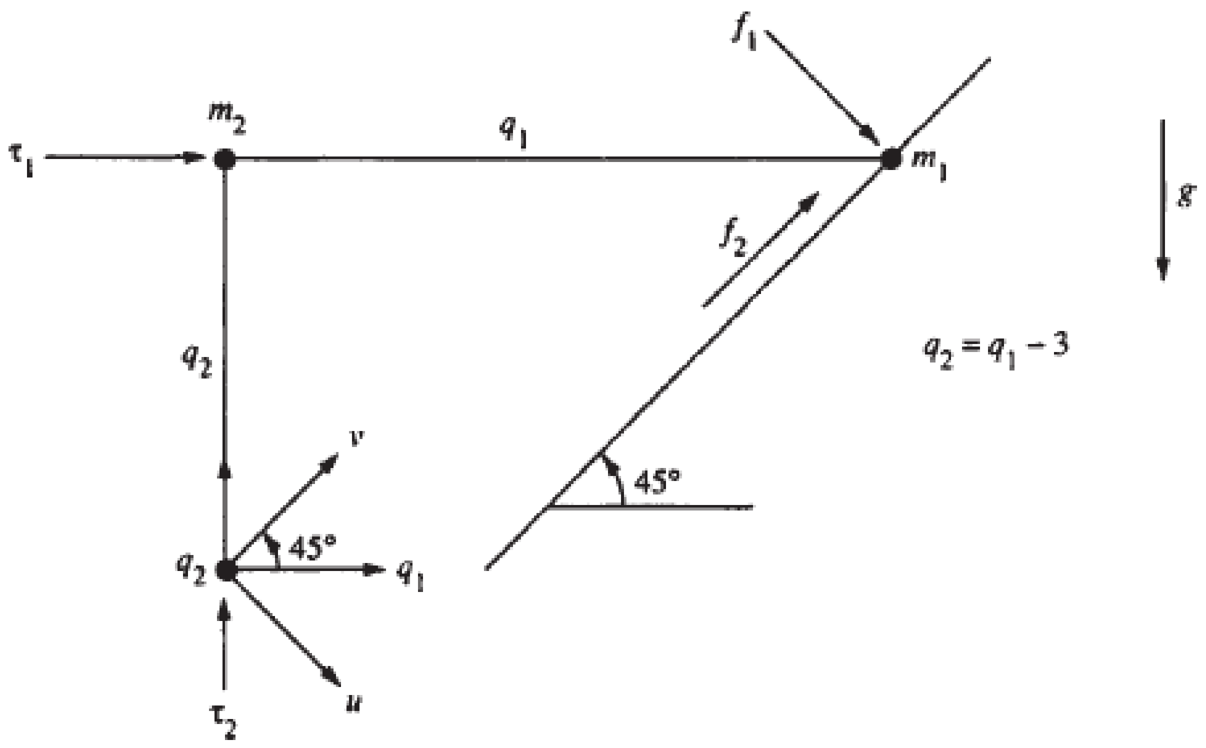

Figure 9.2.5: Manipulator moving along slanted surface.

EXAMPLE 9.2–1: Task Space Formulation for a Slanted Surface

We want to find the manipulator dynamics for the Cartesian manipulator system (i.e., both joints are prismatic) given in Figure 9.2.5 and to decompose the forces exerted on the surface into a normal force and a tangent force. First, the motion portion of the dynamics can easily be determined when the robot is not constrained by the surface. After removing the surface and the interaction forces f_1, and f_2, the manipulator dynamics can be shown to be

$$\tau = M\ddot{q} + G + F(\dot{q}), \tag{1}$$

where

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \; q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \; G = \begin{bmatrix} 0 \\ (m_1 + m_2)\, g \end{bmatrix},$$

$$M = \begin{bmatrix} m_1 & 0 \\ 0 & m_1 + m_2 \end{bmatrix},$$

To account for the interaction forces, let *x* be the 2×1 task space vector defined by

$$x = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2}$$

where *u* and *v* define a fixed coordinate system such that *u* represents the normal distance to the surface, and *v* represents the tangent distance along the surface. As in (9.2.9), the task space coordinates can be expressed in terms of the joint space coordinates by

$$x = h(q), \tag{3}$$

where *h(q)* is found from the geometry of the problem to be

$$h(q) = \frac{1}{\sqrt{2}} \begin{bmatrix} q_1 - q_2 \\ q_1 + q_2 \end{bmatrix}. \tag{4}$$

The task space Jacobian matrix is found from (9.2.11) by utilizing the fact that *T* is the identity matrix for this problem because we do not have to concern ourselves with any end-effector angles of orientation. That is, *J(q)* is given as

$$J = \frac{\partial h(q)}{\partial q} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}. \tag{5}$$

Following (9.2.14), the robot manipulator equation is given by

$$\tau = M\ddot{q} + G + F(\dot{q}) + J^T f, \tag{6}$$

where

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$

It is important to realize that the normal force (i.e., f_1) and the tangent force (i.e., f_2) are drawn in the direction of the task space coordinate system given by (2) (see Fig. 9.2.5).

EXAMPLE 9.2–2: Task Space Formulation for an Elliptical Surface

We wish to find the manipulator dynamics for the Cartesian manipulator system given in Figure 9.2.6 and to decompose the forces exerted on the surface into a normal force and a tangent force. The motion portion of the dynamics is the same as in Example 9.2.1; however, due to the change in the environmental surface, a new task space coordinate system must be defined. Specifically, let x be the 2×1 task space vector defined by

$$x = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{1}$$

where u and v define a rotating coordinate system such that u represents the normal distance to the surface and v represents the tangent distance along the surface. As in (9.2.9), the task space coordinates can be expressed in terms of the joint space coordinates by

$$x = h(q), \tag{2}$$

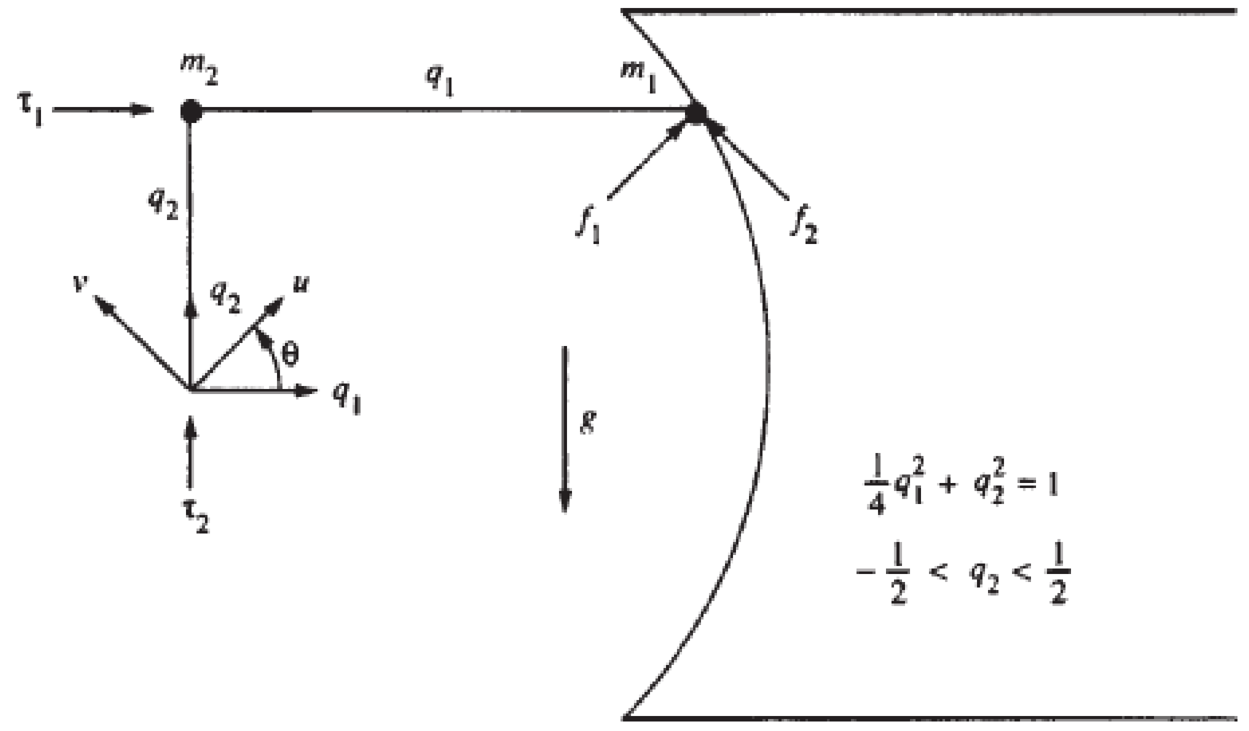

Figure 9.2.6: Manipulator moving along elliptical surface.

where *h(q)* is found to be

$$h(q) = \begin{bmatrix} \bar{u}\cdot\bar{q}_1 & \bar{u}\cdot\bar{q}_2 \\ \bar{v}\cdot\bar{q}_1 & \bar{v}\cdot\bar{q}_2 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \tag{3}$$

with *u*, *v*, q_1 and q_2 being appropriately defined unit vectors used in the dot product notation given in (3). The unit vectors q_1 and q_2 are defined in terms of the fixed coordinate set given by q_1 and q_2. These unit vectors are defined as

$$\bar{q}_1 = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \bar{q}_2 = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4}$$

To find the unit vectors *u* and *v*, we first use the function of the surface

$${}^1\!/\!_4 q_1^2 + q_2^2 = 1 \tag{5}$$

to parameterize the surface in terms of one variable (i.e., q_2) as follows:

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 2\sqrt{1-q_2^2} \\ q_2 \end{bmatrix}. \tag{6}$$

The partial derivative of (6) with respect to q_2 divided by the length of the vector yields a unit vector (i.e., ν) that is always tangent to the surface. That is, ν is given by

$$\bar{\nu} = \begin{bmatrix} \frac{\partial q_1}{\partial q_2} \\ \frac{\partial q_2}{\partial q_2} \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} -2q_2 \left[1 - q_2^2\right]^{-1/2} \\ 1 \end{bmatrix}, \tag{7}$$

where

$$\Delta = \sqrt{1 + 4q_2^2 \left(1 - q_2^2\right)^{-1}}.$$

By using (5) again, the expression for ν can be simplified to yield

$$\bar{\nu} = \frac{1}{\Delta} \begin{bmatrix} -4q_2/q_1 \\ 1 \end{bmatrix}, \tag{8}$$

where

$$\Delta = \sqrt{1 + 16q_2^2/q_1^2}.$$

Because the vectors *u* and *v* must be orthogonal (i.e., *u.v*=0), via (8) and the geometry of the problem, we know that

$$\bar{u} = \frac{1}{\Delta}\begin{bmatrix} 1 \\ 4q_2/q_1 \end{bmatrix}. \tag{9}$$

Substituting (4), (8), and (9) into (3) yields

$$h(q) = \frac{1}{\Delta}\begin{bmatrix} q_1 + 4q_2^2/q_1 \\ -3q_2 \end{bmatrix}. \tag{10}$$

The task space Jacobian matrix is found from (9.2.11) by utilizing the fact that *T* is the identity matrix. That is, *J(q)* is given as

$$J(q) = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \tag{11}$$

$$
\begin{aligned}
J_{11} &= {}^1\!/_2 q_1 \left[{}^1\!/_4 q_1^2 + 7q_2^2\right] \left[{}^1\!/_4 q_1^2 + 4q_2^2\right]^{-3/2}, \\
J_{12} &= q_2 \left[8q_2^2 - q_1^2\right] \left[{}^1\!/_4 q_1^2 + 4q_2^2\right]^{-3/2}, \\
J_{21} &= -6q_2^3 \left[{}^1\!/_4 q_1^2 + 4q_2^2\right]^{-3/2}, \\
J_{22} &= -{}^3\!/_8 q_1^3 \left[{}^1\!/_4 q_1^2 + 4q_2^2\right]^{-3/2},
\end{aligned}
$$

Following (9.2.14), the robot manipulator equation is given by

$$\tau = M\ddot{q} + G + F\left(\dot{q}\right) + J^T\left(q\right) f, \tag{12}$$

where τ, M, q, G, f, and $F(q)$ are as defined in Example 9.2.1. It is important to note that the normal force (i.e., f_1) and the tangent force (i.e., f_2) are drawn in the direction of the task space coordinate system given by (1) (see Fig. 9.2.6).

Torque Controller:

$$\tau = J^T(q)\left(-K_v \dot{x} + K_p \tilde{x}\right) + G(q) + F(\dot{q})$$

where $J(q)$ is the task space Jacobian.

Stability:

Nonconstrained directions: set-point positional control

$$\lim_{t \to \infty} x_i(t) = x_{di}$$

Constrained directions: steady-state force control approximated by

$$\lim_{t \to \infty} f_i(t) \cong K_{pi}\left(x_{di} - x_{ei}\right)$$

Comments: Control gains K_{pi} are used to adjust stiffness of the manipulator.

EXAMPLE 9.2–3: Stiffness Controller for a Cartesian Manipulator

We want to design and simulate a stiffness controller for the robot manipulator system given in Figure 9.2.5. The control objective is to move the end effector to a desired final position of ν_d=3 m while exerting a final desired normal force of f_{d1}=2 N. We neglect the surface friction (i.e., f_2) and joint friction, and assume that the normal force (i.e., f_1) satisfies the relationship

$$f_1 = k_e\left(u - u_e\right), \tag{1}$$

where $u_e = 3/\sqrt{2}$ m and k_e=1000 N/m. The robot link masses are assumed to be unity, and the initial end-effector position is given by

$$\nu\left(0\right) = 5\text{m} \quad \text{and} \quad u\left(0\right) = 3\Big/\sqrt{2}\,\text{m}. \tag{2}$$

To accomplish the control objective, the stiffness controller from Table 9.2.1 is given by

$$\tau = J^T\left(q\right)\left(-K_v\dot{x} + K_p\tilde{x}\right) + G\left(q\right), \tag{3}$$

where

$$\tilde{x} = \begin{bmatrix} u_d - u \\ \nu_d - \nu \end{bmatrix},$$

τ, J, G, and x are as defined in Example 9.2.1, u_d is defined as the desired normal position, and the gain matrices K_v and K_p have been taken to be K_v=k_vI and K_p=k_pI. For this example we select k_v= k_p=10, which will guarantee that $k_p \ll k_e$ as required in the stiffness control formulation. To satisfy the control objective that f_{d1}=2 N, we utilize (9.2.27) to determine the desired normal position. Specifically, substituting the values of f_{d1}, k_e, and u_e into

$$f_{d1} = k_p \left(u_d - u_e\right) \tag{4}$$

yields $u_d = (0.2 + 3/\sqrt{2})$ m.

The simulation of the stiffness controller given by (3) for the robot manipulator system (Figure 9.2.5) is given in Figure 9.2.7. As indicated by the simulation, the desired tangential position and normal force are reached in about 4 s.

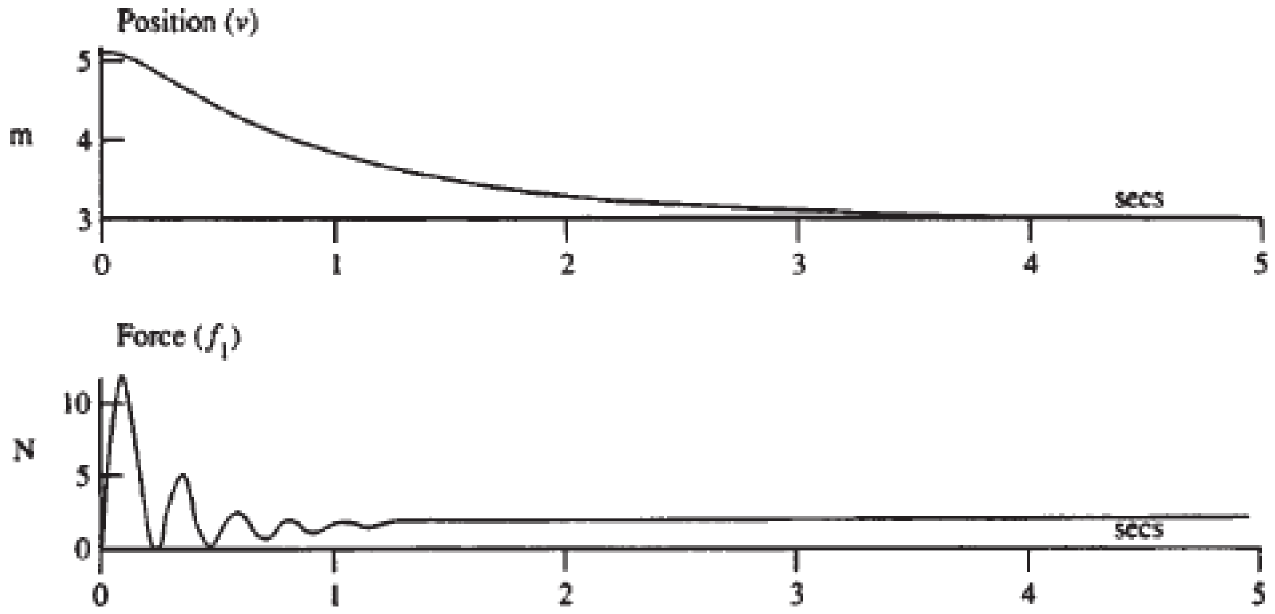

Figure 9.2.7: Simulation of stiffness controller.

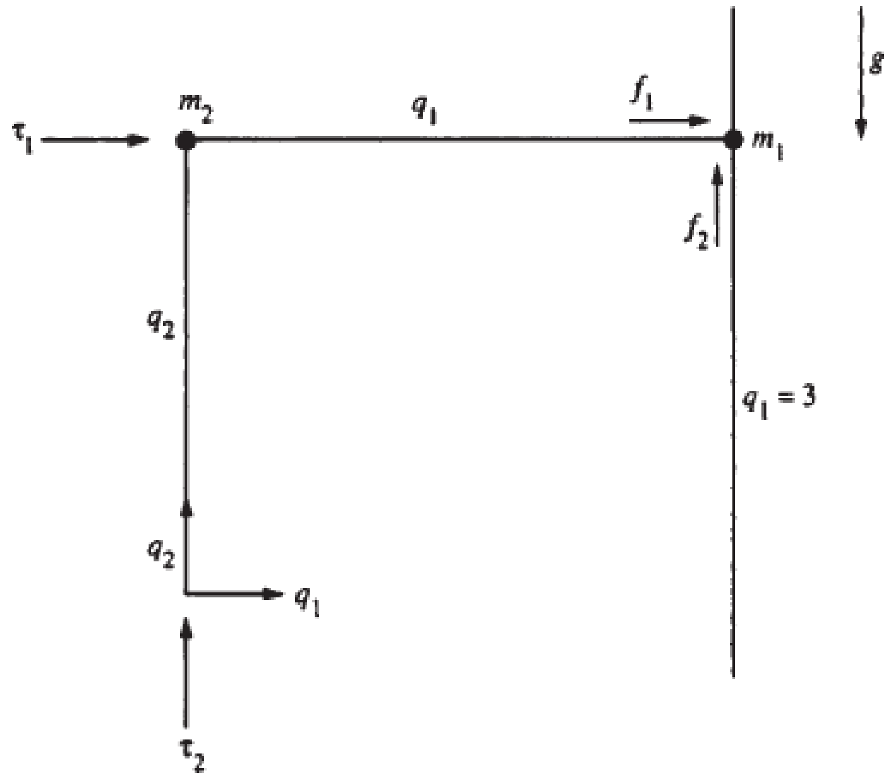

Figure 9.3.1: Manipulator moving along perpendicular surface.

$$x = \begin{bmatrix} u \\ \nu \end{bmatrix} = h(q) = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \qquad (9.3.1)$$

with the task-space Jacobian matrix given by

$$J = \frac{\partial h(q)}{\partial q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

To design the position/force controller for the manipulator system, we must first determine the dynamic equations for the task space formulation given by (9.3.1). Using this task space formulation and neglecting joint friction, the manipulator dynamics can be shown to be

$$\tau = M\ddot{q} + G + f, \qquad (9.3.2)$$

where τ, M, G, and f are as defined in Example 9.2.1. The two dynamic equations given in the matrix form represented by (9.3.2) are

$$\tau_1 = m_1\ddot{q}_1 + f_1 \qquad (9.3.3)$$

and

$$\tau_2 = (m_1 + m_2)\,\ddot{q}_2 + (m_1 + m_2)\,g + f_2. \qquad (9.3.4)$$

In formulating a hybrid position/force controller, we design separate controllers for the dynamics given by (9.3.3) and (9.3.4). As illustrated by Figure 9.3.1, the position along the task space direction q_2 should be position controlled; therefore, we should use (9.3.4) for designing the position controller. [This is obvious because the dynamics given by (9.3.3) do not contain the task space variable q2.]

Because we are designing a position controller to track a desired trajectory, we will define the "tangent space" tracking error to be

$$\tilde{x} = q_{d2} - q_2, \tag{9.3.5}$$

where q_{d2} represents the desired position trajectory along or tangent to the surface. The position controller will be the computed-torque controller (see Chapter 3)

$$\tau_2 = (m_1 + m_2)\, a_T + (m_1 + m_2)\, g + f_2, \tag{9.3.6}$$

where

$$a_T = \ddot{q}_{d2} + k_{Tv}\dot{\tilde{x}} + k_{Tp}\tilde{x}, \tag{9.3.7}$$

with k_{Tv} and k_{Tp} being positive control gains. Substituting (9.3.6) into (9.3.4) gives the position tracking error system

$$\ddot{\tilde{x}} + k_{Tv}\dot{\tilde{x}} + k_{Tp}\tilde{x} = 0 \tag{9.3.8}$$

By using the fact that k_{Tv} and k_{Tp} are positive, we can apply standard linear control results to (9.3.8) to yield

$$\lim_{t\to\infty} \tilde{x} = 0;$$

Specifically, the normal force f_1 exerted on the environment is given by

$$f_1 = k_e\left(q_1 - q_e\right), \tag{9.3.9}$$

where k_e represents the environment stiffness and q_e=3. Taking the second derivative of (9.3.9) with respect to time gives the expression

$$\ddot{q}_1 = \frac{1}{k_e}\ddot{f}_1, \tag{9.3.10}$$

where the normal task space acceleration is written in terms of the second derivative of the normal force. Substituting (9.3.10) into (9.3.3) yields the force dynamic equation

$$\tau_1 = \frac{m_1}{k_e}\ddot{f}_1 + f_1. \tag{9.3.11}$$

$$\tilde{f} = f_{d1} - f_1, \tag{9.3.12}$$

where f_{d1} represents the desired normal force that is to be exerted on the environment. Similar to the position controller, the force controller will be the computed-torque controller

$$\tau_1 = \frac{m_1}{k_e}a_N + f_1. \tag{9.3.13}$$

.

$$a_N = \ddot{f}_{d1} + k_{Nv}\dot{\tilde{f}} + k_{Np}\tilde{f}, \tag{9.3.14}$$

with k_{Nv} and k_{Np} being positive control gains. Substituting (9.3.13) into (9.3.11) gives the force tracking error system,

$$\ddot{\tilde{f}} + k_{Nv}\dot{\tilde{f}} + k_{Np}\tilde{f} = 0. \tag{9.3.15}$$

Using the fact k_{Nv} and k_{Np} are positive in (9.3.15) yields

$$\lim_{t\to\infty} \tilde{f} = 0;$$

therefore, asymptotic force tracking is guaranteed with the controller given by (9.3.13). It is important to realize that the force controller requires measurement of the normal force and the derivative of the normal force. Because the force derivative is often not available for measurement, it is manufactured from (9.3.9), that is,

$$\dot{f}_1 = k_e\dot{q}_1; \tag{9.3.16}$$

therefore, the stiffness of the environment and the normal task space velocity are used to simulate the derivative of the force.

EXAMPLE 9.3–1: Hybrid Position/Force Control Along a Slanted Surface

We want to design and simulate a hybrid position/force controller for the robot manipulator system given in Figure 9.2.5. The control objective is to move the end effector with a desired surface trajectory of $v_d=\sin(t)$ m while exerting a normal force trajectory of $f_{d1} = 1 - e^{-t}N$. We neglect joint friction and assume that the normal force (i.e., f_1) satisfies the relationship

$$f_1 = k_e\left(u - u_e\right), \tag{1}$$

where $u_e = 3/\sqrt{2}$ m and k_e=1000 N/m. The robot link masses are assumed to be unity, and the initial end-effector position is given by

$$v(0) = 0\ \text{m} \quad \text{and} \quad u(0) = 3\Big/\sqrt{2}\ \text{m}. \tag{2}$$

To accomplish the control objective, the hybrid position/force controller from Table 9.3.1 is given by

$$\tau = MJ^{-1}\text{a} + G + J^T f, \tag{3}$$

where a is a 2×1 vector representing the linear position and force controllers with τ, *J*, *G*, and *f* as defined in Example 9.2.1. The controller given by (3) decouples the robot dynamics in the task space as follows:

$$\ddot{x} = \begin{bmatrix} \ddot{u} \\ \ddot{v} \end{bmatrix} = \mathbf{a}. \tag{4}$$

From Figure 9.2.5, we can see that the task space variable u represents the normal space, and the task space variable v represents the tangent space; therefore, (4) may rewritten in the notation given in Table 9.3.1 as

$$\ddot{x} = \begin{bmatrix} \ddot{u} \\ \ddot{v} \end{bmatrix} = \begin{bmatrix} \ddot{x}_{N1} \\ \ddot{x}_{T1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{N1} \\ \mathbf{a}_{T1} \end{bmatrix} = \mathbf{a}. \tag{5}$$

From Table 9.3.1, the corresponding linear position and force controllers are then given by

$$\mathbf{a}_{T1} = \ddot{x}_{Td1} + k_{Tv1}\dot{\tilde{x}}_{T1} + k_{Tp1}\tilde{x}_{T1} \tag{6}$$

and

$$\mathrm{a}_{N1} = \frac{1}{k_{e1}}\left(\ddot{f}_{Nd1} + k_{Nv1}\dot{\tilde{f}}_{N1} + k_{Np1}\tilde{f}_{N1}\right) \tag{7}$$

where $x_{Tdl} = \sin t$, $f_{Nd1} = 1 - e^{-t}$, and k_{e1}=1000.

The simulation of the hybrid position/force controller given by (3), (6), and (7) for the robotic manipulator system (Figure 9.2.5) is given in Figure 9.3.2. The controller gains were selected as

$$k_{Nv1} = k_{Np1} = k_{Tv1} = k_{Tp1} = 10.$$

As indicated by the simulation, the position and force tracking error go to zero in about 4 s.

—

Table 9.3.1: Hybrid Position/Force Controller

Torque Controller:

$$\tau = M(q)\,J^{-1}(q)\left(\mathrm{a} - \dot{J}(q)\,\dot{q}\right) + V_m(q,\dot{q})\,\dot{q} + G(q) + F(\dot{q}) + J^T(q)\,f$$

where $J(q)$ is the task space Jacobian.

Position control:

$$\mathrm{a}_{Ti} = \ddot{x}_{Tdi} + k_{Tvi}\dot{\tilde{x}}_{Ti} + k_{Tpi}\tilde{x}_{Ti}$$

Force control:

$$\mathrm{a}_{Nj} = \tfrac{1}{k_{ej}}\left(\ddot{f}_{Ndj} + k_{Nvj}\dot{\tilde{f}}_{Nj} + k_{Npj}\tilde{f}_{Nj}\right)$$

Stability:

Nonconstrained directions: position tracking control

$$\lim_{t\to\infty} x_{Ti}(t) = x_{Tdi}(t)$$

Constrained directions: force tracking control

$$\lim_{t\to\infty} f_{Nj}(t) = f_{Ndj}(t)$$

Comments: Environment is modeled as a spring.