where the first three axes are prismatic and orthogonal. This kind of robot is equivalent to three single-joint systems.

Approximate decoupling is also achieved in highly geared robots in the absence of gravity. The mass matrix $M(\theta)$ is nearly diagonal, as it is dominated by the apparent inertias of the motors themselves (see Section 8.9.2). Significant friction at the individual joints also contributes to the decoupling of the dynamics.

11.4.2.2 Centralized Multi-joint Control

When gravity forces and torques are significant and coupled, or when the mass matrix $M(\theta)$ is not well approximated by a diagonal matrix, decentralized control may not yield acceptable performance. In this case the computed torque controller (11.35) of Figure 11.18 can be generalized to a multi-joint robot. The configurations θ and θ_d and the error $\theta_e = \theta_d - \theta$ are now n-vectors, and the positive scalar gains become positive-definite matrices K_p, K_i, K_d:

$$\boxed{\tau = \tilde{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e\right) + \tilde{h}(\theta, \dot{\theta}).} \tag{11.37}$$

Typically, we choose the gain matrices as k_pI, k_iI, and k_dI, where k_p, k_i, and k_d are nonnegative scalars. Commonly, k_i is chosen to be zero. In the case of an exact dynamics model for $\tilde{M}$ and $\tilde{h}$, the error dynamics of each joint reduces to the linear dynamics (11.33). The block diagram and pseudocode for this control algorithm are found in Figures 11.18 and 11.20, respectively.

Implementing the control law (11.37) requires calculating potentially complex dynamics. We may not have a good model of these dynamics, or the equations may be too computationally expensive to calculate at servo rate. In this case, if the desired velocities and accelerations are small, an approximation to (11.37) can be obtained using only PID control and gravity compensation:

$$\boxed{\tau = K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e + \tilde{g}(\theta).} \tag{11.38}$$

With zero friction, perfect gravity compensation, and PD setpoint control ($K_i = 0$ and $\dot{\theta}_d = \ddot{\theta}_d = 0$), the controlled dynamics can be written as

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} = K_p\theta_e - K_d\dot{\theta}, \tag{11.39}$$

where the Coriolis and centripetal terms are written $C(\theta, \dot{\theta})\dot{\theta}$. We can now define a virtual "error energy," which is the sum of an "error potential energy" stored in the virtual spring K_p and an "error kinetic energy":

$$V(\theta_e, \dot{\theta}_e) = \frac{1}{2}\theta_e^{\mathrm{T}}K_p\theta_e + \frac{1}{2}\dot{\theta}_e^{\mathrm{T}}M(\theta)\dot{\theta}_e. \tag{11.40}$$

Since $\dot{\theta}_d = 0$, this reduces to

$$V(\theta_e, \dot{\theta}) = \frac{1}{2}\theta_e^{\mathrm{T}}K_p\theta_e + \frac{1}{2}\dot{\theta}^{\mathrm{T}}M(\theta)\dot{\theta}. \tag{11.41}$$

Taking the time derivative and substituting (11.39) into it, we get

$$\begin{aligned}\dot{V} &= -\dot{\theta}^{\mathrm{T}} K_p \theta_e + \dot{\theta}^{\mathrm{T}} M(\theta)\ddot{\theta} + \frac{1}{2}\dot{\theta}^{\mathrm{T}}\dot{M}(\theta)\dot{\theta} \\ &= -\dot{\theta}^{\mathrm{T}} K_p \theta_e + \dot{\theta}^{\mathrm{T}}\left(K_p\theta_e - K_d\dot{\theta} - C(\theta,\dot{\theta})\dot{\theta}\right) + \frac{1}{2}\dot{\theta}^{\mathrm{T}}\dot{M}(\theta)\dot{\theta}. \qquad (11.42)\end{aligned}$$

Rearranging, and using the fact that $\dot{M} - 2C$ is skew symmetric (Proposition 8.1.2), we get

$$\begin{aligned}\dot{V} &= -\dot{\theta}^{\mathrm{T}} K_p \theta_e + \dot{\theta}^{\mathrm{T}}\left(K_p\theta_e - K_d\dot{\theta}\right) + \cancelto{0}{\frac{1}{2}\dot{\theta}^{\mathrm{T}}\left(\dot{M}(\theta) - 2C(\theta,\dot{\theta})\right)\dot{\theta}} \\ &= -\dot{\theta}^{\mathrm{T}} K_d \dot{\theta} \ \leq 0. \qquad (11.43)\end{aligned}$$

This shows that the error energy is decreasing when $\dot{\theta} \neq 0$. If $\dot{\theta} = 0$ and $\theta \neq \theta_d$, the virtual spring ensures that $\ddot{\theta} \neq 0$, so $\dot{\theta}_e$ will again become nonzero and more error energy will be dissipated. Thus, by the Krasovskii–LaSalle invariance principle (Exercise 11.12), the total error energy decreases monotonically and the robot converges to rest at θ_d ($\theta_e = 0$) from any initial state.

11.4.3 Task-Space Motion Control

In Section 11.4.2 we focused on motion control in joint space. On the one hand, this is convenient because joint limits are easily expressed in this space, and the robot should be able to execute any joint-space path respecting these limits. Trajectories are naturally described by the joint variables, and there are no issues of singularities or redundancy.

On the other hand, since the robot interacts with the external environment and objects in it, it may be more convenient to express the motion as a trajectory of the end-effector in task space. Let the end-effector trajectory be specified by $(X(t), \mathcal{V}_b(t))$, where $X \in SE(3)$ and $[\mathcal{V}_b] = X^{-1}\dot{X}$, i.e., the twist $\mathcal{V}_b$ is expressed in the end-effector frame {b}. Provided that the corresponding trajectory in joint space is feasible, we now have two options for control: (1) convert to a joint-space trajectory and proceed with controls as in Section 11.4.2 or (2) express the robot dynamics and control law in the task space.

The first option is to convert the trajectory to joint space. The forward kinematics are $X = T(\theta)$ and $\mathcal{V}_b = J_b(\theta)\dot{\theta}$. Then the joint-space trajectory is obtained from the task-space trajectory using inverse kinematics (Chapter 6):

$$\begin{aligned}
\text{(inverse kinematics)} \quad \theta(t) &= T^{-1}(X(t)), && (11.44)\\
\dot{\theta}(t) &= J_b^{\dagger}(\theta(t))\mathcal{V}_b(t), && (11.45)\\
\ddot{\theta}(t) &= J_b^{\dagger}(\theta(t))\left(\dot{\mathcal{V}}_b(t) - \dot{J}_b(\theta(t))\dot{\theta}(t)\right). && (11.46)
\end{aligned}$$

A drawback of this approach is that we must calculate the inverse kinematics, $J_b^{\dagger}$, and $\dot{J}_b$, which may require significant computing power.

The second option is to express the robot's dynamics in task-space coordinates, as discussed in Section 8.6. Recall the task-space dynamics

$$\mathcal{F}_b = \Lambda(\theta)\dot{\mathcal{V}}_b + \eta(\theta, \mathcal{V}_b).$$

The joint forces and torques τ are related to the wrenches $\mathcal{F}_b$ expressed in the end-effector frame by $\tau = J_b^{\mathrm{T}}(\theta)\mathcal{F}_b$.

We can now write a control law in task space inspired by the computed torque control law in joint coordinates (11.37),

$$\tau = J_b^{\mathrm{T}}(\theta)\left(\tilde{\Lambda}(\theta)\left(\frac{d}{dt}([\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d) + K_pX_e + K_i\int X_e(\mathrm{t})d\mathrm{t} + K_d\mathcal{V}_e\right) + \tilde{\eta}(\theta,\mathcal{V}_b)\right), \tag{11.47}$$

where $\{\tilde{\Lambda}, \tilde{\eta}\}$ represents the controller's dynamics model and $\frac{d}{dt}([\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d)$ is the feedforward acceleration expressed in the actual end-effector frame at X (this term can be approximated as $\dot{\mathcal{V}}_d$ at states close to the reference state). The configuration error X_e satisfies $[X_e] = \log(X^{-1}X_d)$: X_e is the twist, expressed in the end-effector frame, which, if followed for unit time, would move the current configuration X to the desired configuration X_d. The velocity error is calculated as

$$\mathcal{V}_e = [\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d - \mathcal{V}.$$

The transform $[\mathrm{Ad}_{X^{-1}X_d}]$ expresses the reference twist $\mathcal{V}_d$, which is expressed in the frame X_d, as a twist in the end-effector frame at X, in which the actual velocity $\mathcal{V}$ is represented, so the two expressions can be differenced.

11.5 Force Control

When the task is not to create motions at the end-effector but to apply forces and torques to the environment, **force control** is needed. Pure force control is only possible if the environment provides resistance forces in every direction (e.g., if the end-effector is embedded in concrete or attached to a spring providing resistance in every motion direction). Pure force control is something of an abstraction, as robots are usually able to move freely in at least *some* direction. It is a useful abstraction, however, that leads to hybrid motion-force control as discussed in Section 11.6.

In ideal force control, the force applied by the end-effector is unaffected by disturbance motions applied to the end-effector. This is dual to the case of ideal motion control, where the motion is unaffected by disturbance forces.

Let $\mathcal{F}_{\text{tip}}$ be the wrench applied by the manipulator to the environment. The manipulator dynamics can be written as

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) + b(\dot{\theta}) + J^{\text{T}}(\theta)\mathcal{F}_{\text{tip}} = \tau, \tag{11.48}$$

where $\mathcal{F}_{\text{tip}}$ and $J(\theta)$ are defined in the same frame (the space frame or the end-effector frame). Since the robot typically moves slowly (or not at all) during a force control task, we can ignore the acceleration and velocity terms to get

$$g(\theta) + J^{\text{T}}(\theta)\mathcal{F}_{\text{tip}} = \tau. \tag{11.49}$$

In the absence of any direct measurements of the force–torque at the robot end-effector, joint-angle feedback alone can be used to implement the force-control law

$$\tau = \tilde{g}(\theta) + J^{\text{T}}(\theta)\mathcal{F}_d, \tag{11.50}$$

where $\tilde{g}(\theta)$ is a model of the gravitational torques and $\mathcal{F}_d$ is the desired wrench. This control law requires a good model for gravity compensation as well as precise control of the torques produced at the robot joints. In the case of a DC electric motor without gearing, torque control can be achieved by current control of the motor. In the case of a highly geared actuator, a large friction torque in the gearing can degrade the quality of torque control achieved using only current control. In this case, the output of the gearing can be instrumented with strain gauges to measure the joint torque directly; this information is fed back to a local controller that modulates the motor current to achieve the desired output torque.

Another solution is to equip the robot arm with a six-axis force-torque sensor between the arm and the end-effector to directly measure the end-effector wrench $\mathcal{F}_{\text{tip}}$ (Figure 11.21). Consider a PI force controller[2] with a feedforward term and

[2]Derivative control is not typically relevant for two reasons: (1) force measurements are often noisy, so their computed time derivatives are nearly meaningless; and (2) we are assuming the direct control of the joint torques and forces, and our simple rigid-body dynamics models imply direct transmission to the end-effector forces – there is no dynamics that integrates our control commands to produce the desired behavior, unlike with motion control.

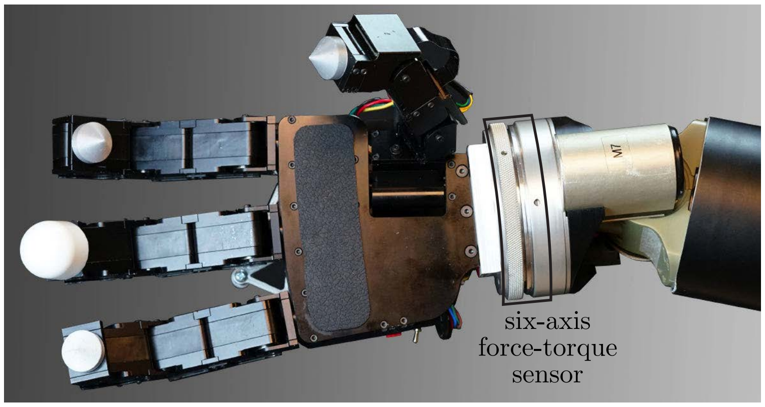


Figure 11.21: A six-axis force–torque sensor mounted between a robot arm and its end-effector.

gravity compensation,

$$\boxed{\tau = \tilde{g}(\theta) + J^{\mathrm{T}}(\theta)\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(\mathrm{t})dt\right),} \qquad (11.51)$$

where $\mathcal{F}_e = \mathcal{F}_d - \mathcal{F}_{\mathrm{tip}}$ and K_{fp} and K_{fi} are positive-definite proportional and integral gain matrices, respectively. In the case of perfect gravity modeling, plugging the force controller (11.51) into the dynamics (11.49), we get the error dynamics

$$K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(\mathrm{t})dt = 0. \qquad (11.52)$$

In the case of a nonzero but constant force disturbance on the right-hand side of (11.52), arising from an incorrect model of $\tilde{g}(\theta)$, for example, we take the derivative to get

$$K_{fp}\dot{\mathcal{F}}_e + K_{fi}\mathcal{F}_e = 0, \qquad (11.53)$$

showing that $\mathcal{F}_e$ converges to zero for positive-definite K_{fp} and K_{fi}.

The control law (11.51) is simple and appealing but potentially dangerous if incorrectly applied. If there is nothing for the robot to push against, it will accelerate in a failing attempt to create end-effector forces. Since a typical force-control task requires little motion, we can limit this acceleration by adding velocity damping. This gives the modified control law

$$\boxed{\tau = \tilde{g}(\theta) + J^{\mathrm{T}}(\theta)\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(\mathrm{t})d\mathrm{t} - K_{\mathrm{damp}}\mathcal{V}\right),} \tag{11.54}$$

where K_{damp} is positive definite.

11.6 Hybrid Motion–Force Control

Most tasks requiring the application of controlled forces also require the generation of controlled motions. **Hybrid motion-force control** is used to achieve this. If the the task space is n-dimensional then we are free to specify n of the $2n$ forces and motions at any time t; the other n are determined by the environment. Apart from this constraint, we also should not specify forces and motions in the "same direction," as then they are not independent.

For example, consider a two-dimensional environment modeled by a damper, $f = B_{\mathrm{env}}v$, where

$$B_{\mathrm{env}} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}.$$

Defining the components of v and f as (v_1, v_2) and (f_1, f_2), we have $f_1 = 2v_1+v_2$ and $f_2 = v_1 + v_2$. We have $n = 2$ freedoms to choose among the $2n = 4$ velocities and forces at any time. As an example, we can specify both f_1 and v_1 independently, because B_{env} is not diagonal. Then v_2 and f_2 are determined by B_{env}. We cannot independently control both f_1 and $2v_1 + v_2$, as these are in the "same direction" according to the environment.

11.6.1 Natural and Artificial Constraints

A particularly interesting case occurs when the environment is infinitely stiff (rigid constraints) in k directions and unconstrained in $n - k$ directions. In this case, we cannot choose *which* of the $2n$ motions and forces to specify – the contact with the environment chooses the k directions in which the robot can freely apply forces and the $n-k$ directions of free motion. For example, consider a task space with the $n = 6$ dimensions of $SE(3)$. Then a robot firmly grasping a cabinet door has $6 - k = 1$ motion freedom of its end-effector, i.e., rotation about the cabinet hinges, and therefore $k = 5$ force freedoms; the robot can apply any wrench that has zero moment about the axis of the hinges without moving the door.

As another example, a robot writing on a chalkboard may freely control the force into the board ($k = 1$), but it cannot penetrate the board; it may freely move with $6 - k = 5$ degrees of freedom (two specifying the motion of the tip of the chalk in the plane of the board and three describing the orientation of the chalk), but it cannot independently control the forces in these directions.

The chalk example comes with two caveats. The first is due to friction – the chalk-wielding robot can actually control forces tangent to the plane of the board provided that the requested motion in the plane of the board is zero and the requested tangential forces do not exceed the static friction limit determined by the friction coefficient and the normal force into the board (see the discussion of friction modeling in Section 12.2). Within this regime, the robot has three motion freedoms (rotations about three axes intersecting at the contact between the chalk and the board) and three linear force freedoms. Second, the robot could decide to pull away from the board. In this regime, the robot would have six motion freedoms and no force freedoms. Thus the configuration of the robot is not the only factor determining the directions of the motion and force freedoms. Nonetheless, in this section we consider the simplified case where the motion and force freedoms are determined solely by the robot's configuration, and all constraints are equality constraints. For example, the inequality velocity constraint of the board (the chalk cannot penetrate the board) is treated as an equality constraint (the robot also does not pull the chalk away from the board).

As a final example, consider a robot erasing a frictionless chalkboard using an eraser modeled as a rigid block (Figure 11.22). Let $X(t) \in SE(3)$ be the configuration of the block's frame {b} relative to a space frame {s}. The body-frame twist and wrench are written $\mathcal{V}_b = (\omega_x, \omega_y, \omega_z, v_x, v_y, v_z)$ and $\mathcal{F}_b = (m_x, m_y, m_z, f_x, f_y, f_z)$, respectively. Maintaining contact with the board puts $k = 3$ constraints on the twist:

$$\begin{aligned}\omega_x &= 0,\\ \omega_y &= 0,\\ v_z &= 0.\end{aligned}$$

In the language of Chapter 2, these velocity constraints are holonomic – the differential constraints can be integrated to give configuration constraints.

These constraints are called **natural constraints**, specified by the environment. There are $6 - k = 3$ natural constraints on the wrench, too: $m_z = f_x = f_y = 0$. In light of the natural constraints, we can freely specify any twist of the eraser satisfying the $k = 3$ velocity constraints and any wrench satisfying the $6 - k = 3$ wrench constraints (provided that $f_z < 0$, to maintain contact with the board). These motion and force specifications are

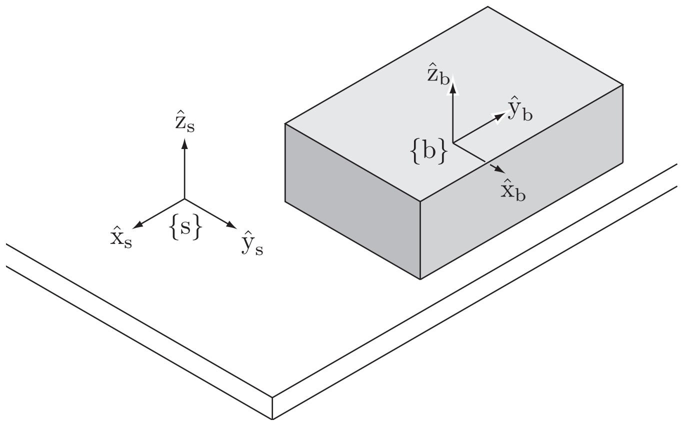


Figure 11.22: The fixed space frame {s} is attached to the chalkboard and the body frame {b} is attached to the center of the eraser.

called **artificial constraints**. Below is an example set of artificial constraints with the corresponding natural constraints:

natural constraint	artificial constraint
$\omega_x = 0$	$m_x = 0$
$\omega_y = 0$	$m_y = 0$
$m_z = 0$	$\omega_z = 0$
$f_x = 0$	$v_x = k_1$
$f_y = 0$	$v_y = 0$
$v_z = 0$	$f_z = k_2 < 0$

The artificial constraints cause the eraser to move with $v_x = k_1$ while applying a constant force k_2 against the board.

11.6.2 A Hybrid Motion–Force Controller

We now return to the problem of designing a hybrid motion–force controller. If the environment is rigid, then we can express the k natural constraints on the velocity in task space as the Pfaffian constraints

$$A(\theta)\mathcal{V} = 0, \tag{11.55}$$

where $A(\theta) \in \mathbb{R}^{k\times 6}$ for twists $\mathcal{V} \in \mathbb{R}^6$. This formulation includes holonomic and nonholonomic constraints.

If the task-space dynamics of the robot (Section 8.6), in the absence of constraints, is given by

$$\mathcal{F} = \Lambda(\theta)\dot{\mathcal{V}} + \eta(\theta, \mathcal{V}),$$

where $\tau = J^{\mathrm{T}}(\theta)\mathcal{F}$ are the joint torques and forces created by the actuators, then the constrained dynamics, following Section 8.7, is

$$\mathcal{F} = \Lambda(\theta)\dot{\mathcal{V}} + \eta(\theta, \mathcal{V}) + \underbrace{A^{\mathrm{T}}(\theta)\lambda}_{\mathcal{F}_{\mathrm{tip}}}, \tag{11.56}$$

where $\lambda \in \mathbb{R}^k$ are Lagrange multipliers and $\mathcal{F}_{\mathrm{tip}}$ is the wrench that the robot applies against the constraints. The requested wrench $\mathcal{F}_d$ must lie in the column space of $A^{\mathrm{T}}(\theta)$.

Since Equation (11.55) must be satisfied at all times, we can replace it by the time derivative

$$A(\theta)\dot{\mathcal{V}} + \dot{A}(\theta)\mathcal{V} = 0. \tag{11.57}$$

To ensure that Equation (11.57) is satisfied when the system state already satisfies $A(\theta)\mathcal{V} = 0$, any requested acceleration $\dot{\mathcal{V}}_d$ should satisfy $A(\theta)\dot{\mathcal{V}}_d = 0$.

Now solving Equation (11.56) for $\dot{\mathcal{V}}$, substituting the result into (11.57), and solving for λ, we get

$$\lambda = (A\Lambda^{-1}A^{\mathrm{T}})^{-1}(A\Lambda^{-1}(\mathcal{F} - \eta) - A\dot{\mathcal{V}}), \tag{11.58}$$

where we have used $-A\dot{\mathcal{V}} = \dot{A}\mathcal{V}$ from Equation (11.57). Using Equation (11.58), we can calculate the wrench $\mathcal{F}_{\mathrm{tip}} = A^{\mathrm{T}}(\theta)\lambda$ that the robot applies against the constraints.

Substituting Equation (11.58) into Equation (11.56) and manipulating, the n equations of the constrained dynamics (11.56) can be expressed as the $n - k$ independent motion equations

$$P(\theta)\mathcal{F} = P(\theta)(\Lambda(\theta)\dot{\mathcal{V}} + \eta(\theta, \mathcal{V})), \tag{11.59}$$

where

$$P = I - A^{\mathrm{T}}(A\Lambda^{-1}A^{\mathrm{T}})^{-1}A\Lambda^{-1} \tag{11.60}$$

and I is the identity matrix. The $n \times n$ matrix $P(\theta)$ has rank $n - k$ and projects an arbitrary manipulator wrench $\mathcal{F}$ onto the subspace of wrenches that move the end-effector tangent to the constraints. The rank-k matrix $I - P(\theta)$ projects an arbitrary wrench $\mathcal{F}$ onto the subspace of wrenches that act against the constraints. Thus P partitions the n-dimensional force space into wrenches that address the motion control task and wrenches that address the force control task.

Our hybrid motion–force controller is simply the sum of a task-space motion controller, derived from the computed torque control law (11.47), and a task-space force controller (11.51), each projected to generate forces in its appropriate subspace. Assuming wrenches and twists expressed in the end-effector frame {b},

$$\tau = J_b^{\mathrm{T}}(\theta)\Bigg(\underbrace{P(\theta)\left(\tilde{\Lambda}(\theta)\left(\frac{d}{dt}([\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d) + K_p X_e + K_i \int X_e(\mathrm{t})d\mathrm{t} + K_d \mathcal{V}_e\right)\right)}_{\text{motion control}}$$
$$+ \underbrace{(I - P(\theta))\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(\mathrm{t})d\mathrm{t}\right)}_{\text{force control}}$$
$$+ \underbrace{\tilde{\eta}(\theta, \mathcal{V}_b)}_{\text{Coriolis and gravity}}\Bigg). \qquad (11.61)$$

Because the dynamics of the two controllers are decoupled by the orthogonal projections P and $I - P$, the controller inherits the error dynamics and stability analyses of the individual force and motion controllers on their respective subspaces.

A difficulty in implementing the hybrid control law (11.61) in rigid environments is knowing the form of the constraints $A(\theta)\mathcal{V} = 0$ active at any time. This is necessary to specify the desired motion and force and to calculate the projections, but any model of the environment will have some uncertainty. One approach to dealing with this issue is to use a real-time estimation algorithm to identify the constraint directions on the basis of force feedback. Another is to sacrifice some performance by choosing low feedback gains, which makes the motion controller "soft" and the force controller more tolerant of force error. We can also build passive compliance into the structure of the robot itself to achieve a similar effect. In any case, some passive compliance is unavoidable, owing to flexibility in the joints and links.

11.7 Impedance Control

Ideal hybrid motion–force control in rigid environments demands extremes in robot **impedance**, which characterizes the change in endpoint motion as a function of disturbance forces. Ideal motion control corresponds to high impedance

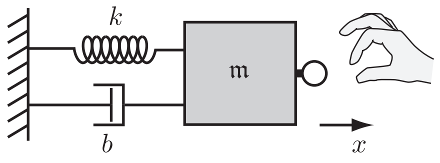


Figure 11.23: A robot creating a one-dof mass–spring–damper virtual environment. A human hand applies a force f to the haptic interface.

(little change in motion due to force disturbances) while ideal force control corresponds to low impedance (little change in force due to motion disturbances). In practice, there are limits to a robot's achievable impedance range.

In this section we consider the problem of impedance control, where the robot end-effector is asked to render particular mass, spring, and damper properties.[3] For example, a robot used as a haptic surgical simulator could be tasked with mimicking the mass, stiffness, and damping properties of a virtual surgical instrument in contact with virtual tissue.

The dynamics for a one-dof robot rendering an impedance can be written

$$\mathfrak{m}\ddot{x} + b\dot{x} + kx = f, \tag{11.62}$$

where x is the position, $\mathfrak{m}$ is the mass, b is the damping, k is the stiffness, and f is the force applied by the user (Figure 11.23). Loosely, we say that the robot renders high impedance if one or more of the $\{\mathfrak{m}, b, k\}$ parameters, usually including b or k, is large. Similarly, we say that the impedance is low if all these parameters are small.

More formally, taking the Laplace transform[4] of Equation (11.62), we get

$$(\mathfrak{m}s^2 + bs + k)X(s) = F(s), \tag{11.63}$$

and the impedance is defined by the transfer function from position perturbations to forces, $Z(s) = F(s)/X(s)$. Thus impedance is frequency dependent, with a low-frequency response dominated by the spring and a high-frequency response dominated by the mass. The **admittance**, $Y(s)$, is the inverse of the impedance: $Y(s) = Z^{-1}(s) = X(s)/F(s)$.

A good motion controller is characterized by high impedance (low admittance), since $\Delta X = Y\Delta F$. If the admittance Y is small then force perturbations

[3] A popular subcategory of impedance control is **stiffness control** or **compliance control**, where the robot renders a virtual spring only.

[4] If you are unfamiliar with the Laplace transform and transfer functions, do not panic! We do not need the details here.

ΔF produce only small position perturbations ΔX. Similarly, a good force controller is characterized by low impedance (high admittance), since $\Delta F = Z\Delta X$ and a small Z implies that motion perturbations produce only small force perturbations.

The goal of impedance control is to implement the task-space behavior

$$M\ddot{x} + B\dot{x} + Kx = f_{\text{ext}}, \tag{11.64}$$

where $x \in \mathbb{R}^n$ is the task-space configuration in a minimum set of coordinates, e.g., $x \in \mathbb{R}^3$; M, B, and K are the positive-definite virtual mass, damping, and stiffness matrices to be simulated by the robot, and f_{ext} is a force applied to the robot, perhaps by a user. The values of M, B, and K may change, depending on the location in the virtual environment, in order to represent distinct objects for instance, but we focus on the case of constant values. We could also replace $\ddot{x}$, $\dot{x}$, and x with small displacements $\Delta\ddot{x}$, $\Delta\dot{x}$, and Δx from reference values in a controlled motion of the robot, but we will dispense with any such extra notation here.

The behavior (11.64) could be implemented in terms of twists and wrenches instead, replacing f_{ext} by the (body or spatial) wrench $\mathcal{F}_{\text{ext}}$, $\dot{x}$ by the twist $\mathcal{V}$, $\ddot{x}$ by $\dot{\mathcal{V}}$, and x by the exponential coordinates $\mathcal{S}\theta$. Alternatively, the linear and rotational behaviors can be decoupled, as discussed in Section 11.4.3.

There are two common ways to achieve the behavior (11.64).

- The robot senses the endpoint motion $x(t)$ and commands joint torques and forces to create $-f_{\text{ext}}$, the force to display to the user. Such a robot is called **impedance controlled**, as it implements a transfer function $Z(s)$ from motions to forces. Theoretically, an impedance-controlled robot should only be coupled to an admittance-type environment.

- The robot senses f_{ext} using a wrist force–torque sensor and controls its motions in response. Such a robot is called **admittance controlled**, as it implements a transfer function $Y(s)$ from forces to motions. Theoretically, an admittance-controlled robot should only be coupled to an impedance-type environment.

11.7.1 Impedance-Control Algorithm

In an impedance-control algorithm, encoders, tachometers, and possibly accelerometers are used to estimate the joint and endpoint positions, velocities, and possibly accelerations. Often impedance-controlled robots are not equipped with a wrist force–torque sensor and instead rely on their ability to precisely control joint torques to render the appropriate end-effector force $-f_{\text{ext}}$ (Equation (11.64)). A good control law might be

$$\tau = J^{\mathrm{T}}(\theta)\left(\underbrace{\tilde{\Lambda}(\theta)\ddot{x} + \tilde{\eta}(\theta, \dot{x})}_{\text{arm dynamics compensation}} - \underbrace{(M\ddot{x} + B\dot{x} + Kx)}_{f_{\text{ext}}}\right), \tag{11.65}$$

where the task-space dynamics model $\{\tilde{\Lambda}, \tilde{\eta}\}$ is expressed in terms of the coordinates x. Addition of an end-effector force–torque sensor allows the use of feedback terms to achieve more closely the desired interaction force $-f_{\text{ext}}$.

In the control law (11.65), it is assumed that $\ddot{x}$, $\dot{x}$, and x are measured directly. Measurement of the acceleration $\ddot{x}$ is likely to be noisy, and there is the problem of attempting to compensate for the robot's mass after the acceleration has been sensed. Therefore, it is not uncommon to eliminate the mass compensation term $\tilde{\Lambda}(\theta)\ddot{x}$ and to set $M = 0$. The mass of the arm will be apparent to the user, but impedance-controlled manipulators are often designed to be lightweight. It is also not uncommon to assume small velocities and replace the nonlinear dynamics compensation with a simpler gravity-compensation model.

Problems can arise when (11.65) is used to simulate stiff environments (the case of large K). On the one hand, small changes in position, measured by encoders for example, lead to large changes in motor torques. This effective high gain, coupled with delays, sensor quantization, and sensor errors, can lead to oscillatory behavior or instability. On the other hand, the effective gains are low when emulating low-impedance environments. A lightweight backdrivable manipulator can excel at emulating such environments.

11.7.2 Admittance-Control Algorithm

In an admittance-control algorithm the force f_{ext} applied by the user is sensed by the wrist load cell, and the robot responds with an end-effector acceleration satisfying Equation (11.64). A simple approach is to calculate the desired end-effector acceleration $\ddot{x}_d$ according to

$$M\ddot{x}_d + B\dot{x} + Kx = f_{\text{ext}},$$

where $(x, \dot{x})$ is the current state. Solving, we get

$$\ddot{x}_d = M^{-1}(f_{\text{ext}} - B\dot{x} - Kx). \tag{11.66}$$

For the Jacobian $J(\theta)$ defined by $\dot{x} = J(\theta)\dot{\theta}$, the desired joint accelerations $\ddot{\theta}_d$ can be solved as

$$\ddot{\theta}_d = J^{\dagger}(\theta)(\ddot{x}_d - \dot{J}(\theta)\dot{\theta}),$$

and inverse dynamics used to calculate the commanded joint forces and torques τ. Simplified versions of this control law can be obtained when the goal is to simulate only a spring or a damper. To make the response smoother in the face of noisy force measurements, the force readings can be low-pass filtered.

Simulating a low-mass environment is challenging for admittance-controlled robots, as small forces produce large accelerations. The effective large gains can produce instability. Admittance control by a highly geared robot can excel at emulating stiff environments, however.

11.8 Low-Level Joint Force/Torque Control

Throughout this chapter we have been assuming that each joint produces the torque or force requested of it. In practice this ideal is not exactly achieved, and there are different approaches to approximating it. Some of the most common approaches using electric motors (Section 8.9.1) are listed below, along with their advantages and disadvantages relative to the previously listed approach. Here we assume a revolute joint and a rotary motor.

Current Control of a Direct-Drive Motor In this configuration, each joint has a motor amplifier and an electric motor with no gearhead. The torque of the motor approximately obeys the relationship $\tau = k_t I$, i.e., the torque is proportional to the current through the motor. The amplifier takes the requested torque, divides by the torque constant k_t, and generates the motor current I. To create the desired current, a current sensor integrated with the amplifier continuously measures the actual current through the motor, and the amplifier uses a local feedback control loop to adjust the time-averaged voltage across the motor to achieve the desired current. This local feedback loop runs at a higher rate than the control loop that generates the requested torques. A typical example is 10 kHz for the local current control loop and 1 kHz for the outer control loop requesting joint torques.

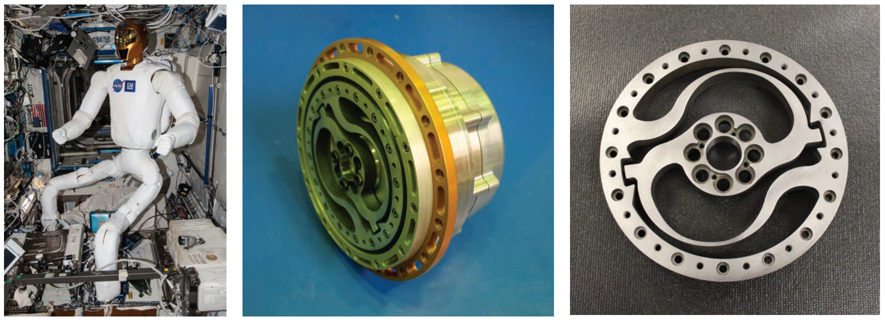
An issue with this configuration is that often an ungeared motor must be quite large to create sufficient torque for the application. The configuration can work if the motors are fixed to the ground and connected to the end-effector through cables or a closed-chain linkage. If the motors are moving, as do the motors at the joints of a serial chain, for example, large ungeared motors are often impractical.

Current Control of a Geared Motor This configuration is similar to the previous one, except that the motor has a gearhead (Section 8.9.1). A gear ratio

$G > 1$ increases the torque available to the joint.
Advantage: A smaller motor can provide the necessary torques. The motor also operates at higher speeds, where it is more efficient at converting electrical power to mechanical power.
Disadvantage: The gearhead introduces backlash (the output of the gearhead can move without the input moving, making motion control near zero velocity challenging) and friction. Backlash can be nearly eliminated by using particular types of gearing, such as harmonic drive gears. Friction, however, cannot be eliminated. The nominal torque at the gearhead output is Gk_tI, but friction in the gearhead reduces the torque available and creates significant uncertainty in the torque actually produced.

Current Control of a Geared Motor with Local Strain Gauge Feedback This configuration is similar to the previous one, except that the harmonic drive gearing is instrumented with strain gauges that sense how much torque is actually being delivered at the output of the gearhead. This torque information is used by the amplifier in a local feedback controller to adjust the current in the motor so as to achieve the requested torque.
Advantage: Putting the sensor at the output of the gearing allows compensation of frictional uncertainties.
Disadvantage: There is additional complexity of the joint configuration. Also, harmonic drive gearing achieves near-zero backlash by introducing some torsional compliance in the gearset, and the added dynamics due to the presence of this torsional spring can complicate high-speed motion control.

Series Elastic Actuator A series elastic actuator (SEA) consists of an electric motor with a gearhead (often a harmonic drive gearhead) and a torsional spring attaching the output of the gearhead to the output of the actuator. It is similar to the previous configuration, except that the torsional spring constant of the added spring is much lower than the spring constant of the harmonic drive gearing. The angular deflection $\Delta\phi$ of the spring is often measured by optical, magnetic, or capacitive encoders. The torque delivered to the output of the actuator is $k\Delta\phi$, where k is the torsional spring constant. The spring's deflection is fed to a local feedback controller that controls the current to the motor so as to achieve the desired spring deflection, and therefore the desired torque.
Advantage: The addition of the torsional spring makes the joint naturally "soft," and therefore well suited for human-robot interaction tasks. It also protects the gearing and motor from shocks at the output, such as when the output link hits something hard in the environment.



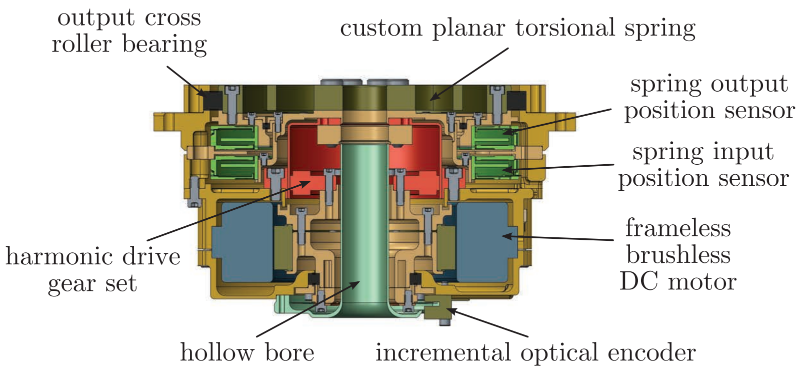


Figure 11.24: (Top left) The Robonaut 2 on the International Space Station. (Top middle) R2's hip joint SEA. (Top right) The custom torsional spring. The outer ring of hole mounts connects to the harmonic gearhead output, and the inner ring of hole mounts is the output of the SEA, connecting to the next link. The spring is designed with hard stops after approximately 0.07 rad of deflection. (Bottom) A cross-section of the SEA. The deflection $\Delta\phi$ of the torsional spring is determined by differencing the deflection readings at the spring input and the spring output. The optical encoder and spring deflection sensors provide an estimate of the joint angle. The motor controller–amplifier is located at the SEA, and it communicates with the centralized controller using a serial communication protocol. The hollow bore allows cables to pass through the interior of the SEA. All images courtesy of NASA.

Disadvantage: There is additional complexity of the joint configuration. Also, the added dynamics due to the softer spring make it more challenging to control high-speed or high-frequency motions at the output.

In 2011, NASA's Robonaut 2 (R2) became the first humanoid robot in space, performing operations on the International Space Station. Robonaut 2 incorporates a number of SEAs, including the hip actuator shown in Figure 11.24.

11.9 Other Topics

Robust Control While all stable feedback controllers confer some amount of robustness of operation to uncertainty, the field of robust control deals with designing controllers that explicitly guarantee the performance of a robot subject to bounded parametric uncertainties such as those in its inertial properties.

Adaptive Control The adaptive control of robots involves estimating the robot's inertial or other parameters during execution and updating the control law in real time to incorporate those estimates.

Iterative Learning Control Iterative learning control (ILC) generally focuses on repetitive tasks. If a robot performs the same pick-and-place operation over and over again, the trajectory errors from the previous execution can be used to modify the feedforward control for the next execution. In this way, the robot improves its performance over time, driving the execution error toward zero. This type of learning control differs from adaptive control in that the "learned" information is generally nonparametric and useful only for a single trajectory. However, ILC can account for effects that have not been parametrized in a particular model.

Passive Compliance and Flexible Manipulators All robots unavoidably have some passive compliance. Modeling this compliance can be as simple as assuming torsional springs at each revolute joint (e.g., to account for finite stiffness in the flexsplines of harmonic drive gearing) or as complicated as treating links as flexible beams. Two significant effects of flexibility are (1) a mismatch between the motor angle reading, the true joint angle, and the endpoint location of the attached link, and (2) increased order of the dynamics of the robot. These issues raise challenging problems in control, particularly when the vibration modes are at low frequencies.

Some robots are specifically designed for passive compliance, particularly those meant for contact interactions with humans or the environment. Such robots may sacrifice motion-control performance in favor of safety. One passively compliant actuator is the series elastic actuator, described above.

Variable-Impedance Actuators The impedance of a joint is typically controlled using a feedback control law, as described in Section 11.7. There are limits to the bandwidth of this control, however; a joint that is actively controlled to behave as a spring will only achieve spring-like behavior in respect of low-frequency perturbations.

A new class of actuators, called **variable-impedance** or **variable-stiffness actuators**, is intended to give actuators desired passive mechanical impedance without the bandwidth limitations of an active control law. As an example, a variable-stiffness actuator may consist of two motors, with one motor independently controlling the mechanical stiffness of the joint (e.g., using the setpoint of an internal nonlinear spring) while the other motor produces a torque.

11.10 Summary

- The performance of a feedback controller is often tested by specifying a nonzero initial error $\theta_e(0)$. The error response is typically characterized by the overshoot, the 2% settling time, and the steady-state error.

- The linear error dynamics
$$a_p\theta_e^{(p)} + a_{p-1}\theta_e^{(p-1)} + \cdots + a_2\ddot{\theta}_e + a_1\dot{\theta}_e + a_0\theta_e = 0$$
is stable, and all initial errors converge to zero, if and only if all the complex roots $s_1, \ldots, s_p$ of the characteristic equation
$$a_p s^p + a_{p-1}s^{p-1} + \cdots + a_2 s^2 + a_1 s + a_0 = 0$$
have real components less than zero, i.e., $\mathrm{Re}(s_i) < 0$ for all $i = 1, \ldots, p$.

- Stable second-order linear error dynamics can be written in the standard form
$$\ddot{\theta}_e + 2\zeta\omega_n\dot{\theta}_e + \omega_n^2\theta_e = 0,$$
where ζ is the damping ratio and ω_n is the natural frequency. The roots of the characteristic equation are
$$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}.$$
The error dynamics are overdamped if $\zeta > 1$, critically damped if $\zeta = 1$, and underdamped if $\zeta < 1$.

- The feedforward plus PI feedback controller generating joint velocity commands for a multi-joint robot is
$$\dot{\theta}(t) = \dot{\theta}_d(t) + K_p\theta_e(t) + K_i\int_0^t \theta_e(\mathrm{t})\,d\mathrm{t},$$
where $K_p = k_pI$ and $K_i = k_iI$. The joint error $\theta_e(t)$ converges to zero as t goes to infinity, for setpoint control or constant reference velocities, provided that $k_p > 0$ and $k_i > 0$.

- A task-space version of the feedforward plus PI feedback controller generating twists expressed in the end-effector frame is written

$$\mathcal{V}_b(t) = [\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d(t) + K_p X_e(t) + K_i \int_0^t X_e(\mathrm{t})\, d\mathrm{t},$$

  where $[X_e] = \log(X^{-1}X_d)$.

- The PID joint-space feedback controller generating joint forces and torques is

$$\tau = K_p\theta_e + K_i \int \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e,$$

  where $\theta_e = \theta_d - \theta$ and θ_d is the vector of the desired joint angles.

- The joint-space computed torque controller is

$$\tau = \tilde{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i \int \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e\right) + \tilde{h}(\theta, \dot{\theta}).$$

  This controller cancels nonlinear terms, uses feedforward control to proactively generate the desired acceleration $\ddot{\theta}_d$, and uses linear feedback control for stabilization.

- For robots without joint friction and a perfect model of gravity forces, joint-space PD setpoint control plus gravity compensation,

$$\tau = K_p\theta_e + K_d\dot{\theta} + \tilde{g}(\theta),$$

  yields global convergence to $\theta_e = 0$ by the Krasovskii–LaSalle invariance principle.

- Task-space force control can be achieved by the controller

$$\tau = \tilde{g}(\theta) + J^{\mathrm{T}}(\theta)\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(\mathrm{t})d\mathrm{t} - K_{\mathrm{damp}}\mathcal{V}\right),$$

  consisting of gravity compensation, feedforward force control, PI force feedback, and damping to prevent fast motion.

- Rigid constraints in the environment specify $6-k$ free motion directions and k constraint directions in which forces can be applied. These constraints can be represented as $A(\theta)\mathcal{V} = 0$. A wrench $\mathcal{F}$ can be partitioned as $\mathcal{F} = P(\theta)\mathcal{F} + (I - P(\theta))\mathcal{F}$, where $P(\theta)$ projects onto wrenches that move the end-effector tangent to the constraints and $I - P(\theta)$ projects onto wrenches that act against the constraints. The projection matrix $P(\theta)$ is written in terms of the task-space mass matrix $\Lambda(\theta)$ and constraints $A(\theta)$ as

$$P = I - A^{\mathrm{T}}(A\Lambda^{-1}A^{\mathrm{T}})^{-1}A\Lambda^{-1}.$$

- An impedance controller measures end-effector motions and creates endpoint forces to mimic a mass–spring–damper system. An admittance controller measures end-effector forces and creates endpoint motions to achieve the same purpose.

11.11 Software

Software functions associated with this chapter are listed below.

```
taulist = ComputedTorque(thetalist,dthetalist,eint,g,
Mlist,Glist,Slist,thetalistd,dthetalistd,ddthetalistd,Kp,Ki,Kd)
```
This function computes the joint controls τ for the computed torque control law (11.35) at a particular time instant. The inputs are the n-vectors of joint variables, joint velocities, and joint error integrals; the gravity vector $\mathfrak{g}$; a list of transforms $M_{i-1,i}$ describing the link center of mass locations; a list of link spatial inertia matrices $\mathcal{G}_i$; a list of joint screw axes $\mathcal{S}_i$ expressed in the base frame; the n-vectors θ_d, $\dot{\theta}_d$, and $\ddot{\theta}_d$ describing the desired motion; and the scalar PID gains k_p, k_i, and k_d, where the gain matrices are just $K_p = k_pI$, $K_i = k_iI$, and $K_d = k_dI$.

```
[taumat,thetamat] = SimulateControl(thetalist,dthetalist,g,
Ftipmat,Mlist,Glist,Slist,thetamatd,dthetamatd,ddthetamatd,
gtilde,Mtildelist,Gtildelist,Kp,Ki,Kd,dt,intRes)
```
This function simulates the computed torque controller (11.35) over a given desired trajectory. The inputs include the initial state of the robot, given by $\theta(0)$ and $\dot{\theta}(0)$; the gravity vector $\mathfrak{g}$; an $N \times 6$ matrix of wrenches applied by the end-effector, where each of the N rows corresponds to an instant in time in the trajectory; a list of transforms $M_{i-1,i}$ describing the link center-of-mass locations; a list of link spatial-inertia matrices $\mathcal{G}_i$; a list of joint screw axes $\mathcal{S}_i$ expressed in the base frame; the $N \times n$ matrices of the desired joint positions, velocities, and accelerations, where each of the N rows corresponds to an instant in time; a (possibly incorrect) model of the gravity vector; a (possibly incorrect) model of the transforms $M_{i-1,i}$; a (possibly incorrect) model of the link inertia matrices; the scalar PID gains k_p, k_i, and k_d, where the gain matrices are $K_p = k_pI$, $K_i = k_iI$, and $K_d = k_dI$; the timestep between each of the N rows

in the matrices defining the desired trajectory; and the number of integration steps to take during each timestep.

11.12 Notes and References

The computed torque controller originates from research in the 1970's [137, 106, 9, 145], and issues with its practical implementation (e.g., its computational complexity and modeling errors) have driven much of the subsequent research in nonlinear control, robust control, iterative learning control, and adaptive control. PD control plus gravity compensation was suggested and analyzed in [184], and subsequent analysis and modification of the basic controller is reviewed in [71].

The task-space approach to motion control, also called operational space control, was originally outlined in [99, 74]. A geometric approach to tracking control for mechanical systems is presented in [22], where the configuration space for the system can be a generic manifold, including $SO(3)$ and $SE(3)$.

The notion of natural and artificial constraints in hybrid motion–force control was first described by Mason [107], and an early hybrid motion–force controller based on these concepts is reported in [144]. As pointed out by Duffy [42], one must take care in specifying the subspaces in which motions and forces can be controlled. The approach to hybrid motion–force control in this chapter mirrors the geometric approach of Liu and Li [91]. Impedance control was first described in a series of papers by Hogan [56, 57, 58]. The stiffness matrix for a rigid body whose configuration is represented by $X \in SE(3)$ is discussed in [60, 93].

Robot control builds on the well-established field of linear control (e.g., [49, 4]) and the growing field of nonlinear control [63, 64, 72, 126, 158]. General references on robot control include the edited volume [33]; the textbooks by Spong et al. [177], Siciliano et al. [171], Craig [32], and Murray et al. [122]; chapters in the Handbook of Robotics on Motion Control [29] and Force Control [190]; the Robot Motion Control chapter in the Encyclopedia of Systems and Control [176]; and, for underactuated and nonholonomic robots, the chapters in the Control Handbook [101] and the Encyclopedia of Systems and Control [100].

The principles governing SEAs are laid out in [141], and NASA's Robonaut 2 and its SEAs are described in [1, 36, 115]. Variable impedance actuators are reviewed in [189].

11.13 Exercises

Exercise 11.1 Classify the following robot tasks as motion control, force control, hybrid motion–force control, impedance control, or some combination. Justify your answer.
(a) Tightening a screw with a screwdriver.
(b) Pushing a box along the floor.
(c) Pouring a glass of water.
(d) Shaking hands with a human.
(e) Throwing a baseball to hit a target.
(f) Shoveling snow.
(g) Digging a hole.
(h) Giving a back massage.
(i) Vacuuming the floor.
(j) Carrying a tray of glasses.

Exercise 11.2 The 2% settling time of an underdamped second-order system is approximately $t = 4/(\zeta\omega_n)$, for $e^{-\zeta\omega_n t} = 0.02$. What is the 5% settling time?

Exercise 11.3 Solve for any constants and give the specific equation for an underdamped second-order system with $\omega_n = 4$, $\zeta = 0.2$, $\theta_e(0) = 1$, and $\dot{\theta}_e(0) = 0$. Calculate the damped natural frequency, approximate overshoot, and 2% settling time. Plot the solution on a computer and measure the exact overshoot and settling time.

Exercise 11.4 Solve for any constants and give the specific equation for an underdamped second-order system with $\omega_n = 10$, $\zeta = 0.1$, $\theta_e(0) = 0$, and $\dot{\theta}_e(0) = 1$. Calculate the damped natural frequency. Plot the solution on a computer.

Exercise 11.5 Consider a pendulum in a gravitational field with $g = 10\ \mathrm{m/s^2}$. The pendulum consists of a 2 kg mass at the end of a 1 m massless rod. The pendulum joint has a viscous-friction coefficient of $b = 0.1$ N m s/rad.
(a) Write the equation of motion of the pendulum in terms of θ, where $\theta = 0$ corresponds to the "hanging down" configuration.
(b) Linearize the equation of motion about the stable "hanging down" equilibrium. To do this, replace any trigonometric terms in θ with the linear term in the Taylor expansion. Give the effective mass and spring con-

stants m and k in the linearized dynamics $\mathfrak{m}\ddot{\theta} + b\dot{\theta} + k\theta = 0$. At the stable equilibrium, what is the damping ratio? Is the system underdamped, critically damped, or overdamped? If it is underdamped, what is the damped natural frequency? What is the time constant of convergence to the equilibrium and the 2% settling time?

(c) Now write the linearized equations of motion for $\theta = 0$ at the balanced upright configuration. What is the effective spring constant k?

(d) You add a motor at the joint of the pendulum to stabilize the upright position, and you choose a P controller $\tau = K_p\theta$. For what values of K_p is the upright position stable?

Exercise 11.6 Develop a controller for a one-dof mass–spring–damper system of the form $\mathfrak{m}\ddot{x} + b\dot{x} + kx = f$, where f is the control force and $\mathfrak{m} = 4$ kg, $b = 2$ Ns/m, and $k = 0.1$ N/m.

(a) What is the damping ratio of the uncontrolled system? Is the uncontrolled system overdamped, underdamped, or critically damped? If it is underdamped, what is the damped natural frequency? What is the time constant of convergence to the origin?

(b) Choose a P controller $f = K_p x_e$, where $x_e = x_d - x$ is the position error and $x_d = 0$. What value of K_p yields critical damping?

(c) Choose a D controller $f = K_d\dot{x}_e$, where $\dot{x}_d = 0$. What value of K_d yields critical damping?

(d) Choose a PD controller that yields critical damping and a 2% settling time of 0.01 s.

(e) For the PD controller above, if $x_d = 1$ and $\dot{x}_d = \ddot{x}_d = 0$, what is the steady-state error $x_e(t)$ as t goes to infinity? What is the steady-state control force?

(f) Now insert a PID controller for f. Assume $x_d \neq 0$ and $\dot{x}_d = \ddot{x}_d = 0$. Write the error dynamics in terms of $\ddot{x}_e$, $\dot{x}_e$, x_e, and $\int x_e(\mathrm{t})d\mathrm{t}$ on the left-hand side and a constant forcing term on the right-hand side. (Hint: You can write kx as $-k(x_d - x) + kx_d$.) Take the time derivative of this equation and give the conditions on K_p, K_i, and K_d for stability. Show that zero steady-state error is possible with a PID controller.

Exercise 11.7 Simulation of a one-dof robot and robot controller.

(a) Write a simulator for a one-joint robot consisting of a motor rotating a link in gravity using the model parameters given in Section 11.4.1. The simulator should consist of: (1) a dynamics function that takes as input the current state of the robot and the torque applied by the motor and gives as output the acceleration of the robot; and (2) a numerical integrator that uses the dynamics function to calculate the new state of the system over a series of timesteps Δt. A first-order Euler integration method suffices for this problem (e.g., $\theta(k+1) = \theta(k) + \dot{\theta}(k)\Delta t$, $\dot{\theta}(k+1) = \dot{\theta}(k) + \ddot{\theta}(k)\Delta t$). Test the simulator in two ways: (1) starting the robot at rest at $\theta = -\pi/2$ and applying a constant torque of 0.5 N m; and (2) starting the robot at rest at $\theta = -\pi/4$ and applying zero torque. For both examples, plot the position as a function of time for sufficient duration to see the basic behavior. Ensure that the behavior makes sense. A reasonable choice of Δt is 1 ms.

(b) Add two more functions to your simulator: (1) a trajectory generator function that takes the current time and returns the desired state and acceleration of the robot; and (2) a control function that takes the current state of the robot and information from the trajectory generator and returns a control torque. The simplest trajectory generator would return $\theta = \theta_{d1}$ and $\dot{\theta} = \ddot{\theta} = 0$ for all time $t < T$, and $\theta = \theta_{d2} \neq \theta_{d1}$ and $\dot{\theta} = \ddot{\theta} = 0$ for all time $t \geq T$. This trajectory is a step function in position. Use a PD feedback controller for the control function and set $K_p = 10$ N m/rad. For a well-tuned choice of K_d, give K_d (including units) and plot the position as a function of time over 2 seconds for an initial state at rest at $\theta = -\pi/2$ and a step trajectory with $\theta_{d1} = -\pi/2$ and $\theta_{d2} = 0$. The step occurs at $T = 1$ s.

(c) Demonstrate two different choices of K_d that yield (1) overshoot and (2) sluggish response with no overshoot. Give the gains and the position plots.

(d) Add a nonzero K_i to your original well-tuned PD controller to eliminate steady-state error. Give the PID gains and plot the results of the step test.

Exercise 11.8 Modify the simulation of the one-joint robot in Exercise 11.7 to model a flexible transmission from the motor to the link with a stiffness of 500 Nm/rad. Tune a PID controller to give a good response for a desired trajectory with a step transition from $\theta = -\pi/2$ to $\theta = 0$. Give the gains and plot the response.

Exercise 11.9 Simulation of a two-dof robot and robot controller (Figure 11.25).

(a) *Dynamics.* Derive the dynamics of a 2R robot under gravity (Figure 11.25). The mass of link i is $\mathfrak{m}_i$, the center of mass is a distance r_i from the joint, the scalar inertia of link i about the joint is $\mathcal{I}_i$, and the length of link i is L_i. There is no friction at the joints.

(b) *Direct drive.* Assume each joint is directly driven by a DC motor with

no gearing. Each motor comes with specifications of the mass $\mathfrak{m}_i^{\text{stator}}$ and inertia $\mathcal{I}_i^{\text{stator}}$ of the stator and the mass $\mathfrak{m}_i^{\text{rotor}}$ and inertia $\mathcal{I}_i^{\text{rotor}}$ of the rotor (the spinning portion). For the motor at joint i, the stator is attached to link $i-1$ and the rotor is attached to link i. The links are thin uniform-density rods of mass $\mathfrak{m}_i$ and length L_i.
In terms of the quantities given above, for each link $i \in \{1,2\}$ give equations for the total inertia $\mathcal{I}_i$ about the joint, the mass $\mathfrak{m}_i$, and the distance r_i from the joint to the center of mass. Think about how to assign the mass and inertia of the motors to the different links.

(c) *Geared robot.* Assume that motor i has gearing with gear ratio G_i and that the gearing itself is massless. As in part (b) above, for each link $i \in \{1,2\}$, give equations for the total inertia $\mathcal{I}_i$ about the joint, mass $\mathfrak{m}_i$, and distance r_i from the joint to the center of mass.

(d) *Simulation and control.* As in Exercise 11.7, write a simulator with (at least) four functions: a dynamics function, a numerical integrator, a trajectory generator, and a controller. Assume that there is zero friction at the joints, $g = 9.81$ m/s^2 in the direction indicated, $L_i = 1$ m, $r_i = 0.5$ m, $\mathfrak{m}_1 = 3$ kg, $\mathfrak{m}_2 = 2$ kg, $\mathcal{I}_1 = 2$ kg m^2, and $\mathcal{I}_2 = 1$ kg m^2. Program a PID controller, find gains that give a good response, and plot the joint angles as a function of time for a reference trajectory which is constant at $(\theta_1, \theta_2) = (-\pi/2, 0)$ for $t < 1$ s and constant at $(\theta_1, \theta_2) = (0, -\pi/2)$ for $t \geq 1$ s. The initial state of the robot is at rest with $(\theta_1, \theta_2) = (-\pi/2, 0)$.

(e) *Torque limits.* Real motors have limits on the available torque. While these limits are generally velocity dependent, here we assume that each motor's torque limit is independent of velocity, $\tau_i \leq |\tau_i^{\max}|$. Assume that $\tau_1^{\max} = 100$ N m and $\tau_2^{\max} = 20$ N m. The control law may request greater torque but the actual torque is saturated at these values. Rerun the PID control simulation in (d) and plot the torques as well as the position as a function of time.

(f) *Friction.* Add a viscous friction coefficient of $b_i = 1$ N m s/rad to each joint and rerun the PID control simulation in (e).

Exercise 11.10 For the two-joint robot of Exercise 11.9, write a more sophisticated trajectory generator function. The trajectory generator should take the following as input:

- the desired initial position, velocity, and acceleration of each joint;
- the desired final position, velocity, and acceleration; and
- the total time of motion T.

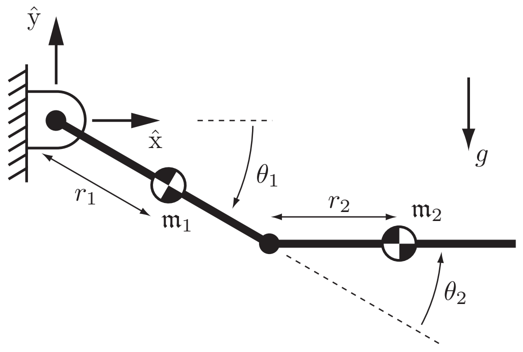


Figure 11.25: A two-link robot arm. The length of link i is L_i and its inertia about the joint is $\mathcal{I}_i$. The acceleration due to gravity is $g = 9.81$ m/s^2.

A call of the form

```
[qd,qdotd,qdotdotd] = trajectory(time)
```

returns the desired position, velocity, and acceleration of each joint at time `time`. The trajectory generator should provide a trajectory that is a smooth function of time.

As an example, each joint could follow a fifth-order polynomial trajectory of the form

$$\theta_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \tag{11.67}$$

Given the desired positions, velocities, and accelerations of the joints at times $t = 0$ and $t = T$, you can uniquely solve for the six coefficients $a_0, \ldots, a_5$ by evaluating Equation (11.67) and its first and second derivatives at $t = 0$ and $t = T$.

Tune a PID controller to track a fifth-order polynomial trajectory moving from rest at $(\theta_1, \theta_2) = (-\pi/2, 0)$ to rest at $(\theta_1, \theta_2) = (0, -\pi/2)$ in $T = 2$ s. Give the values of your gains and plot the reference positions of both joints and the actual positions of both joints. You are free to ignore torque limits and friction.

Exercise 11.11 For the two-joint robot of Exercise 11.9 and fifth-order polynomial trajectory of Exercise 11.10, simulate a computed torque controller to stabilize the trajectory. The robot has no joint friction or torque limits. The modeled link masses should be 20% greater than their actual values to create error in the feedforward model. Give the PID gains and plot the reference and actual joint angles for the computed torque controller as well as for PID control only.

Exercise 11.12 The Krasovskii–LaSalle invariance principle states the following. Consider a system $\dot{x} = f(x), x \in \mathbb{R}^n$ such that $f(0) = 0$ and any energy-like function $V(x)$ such that:

- $V(x) > 0$ for all $x \neq 0$;
- $V(x) \to \infty$ as $x \to \infty$;
- $V(0) = \dot{V}(0) = 0$; and
- $\dot{V}(x) \leq 0$ along all trajectories of the system.

Let $\mathcal{S}$ be the largest set of $\mathbb{R}^n$ such that $\dot{V}(x) = 0$ and trajectories beginning in $\mathcal{S}$ remain in $\mathcal{S}$ for all time. Then, if $\mathcal{S}$ contains only the origin, the origin is globally asymptotically stable – all trajectories converge to the origin.

Using the energy function $V(x)$ from Equation (11.40), show how the Krasovskii–LaSalle principle is violated for centralized multi-joint PD setpoint control with gravity compensation if $K_p = 0$ or $K_d = 0$. For a practical robot system, is it possible to use the Krasovskii–LaSalle invariance principle to demonstrate global asymptotic stability even if $K_d = 0$? Explain your answer.

Exercise 11.13 The two-joint robot of Exercise 11.9 can be controlled in task space using the endpoint task coordinates $X = (x, y)$, as shown in Figure 11.25. The task-space velocity is $\mathcal{V} = \dot{X}$. Give the Jacobian $J(\theta)$ and the task-space dynamics model $\{\tilde{\Lambda}(\theta), \tilde{\eta}(\theta, \mathcal{V})\}$ in the computed torque-control law (11.47).

Exercise 11.14 Choose appropriate space and end-effector reference frames {s} and {b} and express natural and artificial constraints, six each, that achieve the following tasks: (a) opening a cabinet door; (b) turning a screw that advances linearly a distance p for every revolution; and (c) drawing a circle on a chalkboard with a piece of chalk.

Exercise 11.15 Assume that the end-effector of the two-joint robot in Figure 11.25 is constrained to move on the line $x - y = 1$. The robot's link lengths are $L_1 = L_2 = 1$. Write the constraint as $A(\theta)\mathcal{V} = 0$, where $X = (x, y)$ and $\mathcal{V} = \dot{X}$.

Exercise 11.16 Derive the constrained motion equations (11.59) and (11.60). Show all the steps.

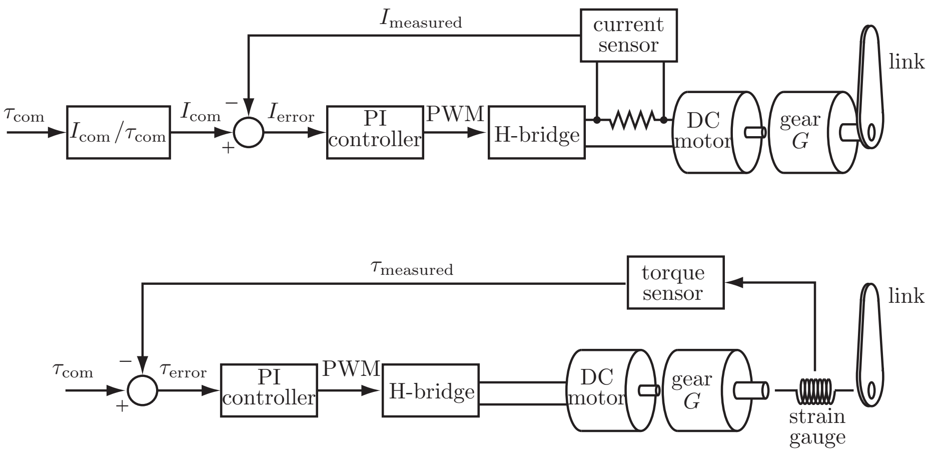


Figure 11.26: Two methods for controlling the torque at a joint driven by a geared DC motor. (Upper) The current to the motor is measured by measuring the voltage across a small resistance in the current path. A PI controller works to make the actual current match better the requested current I_{com}. (Lower) The actual torque delivered to the link is measured by strain gauges.

Exercise 11.17 We have been assuming that each actuator delivers the torque requested by the control law. In fact, there is typically an inner control loop running at each actuator, typically at a higher servo rate than the outer loop, to try to track the torque requested. Figure 11.26 shows two possibilities for a DC electric motor, where the torque τ delivered by the motor is proportional to the current I through the motor, $\tau = k_t I$. The torque from the motor is amplified by the gearhead with gear ratio G.

In the upper control scheme the motor current is measured by a current sensor and compared with the desired current I_{com}; the error is passed through a PI controller which sets the duty cycle of a low-power pulse-width-modulation (PWM) digital signal and the PWM signal is sent to an H-bridge that generates the actual motor current. In the lower scheme, a strain gauge torque sensor is inserted between the output of the motor gearing and the link, and the measured torque is compared directly with the requested torque τ_{com}. Since a strain gauge measures deflection, the element on which it is mounted must have a finite torsional stiffness. Series elastic actuators are designed to have particularly flexible torsional elements, so much so that encoders are used to measure the larger deflections. The torque is estimated from the encoder reading and the torsional spring constant.

(a) For the current sensing scheme, what multiplicative factor should go in the block labeled $I_{\text{com}}/\tau_{\text{com}}$? Even if the PI current controller does its job perfectly ($I_{\text{error}} = 0$) and the torque constant k_t is perfectly known, what effect may contribute to error in the generated torque?

(b) For the strain gauge measurement method, explain the drawbacks, if any, of having a flexible element between the gearhead and the link.

Exercise 11.18 Modify the `SimulateControl` function to allow initial state errors.