Chapter 10

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 pressing the chalk against the board) and motions in others (motion in the plane of the board). When the purpose of the robot is to act as a haptic display, mimicking 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 to achieve the behaviors described above are known as *motion (or position) control*, *force control*, *hybrid motion-force control*, and *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 mechanics, irrespective of the environment: the robot cannot independently control both motions and forces. If the robot imposes a motion, then the environment will determine the force, and vice-versa.

Once we have chosen a control goal consistent with the task and environment, we have a number of ways to achieve it. *Feedback control* uses position, velocity, and force sensors to measure the actual behavior of the robot, compare it to the desired behavior, and modulate the control signals sent to the actuators. Feedback is used in nearly all robot systems. *Feedforward control* uses a model of the dynamics of the robot and its environment to determine actuator signals that achieve the desired change in state. Because of modeling errors, feedforward control is rarely used by itself, but it is often used in conjunction with feedback control. Complementary control strategies include *adaptive control*, which continuously estimates properties of the dynamic system to improve performance; *robust control* to guarantee some level of performance in the face

of an uncertain model of the system; and *iterative learning control* for repetitive tasks, where errors from previous executions of the same task are used to generate more appropriate feedforward controls for future iterations.

In this chapter we focus on feedback and feedforward control for motion control, force control, hybrid motion-force control, and impedance control.

10.1 Control System Overview

A typical control block diagram is shown in Figure 10.1(a). The controller is often a PC or microcontroller. Sensors are typically potentiometers, encoders, or resolvers for joint position/angle sensing; tachometers for joint velocity sensing; strain gauge 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, higher control update rates are of limited benefit, given time constants associated with the dynamics of the robot and environment. In our analysis, we will ignore the nonzero sampling time and treat controllers as if they are 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 position signals at successive time steps. A low-pass filter is often used in combination with the differencing filter to reduce high frequency jitter due to quantization.

Actuators could be brushed DC motors, brushless DC motors, various types of AC motors, hydraulic actuators, or pneumatic actuators, among others. Typically gears or other transmissions are used to lower the speed and increase the force or torque of the actuator. Electric motors are coupled with a power amplifier that converts signals from the controller to high currents to drive the motor. Local feedback of motor current or joint torque may be used in an inner control loop to achieve the forces or torques requested by the controller.

For each robot joint, we will lump the 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 to the one shown in Figure 10.1(b), where the controller produces forces and torques directly. The rest of this chapter deals with the control algorithms that go inside the "Controller" box in Figure 10.1(b).

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

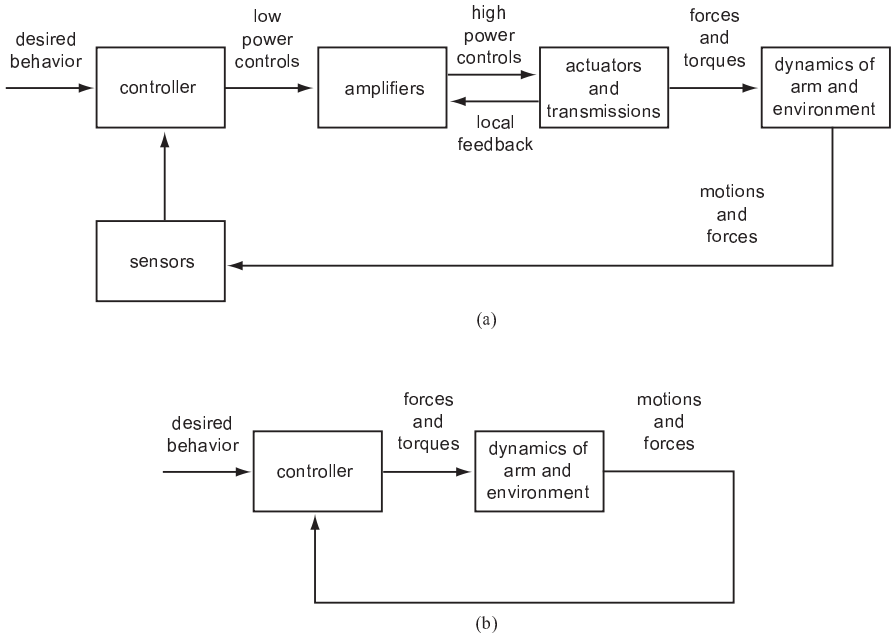

Figure 10.1: (a) A typical robot control system. An inner control loop may be used to help the amplifier and actuator achieve the desired force or torque. For example, a DC motor amplifier in torque control mode may sense the current actually flowing through the motor and implement a local controller to better match the desired current, since the current is proportional to the torque produced by the motor. (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).

10.2 Motion Control

Typically a motion control task is to have the end-effector follow a desired trajectory. If the robot is redundant, there may be more than one set of joint variable histories that yield this end-effector trajectory. We begin by assuming that inverse kinematics and its derivatives, possibly with redundancy resolution, have already been applied to yield a unique set of target joint histories as a function of time. Our goal is to construct controllers that drive the robot to track this trajectory in joint space. We consider the case of a trajectory expressed in the task space in Section 10.2.3.

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

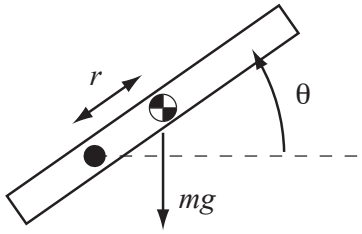

Figure 10.2: A single joint robot rotating in a gravity field.

10.2.1 Motion Control of a Single Joint

Consider a single motor attached to a single link, as shown in Figure 10.2. Let τ be the motor's torque and θ be the angle of the link. The dynamics can be written as

$$\tau = M\ddot{\theta} + mgr\cos\theta, \tag{10.1}$$

where M is the inertia of the link about the axis of rotation, 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 gravitational acceleration.

According to the model (10.1), there is no dissipation: if the link is made to spin and $\tau = 0$, it would spin forever. This is unrealistic, of course; there is friction at the various bearings, gears, and transmissions. Friction modeling is an active research area, but a simple model of rotational friction is viscous friction,

$$\tau_{\text{fric}} = b\dot{\theta}, \tag{10.2}$$

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

$$\tau = M\ddot{\theta} + mgr\cos\theta + b\dot{\theta}, \tag{10.3}$$

which we may write more compactly as

$$\tau = M\ddot{\theta} + h(\theta, \dot{\theta}), \tag{10.4}$$

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

For concreteness in the following simulations, we set $M = 0.5$ kgm^2, $m = 1$ kg, $r = 0.1$ m, and $b = 0.1$ Nms/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.

10.2.1.1 Feedback Control: PID Control

The most common feedback control algorithm is linear PID (proportional-integral-derivative) control. Defining the error between the desired angle θ_d and actual angle θ as

$$\theta_e = \theta_d - \theta, \tag{10.5}$$

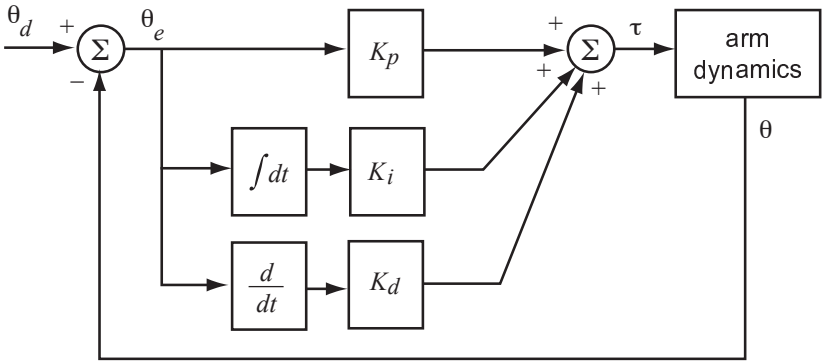

Figure 10.3: Block diagram of a PID controller.

the PID controller is simply

$$\boxed{\tau = K_p\theta_e + K_i \int \theta_e(t)dt + K_d\dot{\theta}_e,} \tag{10.6}$$

where the control gains K_p, K_i, and K_d are nonnegative. The proportional gain K_p acts as a virtual spring that tries to reduce the position error $\theta_d - \theta$, and the derivative gain K_d acts as a virtual damper that tries to reduce the velocity error $\dot{\theta}_d - \dot{\theta}$. The integral gain, as we will see later, can be used to eliminate steady-state errors when the joint is at rest. See the block diagram in Figure 10.3.

For now let's consider the case where $K_i = 0$. This is known as PD control. (We can similarly define PI, P, I, and D control by setting other gains to zero. PD and PI control are the most common variants of PID control.) Let's also assume the robot moves in a horizontal plane, so $g = 0$. Plugging the control law (10.6) into the dynamics (10.3), we get

$$M\ddot{\theta} + b\dot{\theta} = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}). \tag{10.7}$$

If the goal state is rest at a constant θ_d, then $\dot{\theta}_d = \ddot{\theta}_d = 0$. This is called *setpoint* control. Using $\theta_e = \theta_d - \theta$, $\dot{\theta}_e = -\dot{\theta}$, and $\ddot{\theta}_e = -\ddot{\theta}$, Equation (10.7) can be rewritten as the linear mass-spring-damper *error dynamics*

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e = 0. \tag{10.8}$$

Stability Error dynamics, such as Equation (10.8), are an important concept in the study of control systems. A minimum requirement is that the error dynamics be *stable*, i.e., initial errors tend to zero exponentially with time. A linear homogeneous ordinary differential equation of the form

$$a_n\theta_e^{(n)} + a_{n-1}\theta_e^{(n-1)} + \ldots + a_2\ddot{\theta}_e + a_1\dot{\theta}_e + a_0\theta_e = 0$$

is stable if and only if all of the complex roots $s_1, \ldots, s_n$ of its characteristic equation

$$a_n s^n + a_{n-1} s^{n-1} + \ldots + a_2 s^2 + a_1 s + a_0 = 0$$

have real components less than zero, i.e., $\mathrm{Re}(s_i) < 0$ for all $i = 1 \ldots n$. A necessary condition for stability, regardless of the order n of the dynamics, is that $a_i > 0$ for all i. This condition is also sufficient for second-order dynamics such as (10.8). For third-order dynamics, it is also required that $a_2 a_1 > a_3 a_0$.

PD Control and Second-Order Error Dynamics To study the second-order error dynamics (10.8) more formally, we assume stability and rewrite in the standard second-order form

$$\ddot{\theta}_e + \frac{b + K_d}{M}\dot{\theta}_e + \frac{K_p}{M}\theta_e = 0 \quad \rightarrow \quad \ddot{\theta}_e + 2\zeta\omega_n\dot{\theta}_e + \omega_n^2\theta_e = 0, \tag{10.9}$$

where the *damping ratio* ζ and the *natural frequency* ω_n are

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

The characteristic equation of (10.9) is

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0, \tag{10.10}$$

with complex roots

$$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}.$$

There are three types of solutions to the differential equation (10.9), 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 is

  $$\theta_e(t) = c_1 \exp(s_1 t) + c_2 \exp(s_2 t),$$

  where c_1 and c_2 depend on the initial conditions. The response is the sum of two decaying exponentials, with time constants $\tau_{1,2} = -1/s_{1,2}$, where the time constant is the time it takes the exponential to decay to 37% of its original value. The "slower" time constant in the solution is given by the less negative root, $s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1}$.

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

  $$\theta_e(t) = \exp(-\zeta\omega_n t)(c_1 + c_2 t),$$

  i.e., a decaying exponential multiplied by a linear function of time. The time constant of the decaying exponential is $\tau = 1/(\zeta\omega_n)$.

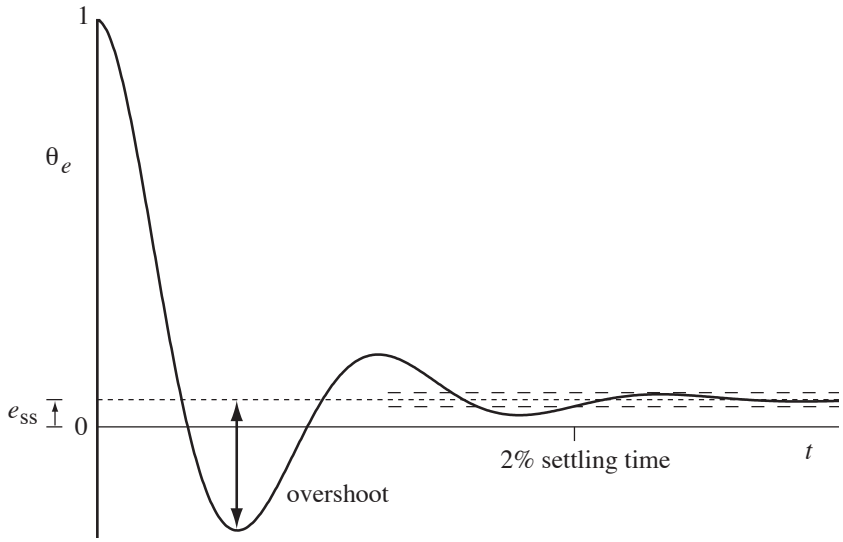

Figure 10.4: The error response to a step input for an underdamped second-order system, showing steady-state error $e_{\rm ss}$, overshoot, and 2% settling time.

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

$$\theta_e(t) = \exp(-\zeta\omega_n t)\left(c_1\cos(\omega_d t) + c_2\sin(\omega_d t)\right),$$

  i.e., a decaying exponential (time constant $\tau = 1/(\zeta\omega_n)$) multiplied by a sinusoid.

To see how to apply these solutions, imagine that the link is originally at rest at $\theta = 0$. At time $t = 0$, the desired position is suddenly changed from $\theta_d = 0$ to $\theta_d = 1$. This is called a *step input* and the resulting motion of the system $\theta(t)$ is called the *step response*. Our interest is in the error response $\theta_e(t)$. We can solve for $c_{1,2}$ in the specific solution by solving $\theta_e(0) = 1$ (the error immediately becomes 1) and $\dot{\theta}_e(0) = 0$ (both $\dot{\theta}_d(0)$ and $\dot{\theta}(0)$ are zero).

The error response can be described by a *transient response* and a *steady-state response* (Figure 10.4). The steady-state response is characterized by the *steady-state error* $e_{\rm ss}$, which is the asymptotic error $\theta_e(t)$ as $t \to \infty$. For the link in zero gravity with a stable PD controller, $e_{\rm ss} = 0$. The transient response is characterized by the *overshoot* and (2%) *settling time*. The 2% settling time is the first time T such that $|\theta_e(t) - e_{\rm ss}| \leq 0.02(1 - e_{\rm ss})$ for all $t \geq T$, and is approximately equal to 4τ, where τ is the slowest time constant in the solution. Overshoot is defined as

$$\text{overshoot} = \left|\frac{\theta_{e,\min} - e_{\rm ss}}{1 - e_{\rm ss}}\right| \times 100\%$$

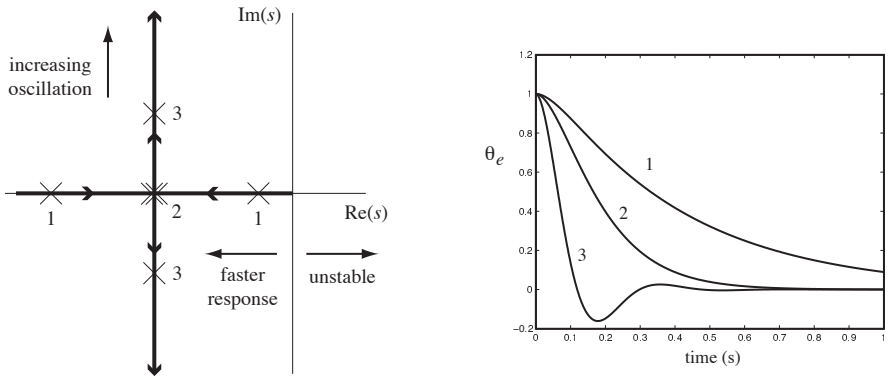

Figure 10.5: (Left) The complex roots of the characteristic equation of the PD-controlled joint for a fixed $K_d = 10$ Nms/rad as K_p increases from zero. This is known as a "root locus" plot. (Right) The response of the system to an initial error $\theta_e = 1$, $\dot{\theta}_e = 0$ is shown for overdamped ($\zeta = 1.5$, roots at "1"), critically damped ($\zeta = 1$, roots at "2"), and underdamped ($\zeta = 0.5$, roots at "3") cases.

where $\theta_{e,\min}$ is the least positive value achieved by the error. The overshoot can be calculated to be

$$\text{overshoot} = \exp(-\pi\zeta/\sqrt{1-\zeta^2}) \times 100\%, \quad 0 \leq \zeta < 1.$$

A good transient response is characterized by a low settling time and little or no overshoot.

Figure 10.5 shows the relationship of the location of the roots of (10.10) to the transient response. For a fixed K_d and a small K_p, we have $\zeta > 1$, the system is overdamped, and the response is sluggish due to the "slow" root. As K_p is increased, the damping ratio decreases. The system is critically damped ($\zeta = 1$) at $K_p = (b + K_d)^2/(4M)$, and the two roots are coincident on the negative real axis. This situation corresponds to a relatively fast response and no overshoot. As K_p continues to increase, ζ drops below 1, the roots move off the negative real axis, and we begin to see overshoot and oscillation in the response. The settling time is unaffected as K_p is increased beyond critical damping, as $\zeta\omega_n$ is unchanged.

Practical Bounds on Feedback Gains According to our simple model, we could increase K_p and K_d without bound to make the real components of the roots more and more negative, achieving arbitrarily fast response. In practice, however, large gains lead to actuator saturation, rapid torque changes (chattering), vibrations of the structure due to unmodeled flexibility in the joints and links, and even instability due to the finite servo rate frequency. Thus there are practical limits on the set of useful gains.

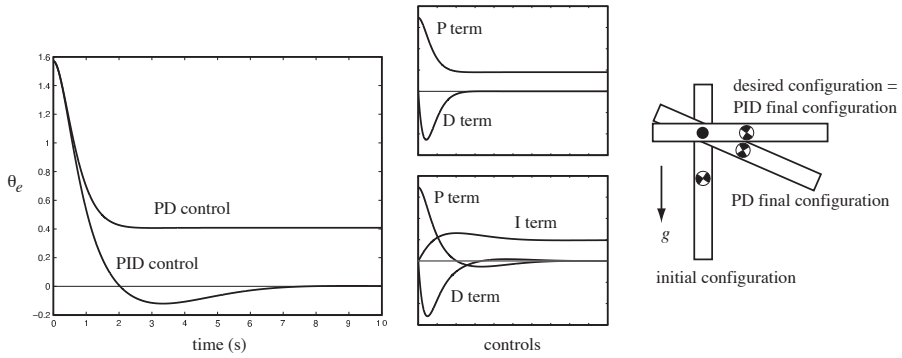

Figure 10.6: (Left) The tracking errors for a PD controller with $K_d = 2$ Nms/rad, $K_p = 2.205$ Nm/rad for critical damping, and a PID controller with the same PD gains and $K_i = 1$ Nm/(rad s). The arm starts at $\theta(0) = -\pi/2, \dot{\theta}(0) = 0$, with a goal state $\theta_d = 0, \dot{\theta}_d = 0$. (Middle) The individual contributions of the terms in the PD and PID control laws. (Right) The initial and final configurations.

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

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e = mgr\cos\theta. \tag{10.11}$$

This implies that the system comes to rest at a configuration θ satisfying $K_p\theta_e = mgr\cos\theta$, i.e., the final error θ_e is not zero when $\theta_d \neq \pm\frac{\pi}{2}$. This is because the robot must provide a nonzero torque to hold the link at rest at $\theta \neq \pm\frac{\pi}{2}$, but the PD control law only creates a nonzero torque at rest 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 10.6 demonstrates the addition of the integral term to the controller.

To see how this works, write the setpoint error dynamics

$$M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p\theta_e + K_i\int\theta_e(t)dt = \tau_{\text{dist}}, \tag{10.12}$$

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

$$M\theta_e^{(3)} + (b + K_d)\ddot{\theta}_e + K_p\dot{\theta}_e + K_i\theta_e = \dot{\tau}_{\text{dist}}. \tag{10.13}$$

If τ_{dist} is constant, then the right-hand side of (10.13) is zero. If the PID controller is stable, then (10.13) shows that θ_e converges to zero. (While the

disturbance torque due to gravity is not constant as the link rotates, it approaches a constant as $\dot{\theta}$ approaches zero, and therefore similar reasoning holds close to the equilibrium.)

Integral control is useful to eliminate steady-state error in setpoint control, but it may adversely affect the transient response. This is because integral control essentially responds to delayed information—it takes time for the system to respond to error as it integrates. It is well known in control theory that delayed feedback can cause instability. To see this, consider the characteristic equation of (10.13) when τ_{dist} is constant:

$$Ms^3 + (b + K_d)s^2 + K_p s + K_i = 0. \tag{10.14}$$

For all roots to have negative real part, we require $b + K_d > 0$ and $K_p > 0$ as before, but there is also an upper bound on the new gain K_i (Figure 10.7):

$$0 \leq K_i < \frac{(b + K_d)K_p}{M}.$$

Thus a reasonable design strategy is to choose K_p and K_d for a good transient response, then choose K_i small so as not to adversely affect stability. In the example of Figure 10.6, the relatively large K_i worsens the transient response, giving significant overshoot. In practice, $K_i = 0$ for many robot controllers.

Pseudocode for the PID control algorithm is given in Figure 10.8.

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.

10.2.1.2 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

$$\tau = \widehat{M}(\theta)\ddot{\theta} + \widehat{h}(\theta, \dot{\theta}), \tag{10.15}$$

where the model is perfect if $\widehat{M}(\theta) = M(\theta)$ and $\widehat{h}(\theta, \dot{\theta}) = h(\theta, \dot{\theta})$. Note that the inertia model $\widehat{M}(\theta)$ 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 this way allows us to re-use Equation (10.15) for multi-joint systems in Section 10.2.2.

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

$$\boxed{\tau = \widehat{M}(\theta_d)\ddot{\theta}_d + \widehat{h}(\theta_d, \dot{\theta}_d).} \tag{10.16}$$

If the model of the robot dynamics is exact, and there are no initial state errors, then the robot exactly follows the desired trajectory. This is called feedforward control, as no feedback is used.

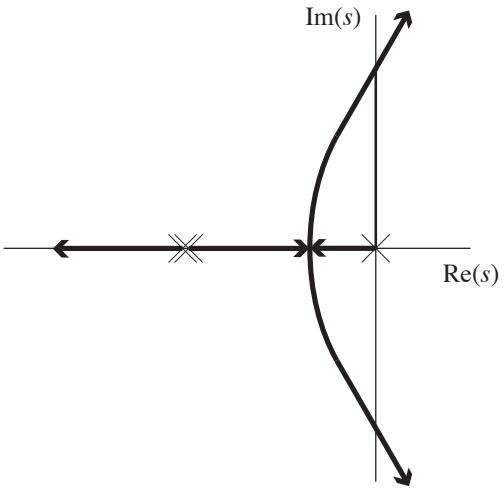

Figure 10.7: The three roots of (10.14) as K_i increases from zero. First a PD controller is chosen with K_p and K_d yielding critical damping, giving rise to two collocated roots on the negative real axis. Adding an infinitesimal gain $K_i > 0$ creates a third root at the origin. As we increase K_i, one of the two collocated roots moves to the left on the negative real axis, while the other two roots move toward each other, break away from the real axis, and 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.

A pseudocode implementation of feedforward control is given in Figure 10.9.

Figure 10.10 shows two examples of trajectory following for the link in gravity. Here, the controller's dynamic model is correct except that it has $\widehat{r} = 0.08$ 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.

10.2.1.3 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 $\{\widehat{M}, \widehat{h}\}$ to achieve the error dynamics

$$\ddot{\theta}_e + K_d\dot{\theta}_e + K_p\theta_e + K_i \int \theta_e(t)dt = 0 \tag{10.17}$$

```
time = 0                          // dt = 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 10.8: Pseudocode for PID control.

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

Figure 10.9: Pseudocode for feedforward control.

along arbitrary trajectories, not just to a setpoint. The error dynamics (10.17) and proper choice of PID gains ensure exponential decay of trajectory error.

Since $\ddot{\theta}_e = \ddot{\theta}_d - \ddot{\theta}$, to achieve the error dynamics (10.17), we choose the robot's commanded acceleration to be

$$\begin{aligned} \ddot{\theta}_{\text{com}} &= \ddot{\theta}_d - \ddot{\theta}_e \qquad \text{then plug in Equation (10.17) to get} \\ &= \ddot{\theta}_d + K_d\dot{\theta}_e + K_p\theta_e + K_i \int \theta_e(t)dt. \end{aligned} \tag{10.18}$$

Plugging $\ddot{\theta}_{\text{com}}$ into a model of the robot dynamics, we get the *feedback linearizing*

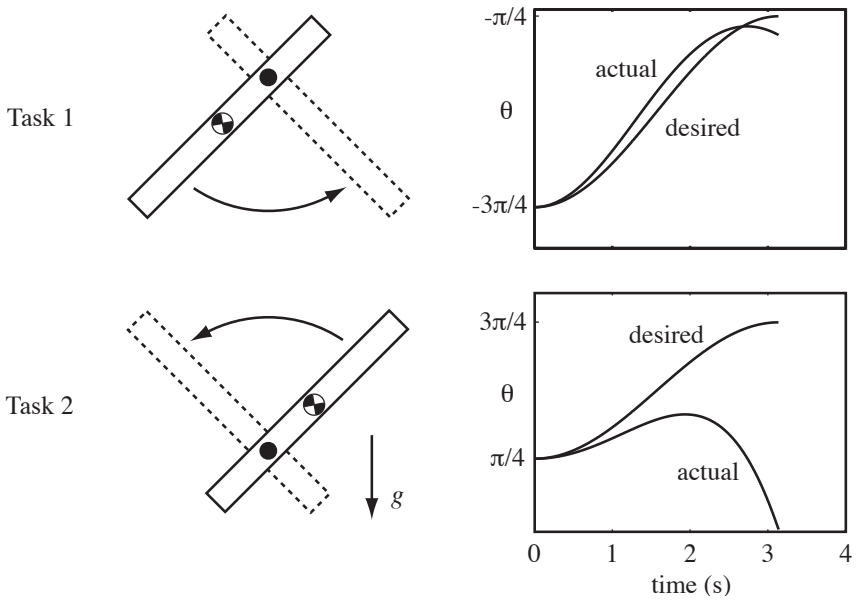

Figure 10.10: Results of feedforward control with an incorrect model: $\hat{r} = 0.08$ m, but $r = 0.1$ m. The desired trajectory in Task 1 is $\theta_d(t) = -\pi/2 - \cos(t)$ for $0 \leq t \leq \pi$. The desired trajectory for Task 2 is $\theta_d(t) = \pi/2 - \cos(t), 0 \leq t \leq \pi$.

controller

$$\boxed{\tau = \widehat{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i\int \theta_e(t)dt + K_d\dot{\theta}_e\right) + \widehat{h}(\theta,\dot{\theta}).} \qquad (10.19)$$

This type of controller is called feedback linearizing because feedback of θ and $\dot{\theta}$ is used to transform the nonlinear control system to a linear one. The $\widehat{h}(\theta,\dot{\theta})$ term cancels dynamics dependent only on the state, and the inertia model $\widehat{M}(\theta)$ converts desired joint accelerations into joint torques, realizing the simple linear error dynamics (10.17). This kind of controller is sometimes called a *computed torque* controller.

A block diagram of the controller is shown in Figure 10.11. The gains K_p, K_i, and K_d are chosen to place the roots of the characteristic equation as desired to achieve good transient response. Under the assumption of a perfect dynamic model, we would choose $K_i = 0$.

Figure 10.12 shows typical behavior of feedback linearizing control relative to feedforward and feedback only. Pseudocode is given in Figure 10.13.

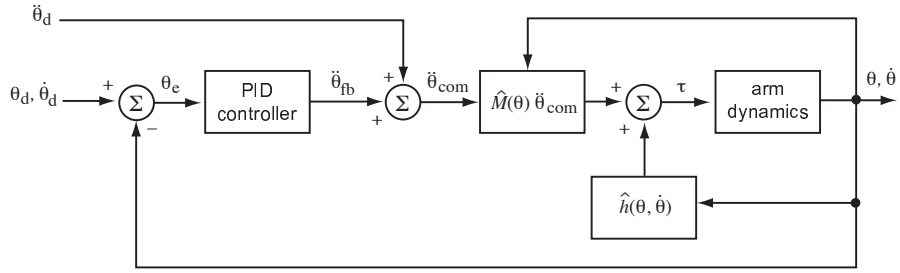

Figure 10.11: Feedback linearizing control. The feedforward acceleration $\ddot{\theta}_d$ is added to the acceleration $\ddot{\theta}_{\text{fb}}$ computed by the PID feedback controller to create the commanded acceleration $\ddot{\theta}_{\text{com}}$.

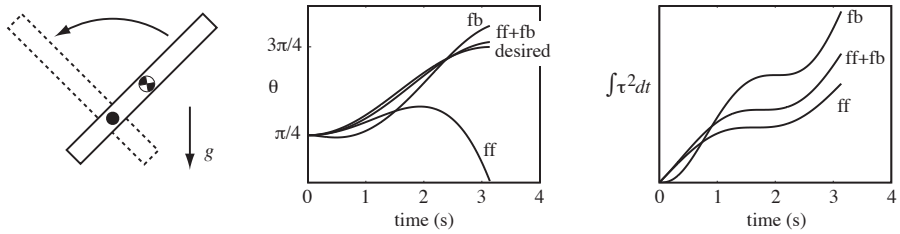

Figure 10.12: Performance of feedforward only (ff), feedback only (fb), and feedback linearizing control (ff+fb). PID gains are taken from Figure 10.6, and the feedforward modeling error is taken from Figure 10.10. The desired motion is Task 2 from Figure 10.10. The middle plot shows the tracking performance of the three controllers. Also plotted is $\int \tau^2(t)dt$, a standard measure of control effort, for each of the three controllers. These plots show typical behavior: the feedback linearizing controller yields better tracking than either feedforward or feedback alone, with less control effort than feedback alone.

10.2.1.4 A Closer Look at Friction

For simplicity, we have been assuming a viscous model of friction at bearings and gears. In reality, friction is a complex phenomenon which is the subject of considerable current research, and any friction model is a gross attempt to capture average behavior of the micromechanics of contact. Friction forces may be a function of the loading force at the contact, the time the contact has been at rest, position (due to spring-like forces before sliding begins, or non-uniformity in bearings), velocity, temperature, etc.

One noteworthy limitation of a viscous friction model is its implication of zero friction force at zero velocity. In fact, common experience indicates that friction forces can be large at zero velocity. For example, you can apply significant horizontal forces to a book on a table before it begins to slide. The force resisting motion at rest is known as *static friction* and is not included in

```
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 = Mhat(q)*(qdotdotd + Kp*e + Kd*edot + Ki*eint) + hhat(q,qdot)
  commandTorque(tau)

  time = time + dt
end loop
```

Figure 10.13: Pseudocode for the feedback linearizing controller.

the viscous model. For a robot joint, nonzero static friction implies that some torque can be applied to the joint at rest without causing any motion. The discontinuity of friction force at zero velocity significantly complicates the problem of controlling low-speed motions, particularly motions involving switches in direction. To address this issue, a more sophisticated model of joint friction can be included in the nonlinear dynamics compensation term $\widehat{h}(\theta, \dot{\theta})$ in feedforward or feedback linearizing control. See Figure 10.14 for some examples of velocity-dependent friction. Other ways of dealing with friction include "dithering" (imposing a high-frequency zero-mean control signal on top of the nominal control signal, to smooth the friction discontinuity at zero velocity) and using larger PID gains in the neighborhood of zero velocity.

10.2.1.5 The Effect of Gearing

Until now we have been considering our actuator as a source of torque, without considering how the actuator generates that torque. For example, if we choose a DC motor with an appropriate power rating for our robot joint, we will likely find that it is capable of producing high speed, up to 10,000 RPM or more, but only low torque. Most robotic applications require significantly lower speeds and higher torques. Therefore, gears, belts and pulleys, and other transmissions are usually used to reduce speed by a gear ratio $G > 1$, while ideally increasing the

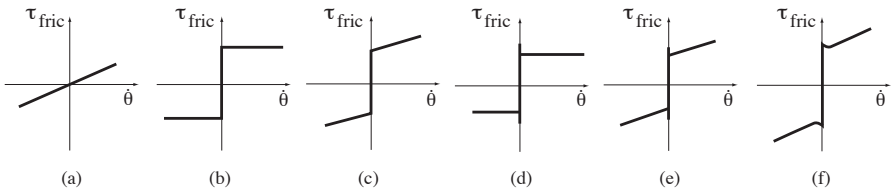

Figure 10.14: Examples of velocity-dependent friction models. (a) Viscous friction. (b) Coulomb friction, $\tau_{\text{fric}} = b\,\text{sgn}(\dot{\theta})$. τ_{fric} can take any value in $[-b, b]$ at zero velocity. (c) Static plus viscous friction, $\tau_{\text{fric}} = b_{\text{static}}\,\text{sgn}(\dot{\theta}) + b_{\text{viscous}}\dot{\theta}$. (d) Static and kinetic friction, requiring $\tau_{\text{fric}} \geq |b_{\text{static}}|$ to initiate motion, and then $\tau_{\text{fric}} = b_{\text{kinetic}}\,\text{sgn}(\dot{\theta})$ during motion, where $b_{\text{static}} > b_{\text{kinetic}}$. (e) Static, kinetic, and viscous friction. (f) A friction law exhibiting the Stribeck effect—at low velocities, friction decreases as velocity increases.

available torque by a factor of G, thereby preserving power:

$$\dot{\theta}_{\text{out}} = \frac{\dot{\theta}_{\text{in}}}{G}, \quad \tau_{\text{out}} = G\tau_{\text{in}}, \quad P_{\text{out}} = \tau_{\text{out}}\dot{\theta}_{\text{out}} = (G\tau_{\text{in}})(\dot{\theta}_{\text{in}}/G) = P_{\text{in}}.$$

In practice, however, some power is dissipated due to friction in the gearing, and the torque available at the output is less than $G\tau_{\text{in}}$. This effect tends to increase with the gear ratio G. Typical choices of G are from single digits to over one hundred.

Another option is to directly drive a robot joint with $G = 1$. Motors used in *direct-drive* configurations typically have much higher power ratings than needed for the application, so that they can provide sufficient torque without gearing. These motors never approach their top-end speeds in typical applications.

Now consider the inertia M of our one-joint robot. It is actually a lumped parameter capturing not only the inertia of the link, but also the inertia of the motor's rotor. Typically the motor's inertia I_{motor} is much smaller than the inertia of the link I_{link}. When there is gearing $G > 1$, however, the motor spins faster than the link; if the link angular velocity is $\dot{\theta}$, then the motor speed is $G\dot{\theta}$. We can write the kinetic energy of the link-rotor system as

$$K = \frac{1}{2}\left(I_{\text{link}}\dot{\theta}^2 + I_{\text{motor}}(G\dot{\theta})^2\right) = \frac{1}{2}\underbrace{(I_{\text{link}} + G^2 I_{\text{motor}})}_{M}\dot{\theta}^2,$$

where $G^2 I_{\text{motor}}$ is the inertia of the motor seen from the gearing output shaft. This is called the motor's *reflected inertia* through the gearbox: the effective inertia of the motor as seen at the output of the gearbox. The time derivative of the kinetic energy $\dot{K}$ is the torque driving the link-rotor system multiplied by the joint velocity, or

$$\dot{K} = (I_{\text{link}} + G^2 I_{\text{motor}})\ddot{\theta}\dot{\theta}.$$

As an example, consider $I_{\text{link}} = 1$ kgm^2 and $I_{\text{motor}} = 10^{-3}$ kgm^2. For $G = 1$, 99.9% of the total inertia is due to the link; only 0.1% of the acceleration torque

accelerates the motor. With a gear ratio $G = 100$, however, the effective inertia of the geared motor is ten times the inertia of the link.

When G is chosen so that $G = \sqrt{I_{\text{link}}/I_{\text{motor}}}$, half of the torque is used to accelerate each of the motor and the link, and the system is said to be *inertia matched*, maximizing the power transferred to the link (Exercise **??**).

Summarizing, we can make two observations comparing direct-drive and highly geared systems:

- The behavior of geared systems is generally less sensitive to changes in the link inertia, as when the arm carries a load, since the link inertia is a smaller percentage of the total inertia due to the high reflected inertia of geared motor.
- Friction forces are larger in highly geared systems. In the limit where friction forces dominate inertial forces, the joint dynamics may be closer to a first-order viscous system than a second-order inertial system.

These properties play an important role in the analysis of control laws for multi-joint systems, discussed next.

10.2.2 Multi-Joint Motion Control

The methods applied above for a single-joint robot carry over directly to n-joint robots. The difference is that the dynamics (10.4) now take the more general vector-valued form

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

where the $n \times n$ positive-definite inertia matrix M is now a function of the configuration θ. We will sometimes find it convenient to explicitly state the components of the term $h(\theta, \dot{\theta})$:

$$\tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + g(\theta) + b(\dot{\theta}), \tag{10.20}$$

where $C(\theta, \dot{\theta})\dot{\theta}$ are Coriolis and centripetal terms, $g(\theta)$ are potential (e.g., gravity) terms, and $b(\dot{\theta})$ are friction terms. In general, the dynamics (10.20) are *coupled*—the force or torque at a joint may be a function of the positions, velocities, and accelerations of other joints.

We distinguish between two types of control of 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.

10.2.2.1 Decentralized Multi-Joint Control

The simplest method for controlling a multi-joint robot, and one that is often used, is to apply an independent controller at each joint. Decentralized control is appropriate when the dynamics of the joints can be decoupled, or at least approximately decoupled. The dynamics are decoupled when the acceleration

of each joint depends only on the torque applied at that joint. This occurs when the inertia matrix is diagonal, as in Cartesian or *gantry* robots, where the first three axes are prismatic and orthogonal along the x-y-z axes. 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 inertia matrix $M(\theta)$ is nearly diagonal, as it is dominated by the reflected inertias of the motors themselves. Variations in $M(\theta)$ due to different joint configurations are small. Significant friction at the individual joints also contributes to the decoupling of the dynamics.

10.2.2.2 Centralized Multi-Joint Control

When gravity forces and torques are significant and coupled, or when the inertia matrix $M(\theta)$ is not well approximated by a diagonal matrix, decentralized control may no longer yield acceptable performance. In this case, the feedback linearizing control law (10.19) of Figure 10.11 can be generalized. 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:

$$\boxed{\tau = \widehat{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i\int \theta_e(t)dt + K_d\dot{\theta}_e\right) + \widehat{h}(\theta,\dot{\theta}).} \tag{10.21}$$

Typically we choose the gain matrices as $k_p\mathcal{I}, k_i\mathcal{I}, k_d\mathcal{I}$, where $\mathcal{I}$ is the $n \times n$ identity matrix and k_p, k_i, and k_d are nonnegative scalars. In the case of an exact dynamics model $\widehat{M}$ and $\widehat{h}$, the dynamics of each joint reduces to the linear dynamics (10.17). The block diagram and pseudocode for this control algorithm are found in Figures 10.11 and 10.13, respectively.

Implementing the control law (10.21) 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 (10.21) can be obtained using only PID control and gravity compensation:

$$\boxed{\tau = K_p\theta_e + K_i\int \theta_e(t)dt + K_d\dot{\theta}_e + \widehat{g}(\theta).} \tag{10.22}$$

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

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} = K_p\theta_e - K_d\dot{\theta}, \tag{10.23}$$

where the Coriolis and centripetal terms are written $C(\theta,\dot{\theta})\dot{\theta}$, and any viscous friction effects are included in K_d for simplicity. We can now define a virtual

"error energy," the sum of an "error potential energy" stored in the virtual spring and an "error kinetic energy":

$$V(\theta_e, \dot{\theta}_e) = \frac{1}{2}\theta_e^T K_p \theta_e + \frac{1}{2}\dot{\theta}_e^T M(\theta)\dot{\theta}_e. \tag{10.24}$$

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

$$V(\theta_e, \dot{\theta}) = \frac{1}{2}\theta_e^T K_p \theta_e + \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta}. \tag{10.25}$$

Taking the time derivative and plugging in (10.23), we get

$$\begin{aligned}\dot{V} &= -\dot{\theta}^T K_p \theta_e + \dot{\theta}^T M(\theta)\ddot{\theta} + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} \\ &= -\dot{\theta}^T K_p \theta_e + \dot{\theta}^T \left(K_p \theta_e - K_d\dot{\theta} - C(\theta,\dot{\theta})\dot{\theta}\right) + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta}.\end{aligned} \tag{10.26}$$

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

$$\begin{aligned}\dot{V} &= -\dot{\theta}^T K_p \theta_e + \dot{\theta}^T \left(K_p \theta_e - K_d\dot{\theta}\right) + \frac{1}{2}\dot{\theta}^T \left(\dot{M}(\theta) - 2C(\theta,\dot{\theta})\right)\dot{\theta} \\ &= -\dot{\theta}^T K_d \dot{\theta} \ \leq 0.\end{aligned} \tag{10.27}$$

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 energy will be dissipated. Thus by the Krasovskii-LaSalle invariance principle (Exercise **??**), the total error energy decreases monotonically and the robot converges to rest at θ_d ($\theta_e = 0$) from any initial state.

10.2.3 Task Space Motion Control

In Section 10.2.2, we focused on motion control in joint space. 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 $(X(t), \mathcal{V}(t))$, where $X \in SE(3)$ or $X \in \mathbb{R}^n$, for example, with $\mathcal{V} \in \mathbb{R}^n$ the velocity. Provided the corresponding trajectory in joint space is feasible and does not pass through a dynamic singularity where $M(\theta)$ loses rank, we now have two options for control: (1) convert to a joint-space trajectory and proceed with control as in Section 10.2.2 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 $X = f(\theta)$ and $\mathcal{V} = J(\theta)\dot{\theta}$, where $J(\theta)$ is the appropriate Jacobian

based on the chosen velocity representation $\mathcal{V}$. Then the joint space trajectory is obtained from the task space trajectory by

$$\text{(inverse kinematics)} \quad \theta(t) = f^{-1}(X(t)) \tag{10.28}$$

$$\dot{\theta}(t) = J^{-1}(\theta(t))\mathcal{V}(t) \tag{10.29}$$

$$\ddot{\theta}(t) = J^{-1}(\theta(t))\left(\dot{\mathcal{V}}(t) - \dot{J}(\theta(t))\dot{\theta}(t)\right). \tag{10.30}$$

If the robot is redundant, i.e., $J(\theta)$ has more columns than rows, a redundancy resolution scheme must be used to solve for f^{-1} and J^{-1}.

A drawback of this approach is that we must calculate the inverse kinematics, which may require significant computing power. The second option is to express the robot's dynamics in task-space coordinates, as discussed in Chapter 9.6. Recall the task-space dynamics

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

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

We can now write a control law in task coordinates inspired by the feedback linearizing control law in joint coordinates (10.21),

$$\boxed{\tau = J^T(\theta)\left(\widehat{\Lambda}(\theta)\left(\dot{\mathcal{V}}_d + K_p X_e + K_i \int X_e(t)dt + K_d \mathcal{V}_e\right) + \widehat{\gamma}(\theta, \mathcal{V}) + \widehat{\eta}(\theta)\right),} \tag{10.31}$$

where $\dot{\mathcal{V}}_d$ is the desired acceleration and $\widehat{\Lambda}$, $\widehat{\gamma}$, and $\widehat{\eta}$ represent the controller's dynamics model.

The task-space control law (10.31) makes use of the configuration error X_e and velocity error $\mathcal{V}_e$. When X is expressed in a minimal set of coordinates ($X \in \mathbb{R}^n$) and $\mathcal{V} = \dot{X}$, a natural choice is $X_e = X_d - X, \mathcal{V}_e = \mathcal{V}_d - \mathcal{V}$. When $X = (R, p) \in SE(3)$, however, there are a number of possible choices, including the following:

- $\mathcal{V} = \mathcal{V}_b$ and $J(\theta) = J_b(\theta)$ in the end-effector frame $\{b\}$. A natural choice would be $X_e = \log_{SE(3)}(X^{-1}X_d)$ and $\mathcal{V}_e = \mathrm{Ad}_{X^{-1}X_d}\mathcal{V}_d - \mathcal{V}$. The expression for X_e gives the body-fixed "direction" from the current configuration X to the desired configuration X_d in the end-effector frame. The transform $\mathrm{Ad}_{X^{-1}X_d}$ transports the desired velocity $\mathcal{V}_d$ from X_d to a velocity expressed in the end-effector frame at X.
- $\mathcal{V} = \mathcal{V}_s$ and $J(\theta) = J_s(\theta)$ in the space frame $\{s\}$. A natural choice would be $X_e = \log_{SE(3)}(X_d X^{-1})$ and $\mathcal{V}_e = \mathcal{V}_d - \mathcal{V}$.
- $\mathcal{V}$ and $J(\theta)$ chosen so that $\mathcal{V} = (\omega, v)$, where ω is the angular velocity of the end-effector relative to $\{s\}$ and $v = \dot{p}$. A natural choice would be

$$X_e = \begin{bmatrix} \log_{SO(3)}(R_d R^T) \\ p_d - p \end{bmatrix} \text{ and } \mathcal{V}_e = \mathcal{V}_d - \mathcal{V}.$$

These choices lead to different behaviors of the robot. In particular, the last choice decouples the rotational and linear corrective terms.

10.3 Force Control

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

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. Force control is dual to motion control in the sense that forces are dual to velocities, with their product being power, an intrinsic, coordinate-free concept.

Let $\mathcal{F}_{\text{app}}$ be the force that the manipulator applies to the environment. The manipulator dynamics can be written

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + g(\theta) + b(\dot{\theta}) + J^T(\theta)\mathcal{F}_{\text{app}} = \tau, \tag{10.32}$$

where the Jacobian $J(\theta)$ satisfies $\mathcal{V} = J(\theta)\dot{\theta}$. Since the robot typically moves slowly (or not at all) during a force control task, we ignore the acceleration and velocity terms to get

$$g(\theta) + J^T(\theta)\mathcal{F}_{\text{app}} = \tau. \tag{10.33}$$

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

$$\tau = \widehat{g}(\theta) + J^T(\theta)\mathcal{F}_d, \tag{10.34}$$

where $\widehat{g}(\theta)$ is the model of gravitational torques and $\mathcal{F}_d$ is the desired force. 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 direct-drive joint, torque control can be achieved by current control of the motor. In the case of a highly geared actuator, however, large friction torque in the gearing degrades 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 directly measure the joint torque, which is fed back to a local controller that modulates the motor current to achieve the desired output torque.

A more common 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 forces $\mathcal{F}_{\text{app}}$ (Figure 10.15). Force-torque measurements are often noisy, so the time derivative of these measurements may not be meaningful. In addition, the desired force $\mathcal{F}_d$ is typically constant or only slowly changing.

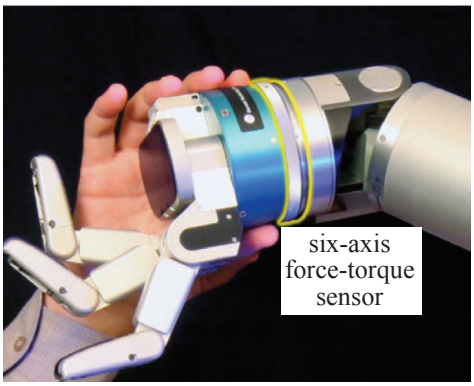

Figure 10.15: A six-axis force-torque sensor, highlighted in yellow, mounted between the Barrett WAM robot arm and its end-effector.

These characteristics suggest a PI controller with a feedforward term and gravity compensation,

$$\boxed{\tau = \widehat{g}(\theta) + J^T(\theta)\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt\right),} \tag{10.35}$$

where $\mathcal{F}_e = \mathcal{F}_d - \mathcal{F}_{\text{app}}$ 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 (10.35) into the dynamics (10.33), we get the error dynamics

$$K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt = 0. \tag{10.36}$$

In the case of a constant force disturbance on the right-hand side of (10.36), arising from an incorrect model of $\widehat{g}(\theta)$, for example, we take the derivative to get

$$K_{fp}\dot{\mathcal{F}}_e + K_{fi}\mathcal{F}_e = 0, \tag{10.37}$$

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

The control law (10.35) is simple and appealing, but potentially dangerous if incorrectly applied. If there is nothing for the robot to push against, the robot 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

$$\boxed{\tau = \widehat{g}(\theta) + J^T(\theta)\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt - K_{\text{damp}}\mathcal{V}\right),} \tag{10.38}$$

where K_{damp} is positive definite.

10.4 Hybrid Motion-Force Control

Most tasks requiring the application of controlled forces also require the application of controlled motions. Control to achieve this is called hybrid motion-force control. 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 they are not independent.

As an example, consider a two-dimensional environment modeled by a damper, $\mathcal{F} = B_{\text{env}}\mathcal{V}$, where

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

Defining the components of $\mathcal{V}$ and $\mathcal{F}$ as $(\mathcal{V}_1, \mathcal{V}_2)$ and $(\mathcal{F}_1, \mathcal{F}_2)$, we have $\mathcal{F}_1 = 2\mathcal{V}_1 + \mathcal{V}_2, \mathcal{F}_2 = \mathcal{V}_1 + \mathcal{V}_2$. We have $n = 2$ freedoms to choose among the $2n = 4$ velocities and forces at any time. For example, we can specify both $\mathcal{F}_1$ and $\mathcal{V}_1$ independently, because B_{env} is not diagonal. Then $\mathcal{V}_2$ and $\mathcal{F}_2$ are determined by B_{env}. We cannot independently control both $\mathcal{F}_1$ and $2\mathcal{V}_1 + \mathcal{V}_2$, however, as these are in the "same direction" according to the damper.

10.4.1 Natural and Artificial Constraints

A particularly interesting case is 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. As an example, consider the task space to have the $n = 6$ dimensions of $SE(3)$. Then a robot opening a cabinet door has $6 - k = 1$ motion freedom, rotation about the cabinet hinges, and therefore $k = 5$ force freedoms—the robot can apply any force and torque that has zero moment about the axis of the hinges.

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; and 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 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 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 friction modeling in Chapter **??**). Within this regime, the robot has three motion freedoms and three force freedoms. Second, the robot could decide to pull away from the board. In this regime, the robot has six motion freedoms and no force freedoms. Thus the configuration of the robot is not the only determinant in the directions of motion and force freedoms. Nonetheless, in this section we consider the simplified case where the motion and force freedoms are determined solely by the

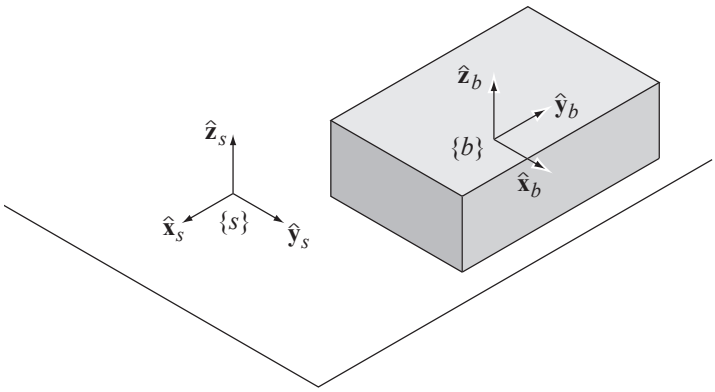

Figure 10.16: The fixed space frame $\{s\}$ attached to the chalkboard and the body frame $\{b\}$ attached to the eraser.

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 chalkboard using an eraser modeled as a rigid block (Figure 10.16). Let the configuration $X(t)$ be expressed in coordinates $q = (\phi, p) = (\phi_x, \phi_y, \phi_z, x, y, z)$, where ϕ are exponential coordinates for the orientation. The velocity is represented as $\mathcal{V} = \dot{q}$. When the eraser is in contact with the board, its configuration $X(t)$ is subject to the constraints

$$\begin{aligned} \phi_x &= 0 \\ \phi_y &= 0 \\ z &= c \end{aligned}$$

where c is half the thickness of the eraser. These constraints can be expressed differentially as

$$\begin{aligned} \dot{\phi}_x &= 0 \\ \dot{\phi}_y &= 0 \\ \dot{z} &= 0 \end{aligned}$$

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

These constraints are called *natural constraints*, specified by the environment. In light of the natural constraints, we can specify any motion of the eraser satisfying these $k = 3$ velocity constraints, giving $6 - k = 3$ motion freedoms. We are also free to specify $k = 3$ forces, $\mathcal{F}_z, \mathcal{F}_{\phi_x}$, and $\mathcal{F}_{\phi_y}$. These motion and force specifications are called *artificial constraints*. Below is a typical set of

artificial constraints corresponding to the natural constraints:

natural constraints	artificial constraints
$\dot{\phi}_x = 0$	$\mathcal{F}_{\phi_x} = 0$
$\dot{\phi}_y = 0$	$\mathcal{F}_{\phi_y} = 0$
$\mathcal{F}_{\phi_z} = 0$	$\dot{\phi}_z = 0$
$\mathcal{F}_x = 0$	$\dot{x} = k_1$
$\mathcal{F}_y = 0$	$\dot{y} = 0$
$\dot{z} = 0$	$\mathcal{F}_z = k_2$

The artificial constraints cause the eraser to move with an x-velocity k_1 while applying a constant force k_2 against the board.

10.4.2 A Hybrid Controller

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

$$A(X)\mathcal{V} = 0, \tag{10.39}$$

where $A(X) \in \mathbb{R}^{k \times n}$. (Alternatively, these constraints could be written in a minimal set of task coordinates as $A(q)\dot{q} = 0$ or in joint coordinates as $A(\theta)\dot{\theta} = 0$.) This formulation includes holonomic and nonholonomic contact constraints with the environment, as well as closure constraints in parallel mechanisms.

If the task-space dynamics of the robot, in the absence of constraints, are

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

then the constrained dynamics are

$$\mathcal{F} = \Lambda(\theta)\dot{\mathcal{V}} + \gamma(\theta, \mathcal{V}) + \eta(\theta) + \underbrace{A^T(X)\lambda}_{\mathcal{F}_{\text{app}}}, \tag{10.40}$$

where $\lambda \in \mathbb{R}^k$ are Lagrange multipliers and $\mathcal{F}_{\text{app}}$ are forces that the robot applies against the constraints. The requested force $\mathcal{F}_d$ must lie in the column space of $A^T(X)$.

Since (10.39) must be satisfied at all times, we can replace (10.39) by the time derivative

$$A(X)\dot{\mathcal{V}} + \dot{A}(X)\mathcal{V} = 0. \tag{10.41}$$

Now, solving (10.40) for $\dot{\mathcal{V}}$, plugging the result into (10.41), and solving for λ, we get

$$\begin{aligned} \lambda &= (A\Lambda^{-1}A^T)^{-1}(A\Lambda^{-1}(F - \gamma - \eta) + \dot{A}\mathcal{V}) \\ &= (A\Lambda^{-1}A^T)^{-1}(A\Lambda^{-1}(F - \gamma - \eta) - A\dot{\mathcal{V}}), \end{aligned} \tag{10.42}$$

where we have plugged in $-A\dot{\mathcal{V}} = \dot{A}\mathcal{V}$ by (10.41). With (10.42), we can calculate the forces $\mathcal{F}_{\text{app}} = A^T(q)\lambda$ the robot applies against the constraints.

Plugging (10.42) into (10.40) and manipulating, the n equations of the constrained dynamics (10.40) can be expressed as the $n-k$ independent motion equations

$$P(X)\mathcal{F} = P(X)(\Lambda(\theta)\dot{\mathcal{V}} + \gamma(\theta, \mathcal{V}) + \eta(\theta)) \tag{10.43}$$

where

$$P = \mathcal{I} - A^T(A\Lambda^{-1}A^T)^{-1}A\Lambda^{-1} \tag{10.44}$$

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

Our hybrid motion-force controller is simply the sum of a task-space motion controller, derived from the feedback linearizing control law (10.31), and a task-space force controller (10.35), each projected to generate forces in its appropriate subspace:

$$\begin{aligned}\tau = J^T(\theta)\Bigg[& \underbrace{P(X)\left(\widehat{\Lambda}(\theta)\left[\dot{\mathcal{V}}_d + K_p X_e + K_i \int X_e(t)dt + K_d \mathcal{V}_e\right]\right)}_{\text{motion control}} \\ & + \underbrace{(\mathcal{I} - P(X))\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt\right)}_{\text{force control}} \\ & + \underbrace{\widehat{\gamma}(\theta, \mathcal{V}) + \widehat{\eta}(\theta)}_{\text{nonlinear compensation}} \Bigg].\end{aligned} \tag{10.45}$$

Because the dynamics of the two controllers are decoupled by the orthogonal projections P and $\mathcal{I} - P$, 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 (10.45) in rigid environments is knowing precisely the constraints $A(X)\mathcal{V} = 0$ 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 based on 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, due to flexibility in the joints and links.

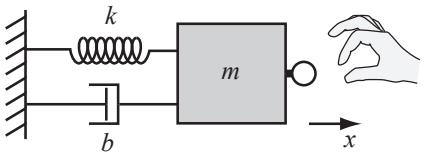

Figure 10.17: A robot creating a one-degree-of-freedom mass-spring-damper virtual environment. The human hand applies a force f to the haptic interface.

10.5 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 (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 mimics particular mass, spring, and damper properties.[1] 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 tissues.

A one-degree-of-freedom environment can be written

$$m\ddot{x} + b\dot{x} + kx = f, \tag{10.46}$$

where x is the position, m is the mass, b is the damping, and k is the stiffness, and f is the force applied to the environment (Figure 10.17). Collectively, we refer to the parameters $\{m, b, k\}$ as the impedance. Loosely, we say that the environment has high impedance if one or more of these parameters, usually including b or k, is large. Similarly, we say that the impedance is low if all of these parameters are small.

More formally, taking the Laplace transform of (10.46), we get

$$(ms^2 + bs + k)X(s) = F(s), \tag{10.47}$$

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 low-frequency response dominated by the spring and high-frequency response dominated by the mass. *Admittance* is the inverse of 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

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

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 small Z implies that motion perturbations produce only small force perturbations.

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

$$D\dot{\mathcal{V}} + B\mathcal{V} + KX = \mathcal{F}_{\text{ext}}, \tag{10.48}$$

where $X \in \mathbb{R}^n$ and $\mathcal{V} = \dot{X}$ are the task-space position and velocity; D, B, and K are the positive-definite virtual inertia, damping, and stiffness to be created by the robot; and $\mathcal{F}_{\text{ext}}$ is a force applied to the robot, perhaps by a user. The values of D, B, and K may change depending on the location in the virtual environment, to represent distinct objects for instance, but we will focus on the case of constant values.

We can replace X and $\mathcal{V}$ in (10.48) by $X(t) - X_{\text{ref}}(t)$ and $\mathcal{V}(t) - \mathcal{V}_{\text{ref}}(t)$. If $X_{\text{ref}}(t)$ is time-varying, it will "pull" the robot approximately along the trajectory. This allows impedance control to achieve motion control, much like a task-space PD controller. Larger stiffness K and damping B achieve more precise trajectory tracking.

There are at least two ways to achieve the behavior (10.48):

- The robot senses motions and commands joint torques to create $-\mathcal{F}_{\text{ext}}$, 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. These robots tend to be lightweight and backdrivable, and are often cable-driven. They are typically good at displaying low impedances, and not as good at displaying stiff, high impedance environments.
- The robot senses $\mathcal{F}_{\text{ext}}$ using a wrist force-torque sensor and controls motions in response. Such a robot is called admittance controlled, as it implements a transfer function $Y(s)$ from forces to motions. These robots tend to be highly geared. They are typically good at displaying stiff, high impedance environments, and not as good at displaying low impedances.

10.5.1 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 these robots are not equipped with a force-torque sensor, and instead rely on their ability to precisely control joint torques with little friction to display the appropriate end-effector force $-\mathcal{F}_{\text{ext}}$ (from (10.48)) to the user. An ideal control law might be

$$\boxed{\tau = J^T(\theta)\left[\underbrace{\widehat{\Lambda}(\theta)\dot{\mathcal{V}} + \widehat{\gamma}(\theta,\mathcal{V}) + \widehat{\eta}(\theta)}_{\text{arm dynamics compensation}} - \underbrace{(D\dot{\mathcal{V}} + B\mathcal{V} + KX)}_{\mathcal{F}_{\text{ext}}}\right].} \tag{10.49}$$

Addition of an end-effector force-torque sensor allows the use of feedback terms to more closely achieve the desired interaction force $-\mathcal{F}_{\text{ext}}$.

In the control law (10.49), it is assumed that $\dot{\mathcal{V}}$, $\mathcal{V}$, and X are measured directly. Measurement of the acceleration $\dot{\mathcal{V}}$ is likely to be noisy, however, and there is the problem of attempting to compensate for the robot's inertia after the acceleration has been sensed. Therefore, it is not uncommon to eliminate the inertial compensation term $\widehat{\Lambda}(\theta)\dot{\mathcal{V}}$ and to set $D = 0$. The inertia of the arm will be apparent to the user, but impedance-controlled manipulators are often designed to be lightweight.

Additionally, instability can arise when (10.49) is used to simulate stiff environments. Small changes in position, as 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 in stiff virtual environments. On the other hand, the effective gains are low when emulating low impedance environments. A lightweight backdrivable manipulator can excel at impedance control.

10.5.2 Admittance Control Algorithm

In an admittance control algorithm, the force $\mathcal{F}_{\text{ext}}$ applied by the user is sensed by the wrist load cell, and the robot responds with an end-effector acceleration satisfying (10.48). A simple approach is to calculate the desired end-effector acceleration $\dot{\mathcal{V}}_d$ according to

$$D\dot{\mathcal{V}}_d + B\mathcal{V} + KX = \mathcal{F}_{\text{ext}},$$

where $(X, \mathcal{V})$ is the current state. Solving, we get

$$\dot{\mathcal{V}}_d = D^{-1}(\mathcal{F}_{\text{ext}} - B\mathcal{V} - KX). \tag{10.50}$$

Given $\dot{\mathcal{V}}_d$, $\mathcal{V}$, and X, integration over the servo timestep Δt yields the desired position and velocity X_d and $\mathcal{V}_d$. The reference $X_d, \mathcal{V}_d, \dot{\mathcal{V}}_d$ can be plugged into the feedback linearizing control law in task coordinates (10.31). To make the response smoother, the force readings can be low-pass filtered.

Simulating a low impedance environment is challenging for an admittance control algorithm, as small forces produce large accelerations. The effective large gains can produce instability. On the other hand, admittance control by a highly geared robot can excel at emulating stiff environments.

10.6 Other Topics

Robust Control While all stable feedback controllers confer some amount of robustness 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 in its inertial properties.

Adaptive Control *Adaptive control* of robots involves estimating the robot's inertia 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. For example, if a robot performs the same pick-and-place operation over and over, 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 execution error toward zero. ILC differs from adaptive control in that the "learned" information is generally nonparametric and focused on a single trajectory.

Passive Compliance and Flexible Manipulators All robots unavoidably have some passive compliance. Models of this compliance could be as simple as a single torsional spring 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 of the attached link, and (2) increased order of the dynamics of the robot. These issues raise challenging problems in control, particularly when 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 position control performance in favor of safety.

Variable Impedance Actuators The impedance of a joint is typically controlled using a feedback control law, as described in Section 10.5. 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 to low-frequency perturbations.

A new class of actuators, called *variable impedance actuators* or *variable stiffness actuators*, aims to give actuators the desired passive mechanical impedance without the bandwidth limitations of an active control law. As an example, a variable stiffness actuator may comprise two motors, allowing the actuator to independently control the mechanical stiffness of the joint (e.g., based on the setpoint of an internal nonlinear spring) and the torque produced by the actuator.

10.7 Summary

- A PID joint-space feedback controller takes the form

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

where $\theta_e = \theta_d - \theta$ and θ_d is the vector of desired joint angles. The proportional term acts like a virtual spring, to pull the robot joints to the desired positions; the derivative term acts like a virtual damper, acting to reduce velocity differences between the actual and desired velocities; and the integral term can be used to eliminate steady-state error in setpoint control.

- The linear error dynamics

$$a_n \theta_e^{(n)} + a_{n-1} \theta_e^{(n-1)} + \ldots + a_2 \ddot{\theta}_e + a_1 \dot{\theta}_e + a_0 \theta_e = 0$$

are stable, i.e., initial errors converge exponentially to zero, if and only if all of the complex roots $s_1, \ldots, s_n$ of the characteristic equation

$$a_n s^n + a_{n-1} s^{n-1} + \ldots + a_2 s^2 + a_1 s + a_0 = 0$$

have real components less than zero, i.e., $\mathrm{Re}(s_i) < 0$ for all $i = 1 \ldots n$.

- 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 system is overdamped if $\zeta > 1$, critically damped if $\zeta = 1$, and underdamped if $\zeta < 1$. The step response of a stable second-order system is generally characterized by its overshoot, 2% settling time, and steady-state error.

- Highly geared robots tend to have greater decoupling of their dynamics due to high friction and large reflected motor inertias.

- The joint-space feedback linearizing (or computed torque) controller is

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

This controller cancels nonlinear terms, uses feedforward control to anticipate the desired acceleration, and uses linear feedback control for stabilization.

- Joint-space PD control plus gravity compensation

$$\tau = K_p \theta_e + K_d \dot{\theta}_e + \widehat{g}(\theta)$$

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

- The task-space feedback linearizing motion controller is

$$\tau = J^T(\theta)\left(\widehat{\Lambda}(\theta)\left(\dot{\mathcal{V}}_d + K_p X_e + K_i \int X_e(t)dt + K_d \mathcal{V}_e\right) + \widehat{\gamma}(\theta, \mathcal{V}) + \widehat{\eta}(\theta)\right).$$

  There are a number of possible choices for calculating X_e and $\mathcal{V}_e$.

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

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

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

- In an n-dimensional task space (where typically $n = 6$), rigid constraints specify $n - k$ free motion directions and k constraint directions in which forces can be applied. These constraints can be represented as $A(X)\mathcal{V} = 0$. A force $\mathcal{F}$ can be partitioned as $\mathcal{F} = P(X)\mathcal{F} + (\mathcal{I} - P(X))\mathcal{F}$, where $P(X)$ projects to forces that move the end-effector tangent to the constraints and $\mathcal{I} - P(X)$ projects to forces against the constraints. The projection matrix $P(X)$ is written in terms of the task-space inertia matrix $\Lambda(\theta)$ and constraints $A(X)$ as

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

- A hybrid motion-force controller using the projection $P(X)$ can be written

$$\begin{aligned}\tau = J^T(\theta)\Bigg[& \underbrace{P(X)\left(\widehat{\Lambda}(\theta)\left[\dot{\mathcal{V}}_d + K_p X_e + K_i \int X_e(t)dt + K_d \mathcal{V}_e\right]\right)}_{\text{motion control}} \\ & + \underbrace{(\mathcal{I} - P(X))\left(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt\right)}_{\text{force control}} \\ & + \underbrace{\widehat{\gamma}(\theta, \mathcal{V}) + \widehat{\eta}(\theta)}_{\text{nonlinear compensation}} \Bigg].\end{aligned}$$

- 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.