4 Motion Control with Torque or Force Inputs

- The robot-control engineers do not want to use the velocity-control modes 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 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.
- The controller generates joint torques and forces to try to track a desired trajectory in joint space or task space.

4.1 Motion Control of a Single Joint

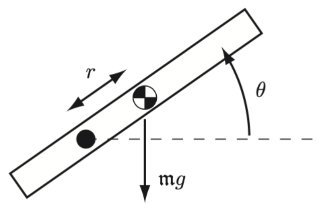

Figure 11.11: A single-joint robot rotating under gravity. The center of mass is indicated by the checkered disk.

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

$$M\ddot{\theta} = \tau - \mathrm{m}gr\cos\theta - b\dot{\theta} \qquad \rightarrow \qquad \tau = M\ddot{\theta} + \mathrm{m}gr\cos\theta + b\dot{\theta}$$

  where M is the scalar 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, b is a viscous friction coefficient due to bearing and transmission, and $g \geq 0$ is the gravitational acceleration.

- Also it may write more compactly as

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

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

Feedback Control: PID Control

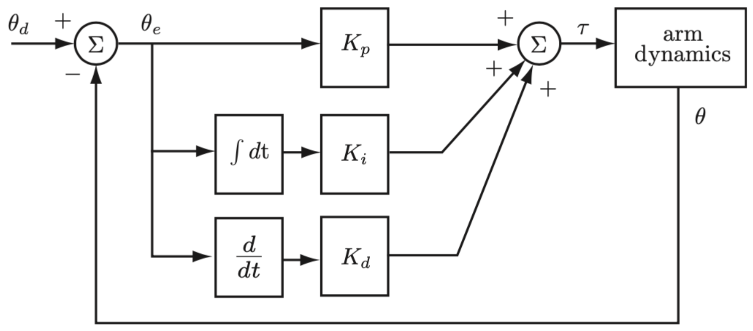

Figure 11.12: Block diagram of a PID controller.

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

$$\tau(t) = K_p\theta_e(t) + K_i\int_0^t \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e(t)$$

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

- The derivative gain K_d acts as a virtual damper that tries to reduce the velocity error $\dot{\theta}_e(t) = \dot{\theta}_d(t) - \dot{\theta}(t)$.

- The integral gain can be used to reduce or eliminate steady-state errors.

PD Control and Second-Order Error Dynamics

- Consider the case where $K_i = 0$, and assume the robot moves in a horizontal plane ($g = 0$).
- Substituting the PD control law into the dynamics, we get

$$M\ddot{\theta} + b\dot{\theta} = K_p\theta_e + K_d\dot{\theta}_e$$

- Take the set-point control task having $\theta_d = constant$ and $\dot{\theta}_d = \ddot{\theta}_d = 0$, so $\ddot{\theta}_e = -\ddot{\theta}$ and $\dot{\theta}_e = -\dot{\theta}$

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

- In the standard second-order form,

$$\ddot{\theta}_e + \frac{(b+K_d)}{M}\dot{\theta}_e + \frac{K_p}{M}\theta_e = 0 \qquad \rightleftarrows \qquad \ddot{\theta}_e + 2\zeta\omega_n\dot{\theta}_e + \omega_n^2\theta_e = 0$$

  where the damping ratio ζ and the natural frequency ω_n are

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

- Since the error dynamics equation is stable, 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 and so on.

PID Control and Third-Order Error Dynamics

- 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

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

- At the steady-state,

$$K_p\theta_e = \mathrm{m}gr\cos\theta$$

  The joint comes to rest at a configuration θ satisfying $K_p\theta_e = \mathrm{m}gr\cos\theta$, i.e., the final error θ_e is nonzero.

- 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 $K_i > 0$. To see how this works, write down the set-point error dynamics

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

  where it is noted that $\tau_{dist} = \mathrm{m}gr\cos\theta$ can be constant at the steady-state.

- Taking derivatives of both sides, we get the third-order error dynamics

$$M\dddot{\theta}_e + (b + K_d)\ddot{\theta}_e + K_p\dot{\theta}_e + K_i\theta_e = \dot{\tau}_{dist} = 0$$

- Its characteristic equation is

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

- (Routh Stability Criterion) Necessary condition: $M > 0$ $b > 0$ $K_p > 0$ $K_d > 0$ $K_i > 0$
  Sufficient condition: the first column should be positive

$$\begin{array}{ll} s^3 : 1 & \dfrac{K_p}{M} \\ s^2 : \dfrac{b + K_d}{M} & \dfrac{K_i}{M} \\ s^1 : \dfrac{K_p}{M} - \dfrac{K_i}{b + K_d} & \\ s^0 : \dfrac{K_i}{M} & \end{array}$$

As a result, the new gain K_i must satisfy both a lower and an upper bound

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

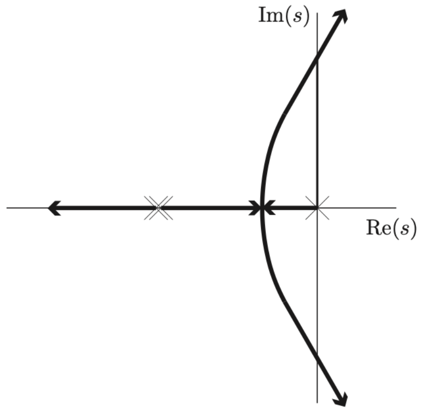

Figure 11.14: The movement of the three roots of Equation (11.30) as K_i increases from zero. First a PD controller is chosen with K_p and K_d yielding critical damping, giving rise to two collocated roots on the negative real axis. Adding an infinitesimal gain $K_i > 0$ creates a third root at the origin. 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.

- (Root Locus) After setting the critical damping $\zeta = 1$, the characteristic equation is rearranged into

$$s^3 + \frac{b + K_d}{M}s^2 + \frac{K_p}{M}s + \frac{K_i}{M} = 0 \qquad \rightarrow \qquad 1 + K_i L(s) = 1 + K_i \frac{1}{s(s+\omega_n)^2} = 0$$

where, use the matlab function *rlocus* if you are not good at the root locus method.

- In the case of actuator saturation due to integral control, integrator anti-windup places a limit on how large the error integral is allowed to grow.

- Pseudocode for the PID control algorithm

```
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 PID controller applies well to trajectory following, but the integral control will not eliminate tracking error along arbitrary trajectories.

Feedforward Plus Feedback Linearization

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

$$\tau = M(\theta)\ddot{\theta} + h(\theta, \dot{\theta}) \qquad \rightarrow \qquad \tau = \widetilde{M}(\theta_d)\ddot{\theta}_d + \widetilde{h}(\theta_d, \dot{\theta}_d)$$

  where $\widetilde{M}, \widetilde{h}$ are models for M, h, respectively.

- All practical controllers use feedback + feedforward because a good model can be used to improve performance and simplify analysis.

- Let us assign the desired error dynamics as

$$\ddot{\theta}_e + K_d\dot{\theta}_e + K_p\theta_e + K_i\int \theta_e(\mathrm{t})d\mathrm{t} = 0$$

  and then let us choose the acceleration from the desired error dynamics:

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

  and finally let us apply the acceleration into the inverse dynamics equation

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

  This controller is called as the inverse dynamics controller or the computed torque controller.

- $\widetilde{M}(\theta)$ transforms the accelerations into the joint torques.

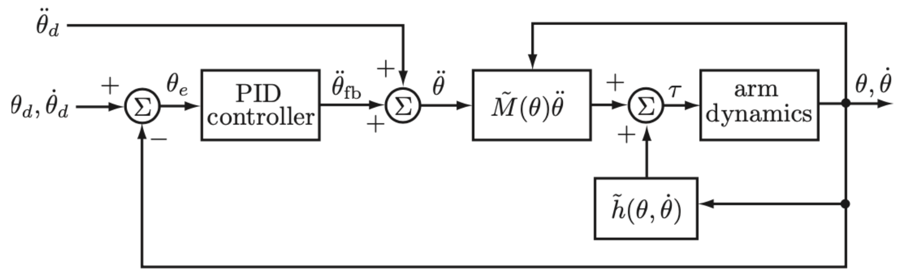
$\ddot{\theta}_d$
$\theta_d, \dot{\theta}_d$
$+$
$-$
θ_e
PID controller
$\ddot{\theta}_{\text{fb}}$
$+$
$+$
$\ddot{\theta}$
$\tilde{M}(\theta)\ddot{\theta}$
$+$
$+$
τ
arm dynamics
$\theta, \dot{\theta}$
$\tilde{h}(\theta, \dot{\theta})$

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 $\ddot{\theta}$.

```
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.

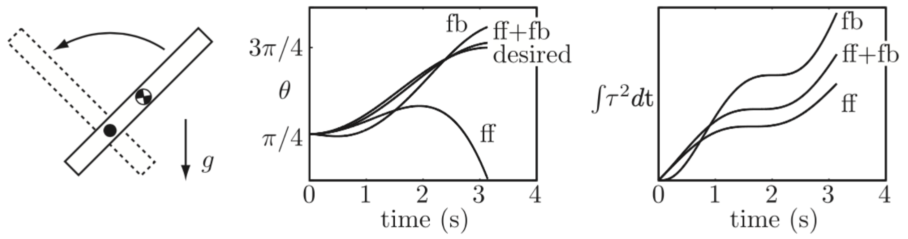

Figure 11.19: Performance of feedforward only (ff), feedback only (fb), and computed torque control (ff+fb). The PID gains are taken from Figure 11.13, and the feedforward modeling error is taken from Figure 11.17. The desired motion is Task 2 from Figure 11.17 (left-hand plot). The center plot shows the tracking performance of the three controllers. The right-hand plot shows $\int \tau^2(t)dt$, 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.

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.
- In general, the components of the dynamics 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
  - centralized control, where full state information for each of the n joints is available to calculate the controls for each joint.

Decentralized Multi-joint Control

- The simplest method for controlling a multi-joint robot is to apply at each joint an independent controller

$$\tau(t) = K_p\theta_e(t) + K_i \int_0^t \theta_e(\mathrm{t})d\mathrm{t} + K_d\dot{\theta}_e(t)$$

  where K_P, K_i, K_d are diagonal matrices.
- Decentralized control is appropriate when the dynamics are decoupled, at least approximately. For example,
  - gantry robots
  - highly geared robots in the absence of gravity.

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 can be generalized to a multi-joint robot.

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

Global Asymptotic Stability by PD Set-Point Control in absence of Gravity

- When PD setpoint control is applied, the controlled dynamics can be written as

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

  where the Coriolis and centripetal terms are written $C(\theta, \dot{\theta})\dot{\theta}$.
- We can now define a virtual error energy stored in the control system

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

- Taking the time derivative and substituting the controlled dynamics into it, we get

$$\begin{aligned}
\dot{V}(\theta_e, \dot{\theta}_e) &= -\dot{\theta}^T K_p \theta_e + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} + \dot{\theta}^T M(\theta)\ddot{\theta} \\
&= -\dot{\theta}^T K_p \theta_e + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} + \dot{\theta}^T[-C(\theta,\dot{\theta})\dot{\theta} + K_p\theta_e - K_d\dot{\theta}] \\
&= \frac{1}{2}\dot{\theta}^T[\dot{M}(\theta) - 2C(\theta,\dot{\theta}]\dot{\theta} - \dot{\theta}^T K_d \dot{\theta} \\
&= -\dot{\theta}^T K_d \dot{\theta}
\end{aligned}$$

- This shows that the error energy is decreasing when $\dot{\theta} \neq 0$.
- When $\dot{\theta} = \ddot{\theta} = 0$ and $\theta \neq \theta_d$, consider the controlled system

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} = K_p\theta_e - K_d\dot{\theta} \qquad \rightarrow \qquad K_p\theta_e = 0$$

  where the virtual spring K_p ensures that $\theta = \theta_d$
- By the Krasovskii-LaSalle invariance principle, the total error energy decreases monotonically and the robot converges to rest at θ_d $(\theta_e = 0)$ from any initial state.

4.3 Task-Space Motion Control

- In the joint-space control, trajectories are naturally described by the joint variables, and there are no issues of singularities or redundancy.
- If the robot interacts with the external environment and objects, 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 $(X_{sb}(t), \mathcal{V}_b(t))$ where $X_{sb} \in SE(3)$ and $[\mathcal{V}_b] = X_{sb}^{-1}\dot{X}_{sb}$, i.e., the twist $\mathcal{V}_b$ is expressed in the end-effector frame {b}.
- The first option is to convert the trajectory to joint space.
  - The forward kinematics are $X_{sb} = T(\theta)$ and $\mathcal{V}_b = J_b(\theta)\dot{\theta}$.
  - Then the joint-space trajectory is obtained from the task-space trajectory using inverse kinematics (IK) (Chapter 6):

$$\textbf{(IK)} \qquad \begin{aligned} \theta(t) &= T^{-1}(X(t)) \\ \dot{\theta}(t) &= J_b^{+}(\theta(t))\mathcal{V}_b(t) \\ \ddot{\theta}(t) &= J_b^{+}(\theta(t))[\dot{\mathcal{V}}_b(t) - \dot{J}_b(\theta(t))\dot{\theta}] \end{aligned}$$

  - A drawback of this approach is that we must calculate the inverse kinematics, $J_b^{+}(\theta)$ and $\dot{J}_b(\theta)$, 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 and consider the error dynamics in task space

$$\begin{aligned}\mathcal{F}_b &= \Lambda(\theta)\dot{\mathcal{V}}_b + \eta(\theta, \mathcal{V}_b)\\ 0 &= \left(\frac{d}{dt}([Ad_{X_{sb}^{-1}X_{sd}}]\mathcal{V}_d) - \dot{\mathcal{V}}\right) + K_d\left([Ad_{X_{sb}^{-1}X_{sd}}]\mathcal{V}_d - \mathcal{V}\right) + K_pX_e + K_i\int X_e(\mathrm{t})d\mathrm{t}\end{aligned}$$

  - The joint forces and torques τ are related to the wrenches $\mathcal{F}_b$ expressed in the end-effector frame by $\tau = J_b^T(\theta)\mathcal{F}_b$.
  - We can now write a control law in task space inspired by the computed torque control law in joint coordinates,

$$\begin{aligned}\tau &= J_b^T(\theta)\left[\widetilde{\Lambda}(\theta)\dot{\mathcal{V}}_b + \widetilde{\eta}(\theta, \mathcal{V}_b)\right]\\ &= J_b^T(\theta)\left[\widetilde{\Lambda}(\theta)\left(\frac{d}{dt}([Ad_{X_{sb}^{-1}X_{sd}}]\mathcal{V}_d) + K_pX_e + K_i\int X_e(\mathrm{t})d\mathrm{t} + K_d\mathcal{V}_e\right) + \widetilde{\eta}(\theta, \mathcal{V}_b)\right]\end{aligned}$$

    * $\frac{d}{dt}([Ad_{X_{sb}^{-1}X_{sd}}]\mathcal{V}_d)$ is the feedforward acceleration expressed in the actual end-effector frame at X_{sb} (this term can be approximated as $\dot{\mathcal{V}}_d$ at states close to the reference state).
    * The configuration error X_e satisfies $[X_e] = \log(X_{sb}^{-1}X_{sd})$: X_e is the twist, expressed in the end-effector frame, which, if followed for unit time, would move the current configuration X_{sb} to the desired configuration X_{sd}.
    * The velocity error is calculated as $\mathcal{V}_e = [Ad_{X_{sb}^{-1}X_{sd}}]\mathcal{V}_d - \mathcal{V}$
    * The transform $[Ad_{X_{sb}^{-1}X_{sd}}]$ expresses the reference twist $\mathcal{V}_d$, which is expressed in the frame X_{sd}, as a twist in the end-effector frame at X_{sb}, in which the actual velocity $\mathcal{V}$ is represented, so the two expressions can be differenced.