3.4 Computed-Torque Control

Through the years there have been proposed many sorts of robot control schemes. As it happens, most of them can be considered as special cases of the class of *computed-torque controllers.* Computed torque, at the same time, is a special application of *feedback linearization* of nonlinear systems, which has gained popularity in modern systems theory [Hunt et al. 1983, Gilbert and Ha 1984]. In fact, one way to classify robot control schemes is to divide them as "computed-torque-like" or "noncomputed-torque-like." Computed-torque-like controls appear in robust control, adaptive control, learning control, and so on.

In the remainder of this chapter we explore this class of robot controllers, which includes such a broad range of designs. Computed-torque control allows us to conveniently derive very effective robot controllers, while providing a framework to bring together classical independent joint control and some modern design techniques, as well as set the stage for the rest of the book. A summary of the different computed-torque-like controllers is given at the end of the section in Table 3.4-1. We shall see that many digital robot controllers are also computed-torque-like controllers (Section 3.5).

Derivation of Inner Feedforward Loop

The robot arm dynamics are

$$M(q)\ddot{q} + V(q,\dot{q}) + F_v\dot{q} + F_d(\dot{q}) + G(q) + \tau_d = \tau \tag{3.4-1}$$

or

$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau, \tag{3.4-2}$$

with the joint variable $q(t) \in R^n$, $\tau(t)$ the control torque, and $\tau_d(t)$ a disturbance. If this equation includes motor actuator dynamics (Section 2.6), then $\tau(t)$ is an input voltage.

Suppose that a desired trajectory $q_d(t)$ has been selected for the arm motion, according to the discussion in Section 3.2. To ensure trajectory tracking by the joint variable, define an output or *tracking error* as

$$e(t) = q_d(t) - q(t). \tag{3.4-3}$$

To demonstrate the influence of the input $\tau(t)$ on the tracking error, differentiate twice to obtain

$$\dot{e} = \dot{q}_d - \dot{q}$$
$$\ddot{e} = \ddot{q}_d - \ddot{q}.$$

Solving now for $\ddot{q}$ in (3.4-2) and substituting into the last equation yields

$$\ddot{e} = \ddot{q}_d + M^{-1}(N + \tau_d - \tau). \tag{3.4-4}$$

Defining the control input function

$$u = \ddot{q}_d + M^{-1}(N - \tau) \tag{3.4-5}$$

and the disturbance function

$$w = M^{-1}\tau_d \tag{3.4-6}$$

we may define a state $x(t) \in \mathbb{R}^{2n}$ by

$$x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{3.4-7}$$

and write the *tracking error dynamics* as

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} w. \tag{3.4-8}$$

This is a linear error system in Brunovsky canonical form consisting of n pairs of double integrators $1/s^2$, one per joint. It is driven by the control input $u(t)$ and the disturbance $w(t)$. Note that this derivation is a special case of the general feedback linearization procedure in Section 2.4.

The feedback linearizing transformation (3.4-5) may be inverted to yield

$$\tau = M(\ddot{q}_d - u) + N. \tag{3.4-9}$$

We call this the *computed-torque control law.* The importance of these manipulations is as follows. There has been no state-space transformation in going from (3.4-1) to (3.4-8). Therefore, if we select a control $u(t)$ that stabilizes (3.4-8) so that $e(t)$ goes to zero, then the nonlinear control input $\tau(t)$ given by (3.4-9) will cause trajectory following in the robot arm (3.4-1). In fact, substituting (3.4-9) into (3.4-2) yields

$$M\ddot{q} + N + \tau_d = M(\ddot{q}_d - u) + N$$

or

$$\ddot{e} = u + M^{-1}\tau_d, \tag{3.4-10}$$

which is exactly (3.4-8).

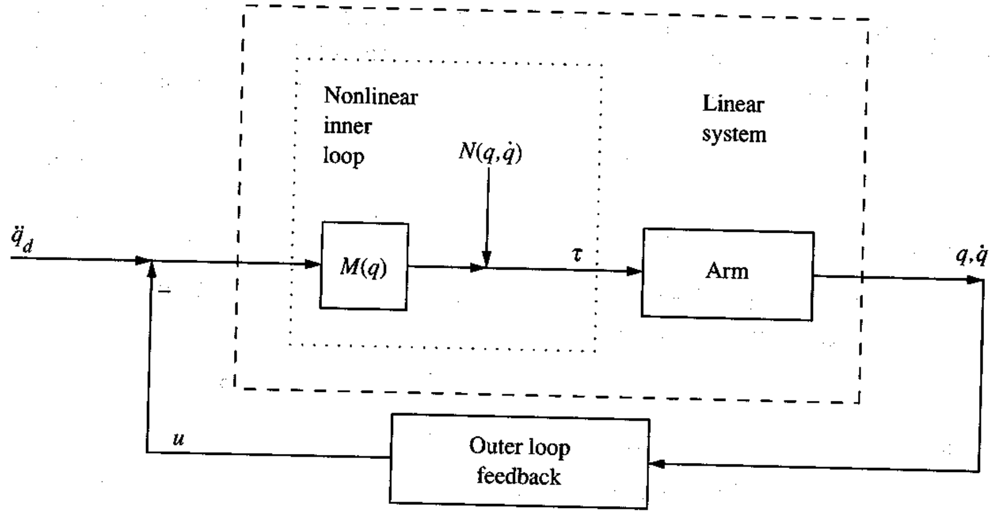

FIGURE 3.4-1 Computed-torque control scheme, showing inner and outer loops.

The stabilization of (3.4-8) is not difficult. In fact, the nonlinear transformation (3.4-5) has converted a complicated nonlinear controls design problem into a simple design problem for a linear system consisting of n decoupled subsystems, each obeying Newton's laws.

The resulting control scheme appears in Fig. 3.4-1. It is important to note that it consists of an *inner nonlinear loop* plus an *outer control signal u(t)*. We shall see several ways for selecting $u(t)$. Since $u(t)$ will depend on $q(t)$ and $\dot{q}(t)$, the outer loop will be a feedback loop. In general, we may select a dynamic compensator $H(s)$ so that

$$U(s) = H(s)E(s). \tag{3.4-11}$$

$H(s)$ can be selected for good closed-loop behavior. According to (3.4-10), the closed-loop error system then has transfer function

$$T(s) = s^2I - H(s). \tag{3.4-12}$$

It is important to realize that computed-torque depends on the inversion of the robot dynamics, and indeed is sometimes called *inverse dynamics control.* In fact, (3.4-9) shows that $\tau(t)$ is computed by substituting $\ddot{q}_d - u$ for $\ddot{q}$ in (3.4-2); that is, by solving the robot *inverse dynamics problem.* The caveats associated with system inversion, including the problems resulting when the system has non-minimum-phase zeros, all apply here. (Note that in the linear case, the system zeros are the poles of the inverse. Such non-minimum-phase notions generalize to nonlinear systems.) Fortunately for us, the rigid arm dynamics are minimum phase.

There are several ways to compute (3.4-9) for implementation purposes. Formal matrix multiplication at each sample time should be avoided. In some cases the expression may be worked out analytically. A good way to compute the torque $\tau(t)$ is to use the efficient Newton–Euler inverse dynamics formulation [Craig 1989] with $\ddot{q}_d - u$ in place of $\ddot{q}(t)$.

The outer-loop signal $u(t)$ can be chosen using many approaches, including robust and adaptive control techniques. In the remainder of this chapter we explore some choices for $u(t)$ and some variations on computed-torque control.

PD Outer-Loop Design

One way to select the auxiliary control signal $u(t)$ is as the proportional-plus-derivative (PD) feedback,

$$u = -K_v\dot{e} - K_p e. \tag{3.4-13}$$

Then the overall robot arm input becomes

$$\tau = M(q)\,(\ddot{q}_d + K_v\dot{e} + K_p e) + N(q,\dot{q}). \tag{3.4-14}$$

This controller is shown in Fig. 3.4-6 with $K_i = 0$.

The closed-loop error dynamics are

$$\ddot{e} + K_v\dot{e} + K_p e = w, \tag{3.4-15}$$

or in state-space form,

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} w. \tag{3.4-16}$$

The closed-loop characteristic polynomial is

$$\Delta_c(s) = |s^2 I + K_v s + K_p|. \tag{3.4-17}$$

Choice of PD Gains. It is usual to take the $n \times n$ gain matrices diagonal so that

$$K_v = \text{diag}\{k_{v_i}\}, \qquad K_p = \text{diag}\{k_{p_i}\}. \tag{3.4-18}$$

Then

$$\Delta_c(s) = \prod_{i=1}^{n} (s^2 + k_{v_i}s + k_{p_i}), \tag{3.4-19}$$

and the error system is asymptotically stable as long as the k_{v_i} and k_{p_i} are all positive. Therefore, as long as the disturbance $w(t)$ is bounded, so is the error $e(t)$. In connection with this, examine (3.4-6) and recall from Table 2.3-1 that M^{-1} is upper bounded. Thus boundedness of $w(t)$ is equivalent to boundedness of $\tau_d(t)$.

It is important to note that although selecting the PD gain matrices diagonal results in decoupled control at the outer-loop level, it does not result in a decoupled joint-control strategy. This is because multiplication by $M(q)$ and addition of the nonlinear feedforward terms $N(q,\dot{q})$ in the inner loop scrambles the signal $u(t)$ among all the joints. Thus, information on all joint positions $q(t)$ and velocities $\dot{q}(t)$ is generally needed to compute the control $\tau(t)$ for any one given joint.

The standard form for the second-order characteristic polynomial is

$$p(s) = s^2 + 2\zeta\omega_n s + \omega_n^2, \tag{3.4-20}$$

with ζ the damping ratio and ω_n the natural frequency. Therefore, desired performance in each component of the error $e(t)$ may be achieved by selecting the PD gains as

$$k_{p_i} = \omega_n^2, \qquad k_{v_i} = 2\zeta\omega_n, \tag{3.4-21}$$

with ζ, ω_n the desired damping ratio and natural frequency for joint error i. It may be useful to select the desired responses at the end of the arm faster than near the base, where the masses that must be moved are heavier.

It is undesirable for the robot to exhibit overshoot, since this could cause impact if, for instance, a desired trajectory terminates at the surface of a workpiece. Therefore, the PD gains are usually selected for *critical damping* $\zeta = 1$. In this case

$$k_{v_i} = 2\sqrt{k_{p_i}}, \qquad k_{p_i} = k_{v_i}^2/4. \tag{3.4-22}$$

Selection of the Natural Frequency. The natural frequency ω_n governs the speed of response in each error component. It should be large for fast responses and is selected depending on the performance objectives. Thus the desired trajectories should be taken into account in selecting ω_n. We discuss now some additional factors in this choice.

There are some *upper limits* on the choice for ω_n [Paul 1981]. Although the links of most industrial robots are massive, they may have some flexibility. Suppose that the frequency of the first flexible or resonant mode of link i is

$$\omega_r = \sqrt{k_r/J} \tag{3.4-23}$$

with J the link inertia and k_r the link stiffness. Then, to avoid exciting the resonant mode, we should select $\omega_n < \omega_r/2$. Of course, the link inertia J changes with the arm configuration, so that its maximum value might be used in computing ω_r.

Another upper bound on ω_n is provided by considerations on actuator saturation. If the PD gains are too large, the torque $\tau(t)$ may reach its upper limits.

Some more feeling for the choice of the PD gains is provided from error-boundedness considerations as follows. The transfer function of the closed-loop error system in (3.4-15) is

$$e(s) = (s^2 I + K_v s + K_p)^{-1} w(s), \tag{3.4-24}$$

or if K_v and K_p are diagonal,

$$e_i(s) = \frac{1}{s^2 + k_{v_i}s + k_{p_i}} w(s) \equiv H(s)w(s) \tag{3.4-25}$$

$$\dot{e}_i(s) = \frac{s}{s^2 + k_{v_i}s + k_{p_i}} w(s) \equiv sH(s)w(s). \tag{3.4-26}$$

We assume that the disturbance and M^{-1} are bounded (Table 2.3-1), so that

$$\|w\| \le \|M^{-1}\| \, \|\tau_d\| \le \overline{m}\overline{d}, \tag{3.4-27}$$

with $\overline{m}$ and $\overline{d}$ known for a given robot arm. Therefore,

$$\|e_i(t)\| \le \|H(s)\| \, \|w\| \le \|H(s)\| \overline{m}\overline{d} \tag{3.4-28}$$

$$\|\dot{e}_i(t)\| \le \|sH(s)\| \, \|w\| \le \|sH(s)\| \overline{m}\overline{d}. \tag{3.4-29}$$

Now selecting the L_2-norm, the operator gain $\| H(s) \|_2$ is the maximum value of the Bode magnitude plot of $H(s)$ (Section 1.4). For a critically damped system,

$$\sup_{\omega} \|H(j\omega)\|_2 = 1/k_{p_i}. \tag{3.4-30}$$

Therefore,

$$\|e_i(t)\|_2 \le \overline{m}\overline{d}/k_{p_i}. \tag{3.4-31}$$

Moreover (see the Problems),

$$\sup_{\omega} \|j\omega H(j\omega)\|_2 = 1/k_{v_i}, \tag{3.4-32}$$

so that

$$\|\dot{e}_i(t)\|_2 \le \overline{m}\overline{d}/k_{v_i}. \tag{3.4-33}$$

Thus, in the case of critical damping, the position error decreases with k_{p_i} and the velocity error decreases with k_{v_i}.

EXAMPLE 3.4-1: Simulation of PD Computed-Torque Control

In this example we intend to show the detailed mechanics of simulating a PD computed-torque controller on a digital computer.

a. Computed-Torque Control Law

In Example 2.2-2 we found the dynamics of the two-link planar elbow arm shown in Fig. 3.2-1 to be

$$\begin{bmatrix} (m_1+m_2)a_1^2 + m_2a_2^2 + 2m_2a_1a_2\cos\theta_2 & m2a_2^2 + m_2a_1a_2\cos\theta_2 \\ m_2a_2^2 + m_2a_1a_2\cos\theta_2 & m2a_2^2 \end{bmatrix} \begin{bmatrix} \ddot\theta_1 \\ \ddot\theta_2 \end{bmatrix} \tag{1}$$

$$+\begin{bmatrix} -m_2a_1a_2(2\dot\theta_1\dot\theta_2 + \dot\theta_2^2)\sin\dot\theta_2 \\ m_2a_1a_2\dot\theta_1^2 \sin\theta_2 \end{bmatrix} + \begin{bmatrix} (m_1+m_2)ga_1\cos\theta_1 + m_2ga_2\cos(\theta_1+\theta_2) \\ m_2ga_2\cos(\theta_1+\theta_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}.$$

These are in the standard form

$$M(q)\ddot q + V(q,\dot q) + G(q) = \tau. \tag{2}$$

Take the link masses as 1 kg and their lengths as 1 m.

The PD computed-torque control law is given as

$$\tau = M(q)(\ddot q_d + K_v\dot e + K_pe) + V(q,\dot q) + G(q), \tag{3}$$

with the tracking error defined as

$$e = q_d - q. \tag{4}$$

b. Desired Trajectory

Let the desired trajectory $q_d(t)$ have the components

$$\begin{aligned} \theta_{1d} &= g_1 \sin(2\pi t/T) \\ \theta_{2d} &= g_2 \cos(2\pi t/T) \end{aligned} \tag{5}$$

with period $T = 2$ s and amplitudes $g_i = 0.1$ rad ≈ 6 deg. For good tracking select the time constant of the closed-loop system as 0.1 s. For critical damping, this means that $K_v = \text{diag}\{k_v\}$, $K_p = \text{diag}\{k_p\}$, where

$$\begin{aligned} &\omega_n = 1/0.1 = 10 \\ &k_p = \omega_n^2 = 100, \qquad k_v = 2\omega_n = 20. \end{aligned} \tag{6}$$

It is important to realize that the selection of controller parameters such as the PD gains depends on the performance objectives—in this case, the period of the desired trajectory.

c. Computer Simulation

Let us simulate the computed-torque controller using program TRESP in Appendix B. Simulation using commercial packages such as MATLAB and SIMNON is quite similar.

The subroutines needed for TRESP are shown in Fig. 3.4-2. They are worth examining closely. Subroutine SYSINP (IT,x,t) is called once per Runge–Kutta integration period and generates the reference trajectory $q_d(t)$, as well as $\dot q_d(t)$ and $\ddot q_d(t)$. Note that the reference signal should be held constant during each integration period.

```
C   FILE 2lnkct.FOR
C   IMPLEMENTATION OF COMPUTED-TORQUE CONTROLLER ON 2-LINK PLANAR ARM
C   SUBROUTINES FOR USE WITH TRESP
C
C   SUBROUTINE TO COMPUTE DESIRED TRAJECTORY
c      The trajectory value must be constant within each Runge-Kutta
c         integration interval
C
       SUBROUTINE SYSINP(IT,x,t)
       REAL x(*)
       COMMON/TRAJ/qd(6), qdp(6), qdpp(6)
       DATA gl, g2, per, twopi/0.1, 0.1, 2., 6.283/

C   COMPUTE DESIRED TRAJECTORY qd(t), qdp(t), qdpp(t)
       fact=  twopi/per
       qd(1)=    gl*sin(fact*t)
       qd(2)=    g2*cos(fact*t)
       qdp(1)=   gl*fact*cos(fact*t)
       qdp(2)=  -g2*fact*sin(fact*t)
       qdpp(1)= -gl*fact**2*sin(fact*t)
       qdpp(2)= -g2*fact**2*cos(fact*t)
C
       RETURN
       END
C
C
C ************************************************************
C   MAIN SUBROUTINE CALLED BY RUNGE-KUTTA INTEGRATOR
       SUBROUTINE F(t,x,xp)
       REAL x(*), xp(*)
C
C   COMPUTED-TORQUE CONTROLLER
       CALL CTL(x)

C   ROBOT ARM DYNAMICS
       CALL ARM(x,xp)
C
       RETURN
       END
C
C
C ************************************************************
C   COMPUTED-TORQUE CONTROLLER SUBROUTINE
       SUBROUTINE CTL(x)
       REAL x(*),m1,m2,M11,M12,M22,N1,N2,kp,kv
       COMMON/CONTROL/t1, t2
C   The next line is to plot the errors and torques
       COMMON/OUTPUT/ e(2), ep(2), tp1, tp2
       COMMON/TRAJ/qd(6), qdp(6), qdpp(6)
       DATA m1,m2,a1,a2, g/1.,1.,1.,1., 9.8/
       DATA kp, kv/ 100,20/

C   COMPUTE TRACKING ERRORS
       e(1) = qd(1)  - x(1)
       e(2) = qd(2)  - x(2)
       ep(1)= qdp(1) - x(3)
       ep(2)= qdp(2) - x(4)

C   COMPUTATION OF M(q), N(q,qp)
       M11= (m1+m2)*a1**2 + m2*a2**2 + 2*m2*a1*a2*cos(x(2))
```