

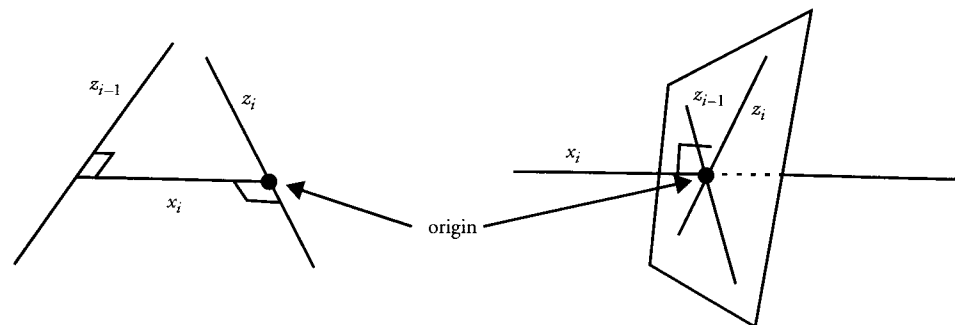
The following procedure can be used to construct the frames for intermediate joints.

1. For each joint, identify the axis of rotation for revolute joints and the axis of displacement for prismatic joints. Refer to this axis as the z -axis of the joint's frame.
2. For each adjacent pair of joints, the i th - 1 and i th for i from 1 to n , construct the common perpendicular between the z -axes or, if they intersect, the perpendicular to the plane that contains them. Refer to the intersection of the perpendicular and the i th frame's z -axis (or the point of intersection of the two axes) as the origin of the i th frame. Refer to the perpendicular as the x -axis of the i th frame. See Figure 4.16.
3. Construct the y -axis of each frame to be consistent with the right-hand rule (assuming right-hand space).

4.2.4 Inverse Kinematics

In inverse kinematics, the desired position and possibly orientation of the end effector are given by the user, and the joint angles required to attain that configuration are calculated. The problem can have zero, one, or more solutions. If there are so many constraints on the configuration that no solution exists, the system is called *overconstrained*. If there are relatively few constraints on the system and there are many solutions to the problem posed, then it is *underconstrained*. The *reachable workspace* is that volume which the end effector can reach. The *dextrous workspace* is the volume that the end effector can reach in any orientation.

If the mechanism is simple enough, then the joint angles (the pose vector) required to produce the final, desired configuration can be calculated analytically. Given an initial pose vector and the final pose vector, intermediate configurations can be formed by interpolation of the values in the pose vectors, thus animating the mechanism from its initial configuration to the final one. However, if the



mechanism is too complicated for analytic solutions, then an incremental approach can be used that employs a matrix of values (the *Jacobian*) that relates changes in the joint angles to changes in the end effector position and orientation. The end effector is iteratively nudged until the final configuration is attained within a given tolerance.

Solving a Simple System by Analysis

For sufficiently simple mechanisms, the joint angles of a final desired position can be determined analytically by inspecting the geometry of the linkage. Consider a simple two-link arm in two-dimensional space. Link lengths are L_1 and L_2 for the first and second link respectively. If a position is fixed for the base of the arm at the first joint, any position beyond $|L_1 - L_2|$ units from the base of the link and within $L_1 + L_2$ of the base can be reached. See Figure 4.17.

Assume for now (without loss of generality) that the base is at the origin. In a simple inverse kinematics problem, the user gives the (X, Y) coordinate of the desired position for the end effector. The joint angles, θ_1 and θ_2 , can be solved for by computing the distance from the base to the goal and using the law of cosines to compute the interior angles. Once the interior angles are computed, the rotation angles for the two links can be computed. See Figure 4.18. Of course, the first step is to make sure that the position of the goal is within the reach of the end effector; that is, $L_1 - L_2 \leq \sqrt{X^2 + Y^2} \leq L_1 + L_2$.

In this simple scenario, there are only two solutions that will give the correct answer; the configurations are symmetric with respect to the line from $(0, 0)$ to (X, Y) . This is reflected in the equation in Figure 4.18 because the arccosine is

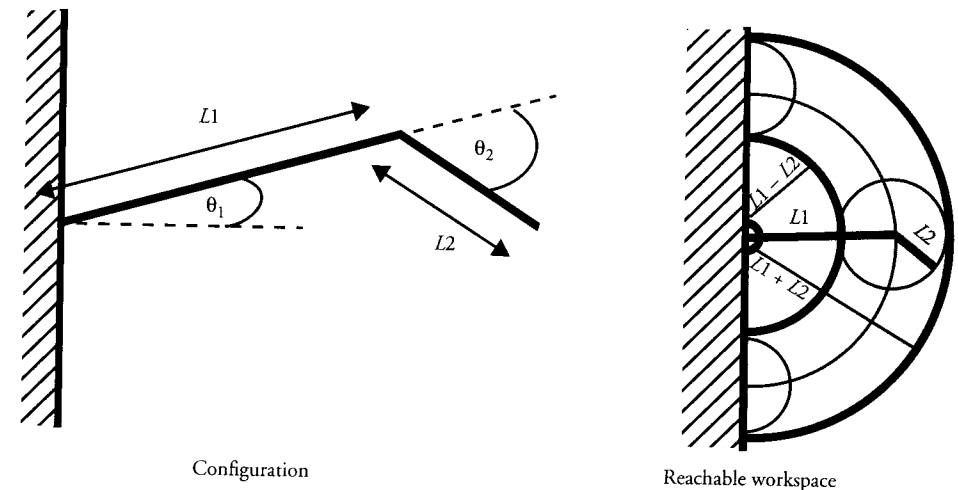
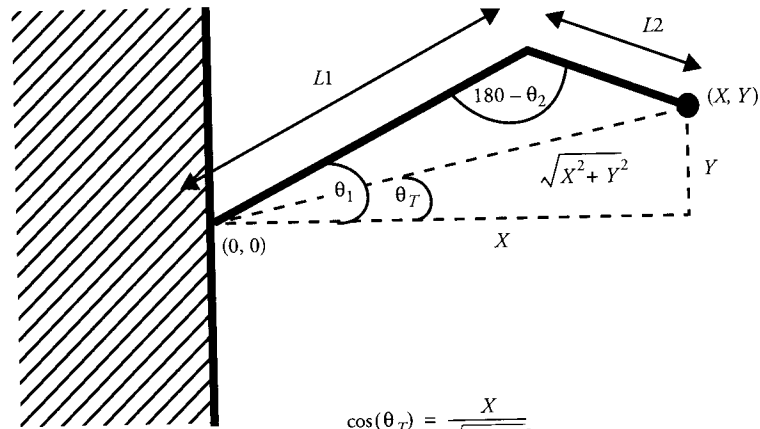


Figure 4.17 Simple linkage



$$\cos(\theta_T) = \frac{X}{\sqrt{X^2 + Y^2}}$$

$$\theta_T = \arccos\left(\frac{X}{\sqrt{X^2 + Y^2}}\right)$$

$$\cos(\theta_1 - \theta_T) = \frac{L1^2 + X^2 + Y^2 - L2^2}{2 \cdot L1 \cdot \sqrt{X^2 + Y^2}} \quad (\text{cosine rule})$$

$$\theta_1 = \arccos\left(\frac{L1^2 + X^2 + Y^2 - L2^2}{2 \cdot L1 \cdot \sqrt{X^2 + Y^2}}\right) + \theta_T$$

$$\cos(180 - \theta_2) = \frac{L1^2 + L2^2 - (X^2 + Y^2)}{2 \cdot L1 \cdot L2} \quad (\text{cosine rule})$$

$$\theta_2 = \arccos\left(\frac{L1^2 + L2^2 - (X^2 + Y^2)}{2 \cdot L1 \cdot L2}\right)$$

Figure 4.18 Equations used in solving simple inverse kinematic problem

two-valued in both plus and minus theta (θ). However, for more complicated armatures, there may be infinitely many solutions that will give the desired end effector location.

The joint angles for relatively simple linkages can be solved by algebraic manipulation of the equations that describe the relationship of the end effector to the base frame. Most linkages used in robotic applications are designed to be simple enough for this analysis. However, for many cases that arise in computer animation, analytic solutions are not tractable. In such cases, iterative numeric solutions must be relied on.

The Jacobian

Most mechanisms of interest to computer animation are too complex to allow an analytic solution. For these, the motion can be incrementally constructed. At each

time step, a computation is performed that determines the best way to change each joint angle in order to direct the current position and orientation of the end effector toward the desired configuration. The computation forms the matrix of partial derivatives called the *Jacobian*.

To explain the Jacobian from a strictly mathematical point of view, consider the six arbitrary functions of Equation 4.7, each of which is a function of six independent variables. Given specific values for the input variables, x_i , each of the output variables, y_i , can be computed by its respective function.

$$y_1 = f_1(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_2 = f_2(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_3 = f_3(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_4 = f_4(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_5 = f_5(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_6 = f_6(x_1, x_2, x_3, x_4, x_5, x_6)$$

(Eq. 4.7)

These equations can also be used to describe the change in the output variables relative to the change in the input variables. The differentials of y_i can be written in terms of the differentials of x_i using the chain rule. This generates Equation 4.8. Equation 4.7 and Equation 4.8 can be put in vector notation, producing Equation 4.9 and Equation 4.10 respectively.

$$\begin{aligned} \delta y_i = & \frac{\delta f_i}{\delta x_1} \cdot \delta x_1 + \frac{\delta f_i}{\delta x_2} \cdot \delta x_2 + \frac{\delta f_i}{\delta x_3} \cdot \delta x_3 + \frac{\delta f_i}{\delta x_4} \cdot \delta x_4 \\ & + \frac{\delta f_i}{\delta x_5} \cdot \delta x_5 + \frac{\delta f_i}{\delta x_6} \cdot \delta x_6 \end{aligned}$$

(Eq. 4.8)

$$Y = F(X)$$

(Eq. 4.9)

$$\delta Y = \frac{\partial F}{\partial X} \cdot \delta X$$

(Eq. 4.10)

The 6x6 matrix of partial derivatives, $\partial F/\partial X$, is called the *Jacobian* and is a function of the current values of x_i . The Jacobian can be thought of as mapping the velocities of X to the velocities of Y (Equation 4.11). At any point in time, the Jacobian is a linear function of x_i . At the next instant of time, X has changed and so has the linear transformation represented by the Jacobian.

$$\dot{Y} = J(X) \cdot \dot{X}$$

(Eq. 4.11)

When one applies the Jacobian to a linked appendage, the input variables, x_i , become the joint angles and the output variables, y_i , become the end effector position and orientation. In this case, the Jacobian relates the velocities of the joint angles to the velocities of the end effector position and orientation (Equation 4.12).

$$V = J(\theta)\dot{\theta} \quad (\text{Eq. 4.12})$$

V is the vector of linear and rotational velocities and represents the desired change in the end effector. The desired change will be based on the difference between its current position/orientation to that specified by the goal configuration. These velocities are vectors in three-space, so each has an x , y , and z component (Equation 4.13). $\dot{\theta}$ is a vector of joint angle velocities which are the unknowns of the equation (Equation 4.14). J , the Jacobian, is a matrix that relates the two and is a function of the current pose (Equation 4.15).

$$V = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T \quad (\text{Eq. 4.13})$$

$$\dot{\theta} = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dots, \dot{\theta}_n]^T \quad (\text{Eq. 4.14})$$

$$J = \begin{bmatrix} \frac{\partial v_x}{\partial \theta_1} & \frac{\partial v_x}{\partial \theta_2} & \dots & \frac{\partial v_x}{\partial \theta_n} \\ \frac{\partial v_y}{\partial \theta_1} & \frac{\partial v_y}{\partial \theta_2} & \dots & \frac{\partial v_y}{\partial \theta_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial \omega_x}{\partial \theta_1} & \frac{\partial \omega_x}{\partial \theta_2} & \dots & \frac{\partial \omega_x}{\partial \theta_n} \end{bmatrix} \quad (\text{Eq. 4.15})$$

Each term of the Jacobian relates the change of a specific joint to a specific change in the end effector. The rotational change in the end effector, ω , is merely the velocity of the joint angle about the axis of revolution at the joint under consideration. The linear change in the end effector is the cross product of the axis of revolution and a vector from the joint to the end effector. The rotation at the joint induces an instantaneous linear direction of travel at the end effector. See Figure 4.19.

The desired angular and linear velocities are computed by finding the difference between the current configuration of the end effector and the desired configuration. The angular and linear velocities of the end effector induced by the rotation of a specific joint axis are determined by the computations shown in Figure 4.19.

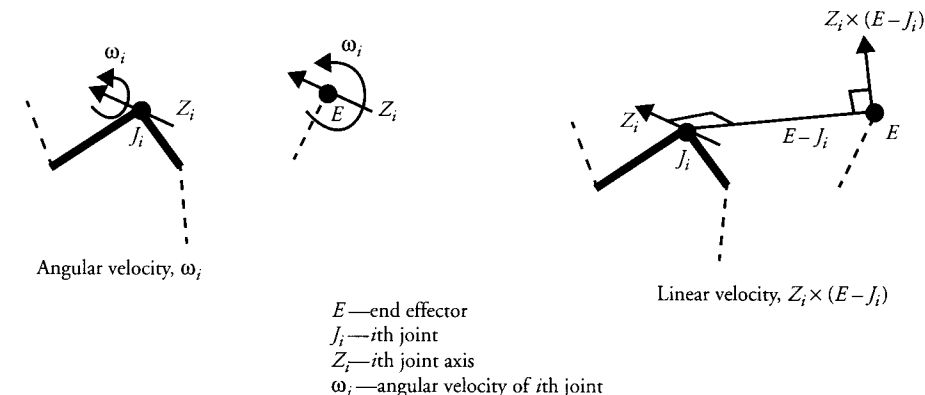


Figure 4.19 Angular and linear velocities induced by joint axis rotation

The problem is to determine the best linear combination of velocities induced by the various joints that would result in the desired velocities of the end effector. The Jacobian is formed by posing the problem in matrix form.

When one assembles the Jacobian, it is important to make sure that all of the coordinate values are in the same coordinate system. It is often the case that joint-specific information is given in the coordinate system local to that joint. In forming the Jacobian matrix, this information must be converted into some common coordinate system such as the global inertial coordinate system or the end effector coordinate system. Various methods have been developed for computing the Jacobian based on attaining maximum computational efficiency given the required information in local coordinate systems, but all methods produce the derivative matrix in a common coordinate system.

A Simple Example

Consider the simple three-revolute-joint, planar manipulator of Figure 4.20. In this example, the objective is to move the end effector, E , to the goal position, G . The orientation of the end effector is of no concern in this example. The axis of rotation of each joint is perpendicular to the figure, coming out of the paper. The effect of an incremental rotation, g_i , of each joint can be determined by the cross product of the joint axis and the vector from the joint to the end effector, V_i (Figure 4.21). Notice that the magnitude of each g_i is a function of the distance between the locations of the joint and the end effector.

The desired change to the end effector is the difference between the current position of the end effector and the goal position. A vector of the desired change in values is set equal to the Jacobian matrix multiplied by a vector of the unknown values, which are the changes to the joint angles (Equation 4.16).

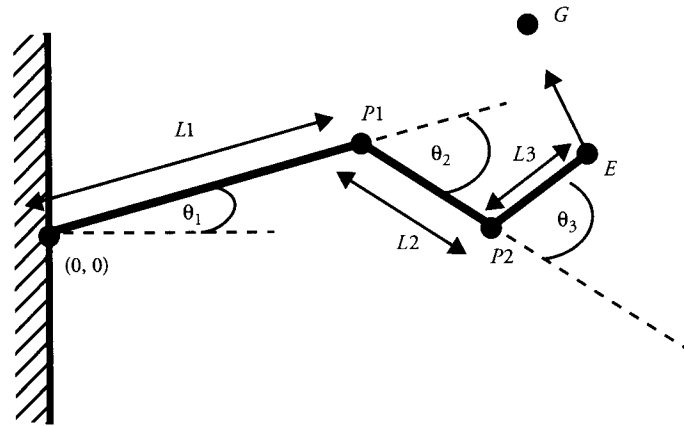


Figure 4.20 Planar, three-joint manipulator

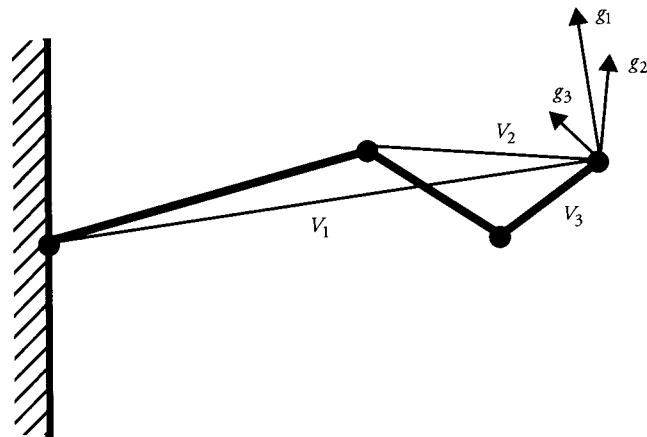


Figure 4.21 Instantaneous changes in position induced by joint angle rotations

$$\begin{bmatrix} (G-E)_x \\ (G-E)_y \\ (G-E)_z \end{bmatrix} = \begin{bmatrix} ((0,0,1) \times E)_x & (0,0,1) \times (E-P_1)_x & (0,0,1) \times (E-P_2)_x \\ ((0,0,1) \times E)_y & (0,0,1) \times (E-P_1)_y & (0,0,1) \times (E-P_2)_y \\ ((0,0,1) \times E)_z & (0,0,1) \times (E-P_1)_z & (0,0,1) \times (E-P_2)_z \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{Eq. 4.16}$$

Solution Using the Inverse Jacobian

Once the Jacobian has been computed, an equation in the form of Equation 4.17 is to be solved. In the case that J is a square matrix, the inverse of the Jacobian, J^{-1} , is used to compute the joint angle velocities given the end effector velocities (Equation 4.18).

$$V = J\dot{\theta} \tag{Eq. 4.17}$$

$$J^{-1}V = \dot{\theta} \tag{Eq. 4.18}$$

If the inverse of the Jacobian (J^{-1}) does not exist, then the system is said to be singular for the given joint angles. A singularity occurs when a linear combination of the joint angle velocities cannot be formed to produce the desired end effector velocities. As a simple example of such a situation, consider a fully extended, planar arm with a goal position somewhere on the forearm (see Figure 4.22). In such a case, a change in each joint angle would produce a vector perpendicular to the desired direction. Obviously, no linear combination of these vectors could produce the desired motion vector. Unfortunately, all of the singularities of a system cannot be determined simply by visually inspecting the possible geometric configurations of the linkage.

Problems with singularities can be reduced if the manipulator is redundant—when there are more degrees of freedom than there are constraints to be satisfied.

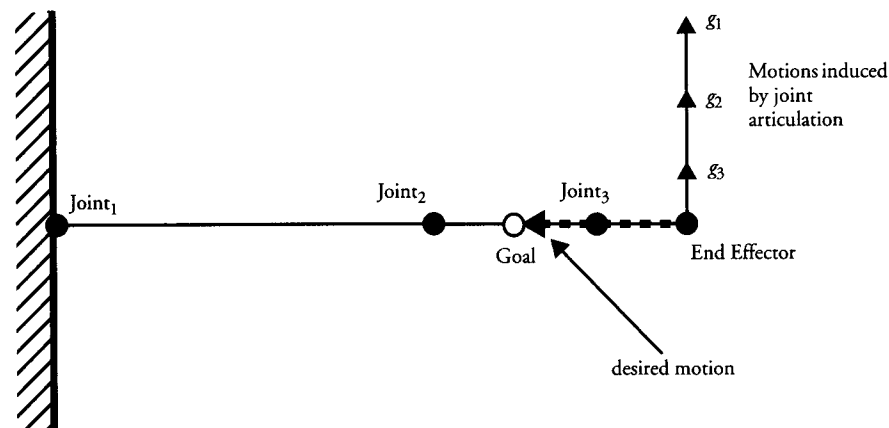


Figure 4.22 Simple example of a singular configuration

In this case, the Jacobian is not a square matrix and there are an infinite number of solutions to the inverse kinematics problem. Because the Jacobian is not square, a conventional inverse does not exist. Instead, the *pseudo inverse*, J^+ , can be used (Equation 4.19). Equation 4.19 works because a matrix multiplied by its own transpose will be a square matrix.

$$\begin{aligned} V &= J\dot{\theta} \\ J^T V &= J^T J \dot{\theta} \\ (J^T J)^{-1} J^T V &= (J^T J)^{-1} J^T J \dot{\theta} \\ J^+ V &= \dot{\theta} \end{aligned} \quad (\text{Eq. 4.19})$$

$J^+ = (J^T J)^{-1} J^T = J^T (J J^T)^{-1}$ is called the *pseudo inverse* of J . It maps the desired velocities of the end effector to the required velocities of the joint angles. After making the substitutions shown in Equation 4.20, LU decomposition can be used to solve Equation 4.21 for β . This can then be substituted into Equation 4.22 to solve for $\dot{\theta}$.

$$\begin{aligned} J^+ V &= \dot{\theta} \\ J^T (J J^T)^{-1} V &= \dot{\theta} \\ \beta &= (J J^T)^{-1} V \end{aligned} \quad (\text{Eq. 4.20})$$

$$(J J^T) \beta = V \quad (\text{Eq. 4.21})$$

$$J^T \beta = \dot{\theta} \quad (\text{Eq. 4.22})$$

It is important to remember that the Jacobian is only valid for the instantaneous configuration for which it is formed. That is, as soon as the configuration of the linkage changes, the Jacobian ceases to accurately describe the relationship between changes in joint angles and changes in end effector position and orientation. This means that if too big a step is taken in joint angle space, the end effector may not appear to travel in the direction of the goal. If this appears to happen during an animation sequence, then taking smaller steps in joint angle space and thus recalculating the Jacobian more often may be in order.

Adding More Control

The pseudo inverse computes one of many possible solutions. It minimizes joint angle rates. The configurations produced, however, do not necessarily correspond to what might be considered natural poses. To better control the kinematic model, a control expression can be added to the pseudo inverse Jacobian solution. The control expression is used to solve for control angle rates with certain attributes. The added control expression, because of its form, contributes nothing to the desired end effector motion. The form for the control expression is shown in Equation 4.23. In Equation 4.24 it is shown that this form of the control expression does not add anything to the velocities. As a consequence, the control expression can be combined with the pseudo inverse Jacobian solution so that the given velocities are still satisfied [27].

$$\dot{\theta} = (J^+ J - I) z \quad (\text{Eq. 4.23})$$

$$\begin{aligned} V &= J\dot{\theta} \\ V &= J(J^+ J - I) z \\ V &= (J J^+ J - J) z \\ V &= (J - J) z \\ V &= 0 \cdot z \\ V &= 0 \end{aligned} \quad (\text{Eq. 4.24})$$

To bias the solution toward specific joint angles, such as the middle angle between joint limits, H is defined as in Equation 4.25, where θ_i are the current joint angles, θ_{ci} are the desired joint angles, α_i are the desired angle gains, and ψ is the ψ th norm (for ψ even). Variable z is equal to the gradient of H , ∇H (Equation 4.26). This does not enforce joint limits as hard constraints, but the solution

can be biased toward the middle values so that violating the joint limits is less probable.

$$H = \sum_{i=1}^n \alpha_i \cdot (\theta_i - \theta_{ci})^\psi \quad (\text{Eq. 4.25})$$

$$z = \nabla_\theta H = \frac{dH}{d\theta} = \psi \sum_{i=1}^n \alpha_i \cdot (\theta_i - \theta_{ci})^{\psi-1} \quad (\text{Eq. 4.26})$$

The desired angles and gains are input parameters. The gain indicates the relative importance of the associated desired angle; the higher the gain, the stiffer the joint.² If the gain for a particular joint is high, then the solution will be such that the joint angle quickly approaches the desired joint angle. The control expression is added to the solution indicated by the conventional pseudo inverse of the Jacobian (Equation 4.27). If all gains are zero, then the solution will reduce to the conventional pseudo inverse of the Jacobian. Equation 4.27 can be solved by rearranging terms as shown in Equation 4.28.

$$\dot{\theta} = J^+V + (J^+J - I)\nabla_\theta H \quad (\text{Eq. 4.27})$$

$$\dot{\theta} = J^+V + (J^+J - I)\nabla_\theta H$$

$$\dot{\theta} = J^+V + J^+J\nabla_\theta H - I\nabla_\theta H$$

$$\dot{\theta} = J^+(V + J\nabla_\theta H) - \nabla_\theta H$$

$$\dot{\theta} = J^T(JJ^T)^{-1}(V + J\nabla_\theta H) - \nabla_\theta H$$

$$\dot{\theta} = J^T[(JJ^T)^{-1}(V + J\nabla_\theta H)] - \nabla_\theta H \quad (\text{Eq. 4.28})$$

To solve Equation 4.28, set $\beta = (JJ^T)^{-1}(V + J\nabla_\theta H)$ so that Equation 4.29 results. Use LU decomposition to solve for β in Equation 4.30. Substitute the solution for β in Equation 4.29 to solve for $\dot{\theta}$.

$$\dot{\theta} = J^T\beta - \nabla_\theta H \quad (\text{Eq. 4.29})$$

$$V + J\nabla_\theta H = (JJ^T)\beta \quad (\text{Eq. 4.30})$$

Simple Euler integration can be used at this point to update the joint angles. The Jacobian has changed at the next time step, so the computation must be performed again and another step taken. This process repeats until the end effector reaches the goal configuration within some acceptable (i.e., user-defined) tolerance.

4.2.5 Summary

Hierarchical models are extremely useful for enforcing certain relationships among the elements so that the animator can concentrate on just the degrees of freedom remaining. Forward kinematics gives the animator explicit control over each degree of freedom but can become cumbersome when the animation is trying to attain a specific position or orientation of an element at the end of a hierarchical chain. Inverse kinematics, using the inverse or pseudo inverse of the Jacobian, allows the animator to concentrate only on the conditions at the end of such a chain but might produce undesirable configurations. Additional control expressions can be added to the pseudo inverse Jacobian solution to express a preference for solutions of a certain character. However, these are all kinematic techniques. Often, more realistic motion is desired and physically based simulations are needed. These approaches are discussed below.

4.3 Rigid Body Simulation

A common objective in computer animation is to create realistic-looking motion. A major component of realistic motion is the physically based reaction of rigid bodies to commonly encountered forces such as gravity, viscosity, friction, and those resulting from collisions. Creating realistic motion with key-frame techniques can be a daunting task. However, the equations of motion can be incorporated into an animation system to automatically calculate these reactions. This can eliminate considerable tedium—if the animator is willing to relinquish precise control over the motion of some objects.

In rigid body simulation, various forces to be simulated are modeled in the system. These forces may arise due to relative positioning of objects (e.g., gravity, collisions), object velocity (e.g., viscosity), or the absolute position of objects in user-specified vector fields (e.g., wind). When applied to objects, these forces induce linear and angular accelerations based on the mass of the object (in the linear case) and mass distribution of the object (in the angular case). These accelerations, which are the time derivative of velocities, are integrated over a delta time step to produce changes in object velocities (linear and angular). These velocities, in turn integrated over a delta time step, produce changes in object positions and orienta-

2. *Stiffness* refers to how much something reacts to being perturbed. A stiff spring is a strong spring. A stiff joint, as used