Chapter 6

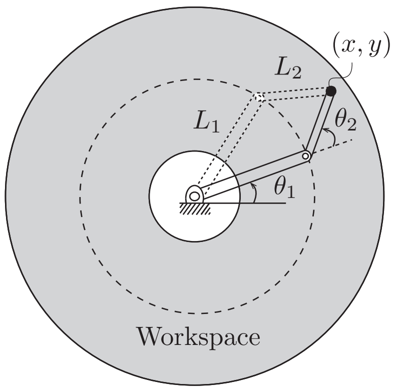
Inverse Kinematics

For a general n degree-of-freedom open chain with forward kinematics $T(\theta)$, $\theta \in \mathbb{R}^n$, the inverse kinematics problem can be stated as follows: given a homogeneous transform $X \in SE(3)$, find solutions θ that satisfy $T(\theta) = X$. To highlight the main features of the inverse kinematics problem, let us examine the two-link planar open chain of Figure 6.1(a) as a motivational example. Considering only the position of the end-effector and ignoring its orientation, the forward kinematics can be expressed as

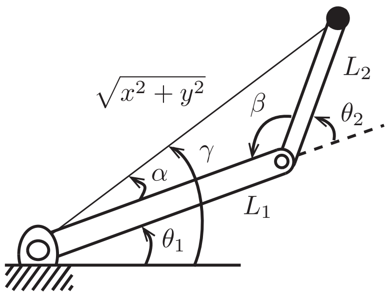
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{bmatrix}. \tag{6.1}$$

Assuming $L_1 > L_2$, the set of reachable points, or the workspace, is an annulus of inner radius $L_1 - L_2$ and outer radius $L_1 + L_2$. Given some end-effector position (x, y), it is not hard to see that there will be either zero, one, or two solutions depending on whether (x, y) lies in the exterior, boundary, or interior of this annulus, respectively. When there are two solutions, the angle at the second joint (the "elbow" joint) may be positive or negative. These two solutions are sometimes called "lefty" and "righty" solutions, or "elbow-up" and "elbow-down" solutions.

Finding an explicit solution (θ_1, θ_2) for a given (x, y) is also not difficult. For this purpose, we will find it useful to introduce the two-argument arctangent function $\operatorname{atan2}(y, x)$, which returns the angle from the origin to a point (x, y) in the plane. It is similar to the inverse tangent $\tan^{-1}(y/x)$, but whereas $\tan^{-1}(y/x)$ is equal to $\tan^{-1}(-y/-x)$, and therefore $\tan^{-1}$ only returns angles in the range $(-\pi/2, \pi/2)$, the atan2 function returns angles in the range $(-\pi, \pi]$. For this reason, atan2 is sometimes called the four-quadrant arctangent.



(a) A workspace, and lefty and righty configurations.



(b) Geometric solution.

Figure 6.1: Inverse kinematics of a 2R planar open chain.

We also recall the law of cosines,

$$c^2 = a^2 + b^2 - 2ab\cos C,$$

where a, b, and c are the lengths of the three sides of a triangle and C is the interior angle of the triangle opposite the side of length c.

Referring to Figure 6.1(b), angle β, restricted to lie in the interval $[0, \pi]$, can be determined from the law of cosines,

$$L_1^2 + L_2^2 - 2L_1L_2\cos\beta = x^2 + y^2,$$

from which it follows that

$$\beta = \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x^2 - y^2}{2L_1L_2}\right).$$

Also from the law of cosines,

$$\alpha = \cos^{-1}\left(\frac{x^2 + y^2 + L_1^2 - L_2^2}{2L_1\sqrt{x^2+y^2}}\right).$$

The angle γ is determined using the two-argument arctangent function, $\gamma = \text{atan2}(y, x)$. With these angles, the righty solution to the inverse kinematics is

$$\theta_1 = \gamma - \alpha, \qquad \theta_2 = \pi - \beta$$

and the lefty solution is

$$\theta_1 = \gamma + \alpha, \qquad \theta_2 = \beta - \pi.$$

If $x^2 + y^2$ lies outside the range $[L_1 - L_2, L_1 + L_2]$ then no solution exists.

This simple motivational example illustrates that, for open chains, the inverse kinematics problem may have multiple solutions; this situation is in contrast with the forward kinematics, where a unique end-effector displacement T exists for given joint values θ. In fact, three-link planar open chains have an infinite number of solutions for points (x, y) lying in the interior of the workspace; in this case the chain possesses an extra degree of freedom and is said to be kinematically redundant.

In this chapter we first consider the inverse kinematics of spatial open chains with six degrees of freedom. At most a finite number of solutions exists in this case, and we consider two popular structures – the PUMA and Stanford robot arms – for which analytic inverse kinematic solutions can be easily obtained. For more general open chains, we adapt the Newton–Raphson method to the inverse kinematics problem. The result is an iterative numerical algorithm which, provided that an initial guess of the joint variables is sufficiently close to a true solution, converges quickly to that solution.

6.1 Analytic Inverse Kinematics

We begin by writing the forward kinematics of a spatial six-dof open chain in the following product of exponentials form:

$$T(\theta) = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6} M.$$

Given some end-effector frame $X \in SE(3)$, the inverse kinematics problem is to find solutions $\theta \in \mathbb{R}^6$ satisfying $T(\theta) = X$. In the following subsections we derive analytic inverse kinematic solutions for the PUMA and Stanford arms.

6.1.1 6R PUMA-Type Arm

We first consider a 6R arm of the PUMA type. Referring to Figure 6.2, when the arm is placed in its zero position: (i) the two shoulder joint axes intersect orthogonally at a common point, with joint axis 1 aligned in the $\hat{z}_0$-direction and joint axis 2 aligned in the $-\hat{y}_0$-direction; (ii) joint axis 3 (the elbow joint) lies in the $\hat{x}_0$–$\hat{y}_0$-plane and is aligned parallel with joint axis 2; (iii) joint axes 4, 5, and 6 (the wrist joints) intersect orthogonally at a common point (the wrist center) to form an orthogonal wrist and, for the purposes of this example, we assume

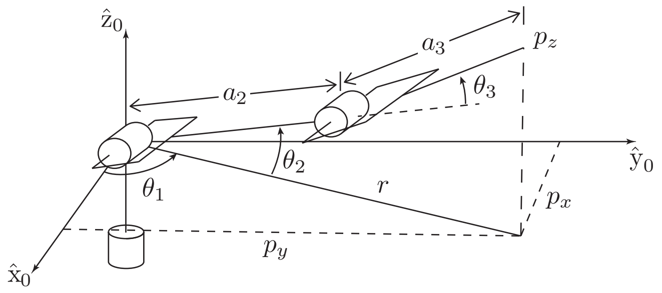
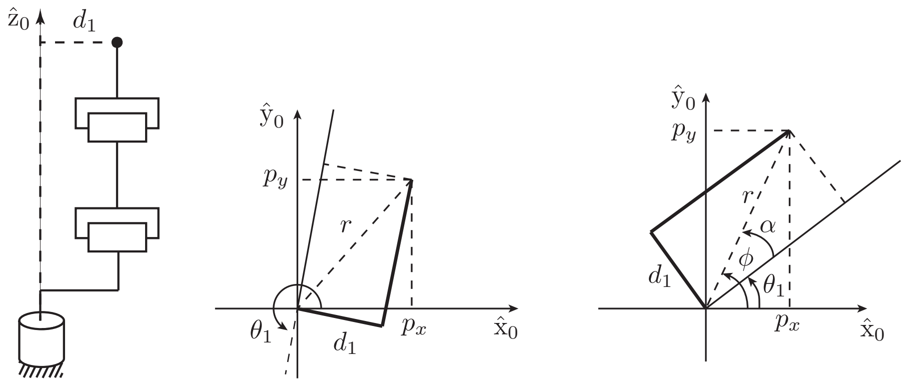


Figure 6.2: Inverse position kinematics of a 6R PUMA-type arm.



(a) Elbow arm with offset.

(b) Kinematic diagram.

Figure 6.3: A 6R PUMA-type arm with a shoulder offset.

that these joint axes are aligned in the $\hat{z}_0$-, $\hat{y}_0$-, and $\hat{x}_0$-directions, respectively. The lengths of links 2 and 3 are a_2 and a_3, respectively. The arm may also have an offset at the shoulder (see Figure 6.3). The inverse kinematics problem for PUMA-type arms can be decoupled into inverse-position and inverse-orientation subproblems, as we now show.

We first consider the simple case of a zero-offset PUMA-type arm. Referring to Figure 6.2 and expressing all vectors in terms of fixed-frame coordinates, denote the components of the wrist center $p \in \mathbb{R}^3$ by $p = (p_x, p_y, p_z)$. Projecting p onto the $\hat{x}_0$–$\hat{y}_0$-plane, it can be seen that

$$\theta_1 = \text{atan2}(p_y, p_x).$$

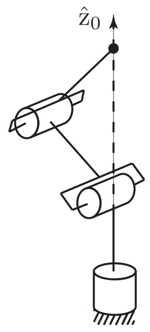


Figure 6.4: Singular configuration of the zero-offset 6R PUMA-type arm.

Note that a second valid solution for θ_1 is given by

$$\theta_1 = \operatorname{atan2}(p_y, p_x) + \pi,$$

when the original solution for θ_2 is replaced by $\pi - \theta_2$. As long as $p_x, p_y \neq 0$ both these solutions are valid. When $p_x = p_y = 0$ the arm is in a singular configuration (see Figure 6.4), and there are infinitely many possible solutions for θ_1.

If there is an offset $d_1 \neq 0$ as shown in Figure 6.3, then in general there will be two solutions for θ_1, the righty and lefty solutions (Figure 6.3). As seen from the figure, $\theta_1 = \phi - \alpha$ where $\phi = \operatorname{atan2}(p_y, p_x)$ and $\alpha = \operatorname{atan2}(d_1, \sqrt{r^2 - d_1^2})$. The second solution is given by

$$\theta_1 = \pi + \operatorname{atan2}(p_y, p_x) + \operatorname{atan2}\left(-\sqrt{p_x^2 + p_y^2 - d_1^2}, d_1\right).$$

Determining angles θ_2 and θ_3 for the PUMA-type arm now reduces to the inverse kinematics problem for a planar two-link chain:

$$\begin{aligned}\cos\theta_3 &= \frac{r^2 - d_1^2 + p_z^2 - a_2^2 - a_3^2}{2a_2a_3} \\ &= \frac{p_x^2 + p_y^2 + p_z^2 - d_1^2 - a_2^2 - a_3^2}{2a_2a_3} = D.\end{aligned}$$

Using D defined above, θ_3 is given by

$$\theta_3 = \operatorname{atan2}\left(\pm\sqrt{1 - D^2}, D\right)$$

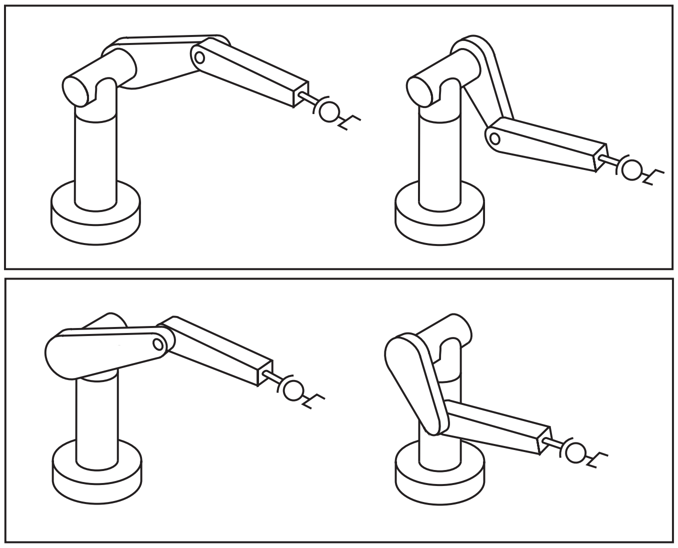


Figure 6.5: Four possible inverse kinematics solutions for the 6R PUMA-type arm with shoulder offset.

and θ_2 can be obtained in a similar fashion as

$$\begin{aligned}\theta_2 &= \operatorname{atan2}\left(p_z, \sqrt{r^2 - d_1^2}\right) - \operatorname{atan2}\left(a_3 \mathrm{s}_3, a_2 + a_3 \mathrm{c}_3\right) \\ &= \operatorname{atan2}\left(p_z, \sqrt{p_x^2 + p_y^2 - d_1^2}\right) - \operatorname{atan2}\left(a_3 \mathrm{s}_3, a_2 + a_3 \mathrm{c}_3\right),\end{aligned}$$

where $\mathrm{s}_3 = \sin\theta_3$ and $\mathrm{c}_3 = \cos\theta_3$. The two solutions for θ_3 correspond to the well-known elbow-up and elbow-down configurations for the two-link planar arm. In general, a PUMA-type arm with an offset will have four solutions to the inverse position problem, as shown in Figure 6.5; the postures in the upper panel are lefty solutions (elbow-up and elbow-down), while those in the lower panel are righty solutions (elbow-up and elbow-down).

We now solve the inverse orientation problem of finding $(\theta_4, \theta_5, \theta_6)$ given the end-effector orientation. This problem is completely straightforward: having found $(\theta_1, \theta_2, \theta_3)$, the forward kinematics can be manipulated into the form

$$e^{[\mathcal{S}_4]\theta_4} e^{[\mathcal{S}_5]\theta_5} e^{[\mathcal{S}_6]\theta_6} = e^{-[\mathcal{S}_3]\theta_3} e^{-[\mathcal{S}_2]\theta_2} e^{-[\mathcal{S}_1]\theta_1} X M^{-1}, \tag{6.2}$$

where the right-hand side is now known, and the ω_i-components of $\mathcal{S}_4$, $\mathcal{S}_5$, and

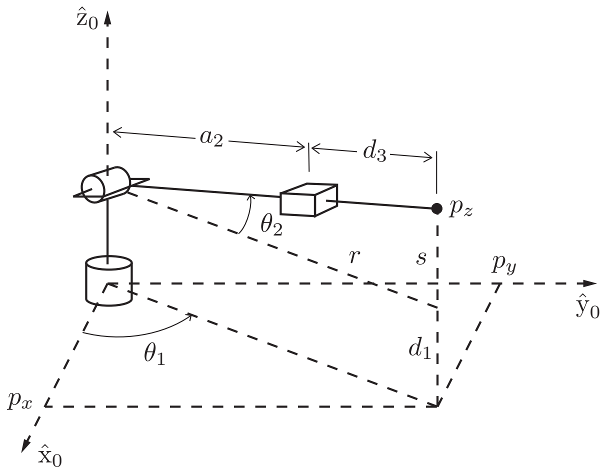


Figure 6.6: The first three joints of a Stanford-type arm.

$\mathcal{S}_6$ are

$$\begin{aligned}\omega_4 &= (0,0,1),\\ \omega_5 &= (0,1,0),\\ \omega_6 &= (1,0,0).\end{aligned}$$

Denoting the $SO(3)$ component of the right-hand side of Equation (6.2) by R, the wrist joint angles $(\theta_4, \theta_5, \theta_6)$ can be determined as the solution to

$$\text{Rot}(\hat{z}, \theta_4)\text{Rot}(\hat{y}, \theta_5)\text{Rot}(\hat{x}, \theta_6) = R,$$

which correspond exactly to the ZYX Euler angles, derived in Appendix B.

6.1.2 Stanford-Type Arms

If the elbow joint in a 6R PUMA-type arm is replaced by a prismatic joint, as shown in Figure 6.6, we then have an RRPRRR Stanford-type arm. Here we consider the inverse position kinematics for the arm of Figure 6.6; the inverse orientation kinematics is identical to that for the PUMA-type arm and so is not repeated here.

The first joint variable θ_1 can be found in similar fashion to the PUMA-type arm: $\theta_1 = \text{atan2}(p_y, p_x)$ (provided that p_x and p_y are not both zero). The variable θ_2 is then found from Figure 6.6 to be

$$\theta_2 = \text{atan2}(s, r),$$

where $r^2 = p_x^2 + p_y^2$ and $s = p_z - d_1$. Similarly to the case of the PUMA-type arm, a second solution for θ_1 and θ_2 is given by

$$\begin{aligned}\theta_1 &= \pi + \operatorname{atan2}(p_y, p_x),\\ \theta_2 &= \pi - \operatorname{atan2}(s, r).\end{aligned}$$

The translation distance θ_3 is found from the relation

$$(\theta_3 + a_2)^2 = r^2 + s^2$$

as

$$\theta_3 = \sqrt{r^2 + s^2} = \sqrt{p_x^2 + p_y^2 + (p_z - d_1)^2} - a_2.$$

Ignoring the negative square root solution for θ_3, we obtain two solutions to the inverse position kinematics as long as the wrist center p does not intersect the $\hat{\mathrm{z}}_0$-axis of the fixed frame. If there is an offset then, as in the case of the PUMA-type arm, there will be lefty and righty solutions.

6.2 Numerical Inverse Kinematics

Iterative numerical methods can be applied if the inverse kinematics equations do not admit analytic solutions. Even in cases where an analytic solution does exist, numerical methods are often used to improve the accuracy of these solutions. For example, in a PUMA-type arm, the last three axes may not exactly intersect at a common point, and the shoulder joint axes may not be exactly orthogonal. In such cases, rather than throw away any analytic inverse kinematic solutions that are available, such solutions can be used as the initial guess in an iterative numerical procedure for solving the inverse kinematics.

There exist a variety of iterative methods for finding the roots of a nonlinear equation, and our aim is not to discuss these in detail – any text on numerical analysis will cover these methods in depth – but rather to develop ways in which to transform the inverse kinematics equations so that they become amenable to existing numerical methods. We will make use of an approach fundamental to nonlinear root-finding, the Newton–Raphson method. Also, methods of optimization are needed in situations where an exact solution may not exist and we seek the closest approximate solution; or, conversely, an infinity of inverse kinematics solutions exists (i.e., if the robot is kinematically redundant) and we seek a solution that is optimal with respect to some criterion. We now therefore discuss the Newton–Raphson method for nonlinear root-finding and also the first-order necessary conditions for optimization.

6.2.1 Newton–Raphson Method

To solve the equation $g(\theta) = 0$ numerically for a given differentiable function $g : \mathbb{R} \to \mathbb{R}$, assume θ^0 is an initial guess for the solution. Write the Taylor expansion of $g(\theta)$ at θ^0 and truncate it at first order:

$$g(\theta) = g(\theta^0) + \frac{\partial g}{\partial \theta}(\theta^0)(\theta - \theta^0) + \text{higher-order terms (h.o.t)}.$$

Keeping only the terms up to first order, set $g(\theta) = 0$ and solve for θ to obtain

$$\theta = \theta^0 - \left(\frac{\partial g}{\partial \theta}(\theta^0)\right)^{-1} g(\theta^0).$$

Using this value of θ as the new guess for the solution and repeating the above, we get the following iteration:

$$\theta^{k+1} = \theta^k - \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k).$$

The above iteration is repeated until some stopping criterion is satisified, e.g., $|g(\theta^k) - g(\theta^{k+1})|/|g(\theta^k)| \leq \epsilon$ for some user-prescribed threshold value ϵ.

The same formula applies for the case when g is multi-dimensional, i.e., $g : \mathbb{R}^n \to \mathbb{R}^n$, in which case

$$\frac{\partial g}{\partial \theta}(\theta) = \begin{bmatrix} \frac{\partial g_1}{\partial \theta_1}(\theta) & \cdots & \frac{\partial g_1}{\partial \theta_n}(\theta) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial \theta_1}(\theta) & \cdots & \frac{\partial g_n}{\partial \theta_n}(\theta) \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The case where the above matrix fails to be invertible is discussed in Section 6.2.2.

6.2.2 Numerical Inverse Kinematics Algorithm

Suppose we express the end-effector frame using a coordinate vector x governed by the forward kinematics $x = f(\theta)$, a nonlinear vector equation mapping the n joint coordinates to the m end-effector coordinates. Assume that $f : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable, and let x_d be the desired end-effector coordinates. Then $g(\theta)$ for the Newton-Raphson method is defined as $g(\theta) = x_d - f(\theta)$, and the goal is to find joint coordinates θ_d such that

$$g(\theta_d) = x_d - f(\theta_d) = 0.$$

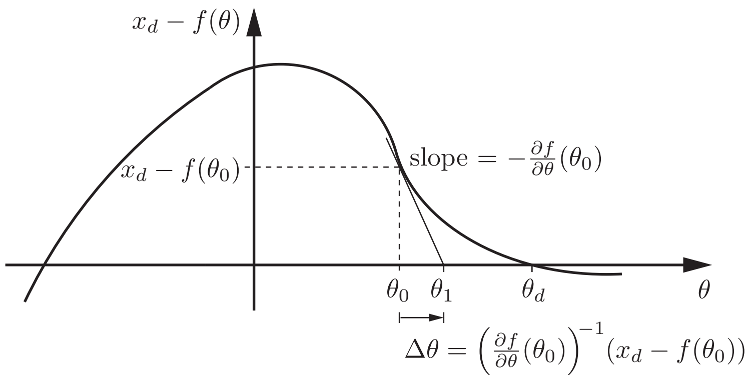

$x_d - f(\theta)$
$x_d - f(\theta_0)$
slope $= -\frac{\partial f}{\partial \theta}(\theta_0)$
θ_0
θ_1
θ_d
θ
$\Delta\theta = \left(\frac{\partial f}{\partial \theta}(\theta_0)\right)^{-1}(x_d - f(\theta_0))$

Figure 6.7: The first step of the Newton–Raphson method for nonlinear root-finding for a scalar x and θ. In the first step, the slope $-\partial f/\partial\theta$ is evaluated at the point $(\theta^0, x_d - f(\theta^0))$. In the second step, the slope is evaluated at the point $(\theta^1, x_d - f(\theta^1))$ and eventually the process converges to θ_d. Note that an initial guess to the left of the plateau of $x_d - f(\theta)$ would be likely to result in convergence to the other root of $x_d - f(\theta)$, and an initial guess at or near the plateau would result in a large initial $|\Delta\theta|$ and the iterative process might not converge at all.

Given an initial guess θ^0 which is "close to" a solution θ_d, the kinematics can be expressed as the Taylor expansion

$$x_d = f(\theta_d) = f(\theta^0) + \underbrace{\left.\frac{\partial f}{\partial \theta}\right|_{\theta^0}}_{J(\theta^0)} \underbrace{(\theta_d - \theta^0)}_{\Delta\theta} + \text{h.o.t.}, \tag{6.3}$$

where $J(\theta^0) \in \mathbb{R}^{m\times n}$ is the coordinate Jacobian evaluated at θ^0. Truncating the Taylor expansion at first order, we can approximate Equation (6.3) as

$$J(\theta^0)\Delta\theta = x_d - f(\theta^0). \tag{6.4}$$

Assuming that $J(\theta^0)$ is square ($m = n$) and invertible, we can solve for $\Delta\theta$ as

$$\Delta\theta = J^{-1}(\theta^0)\left(x_d - f(\theta^0)\right). \tag{6.5}$$

If the forward kinematics is linear in θ, i.e., the higher-order terms in Equation (6.3) are zero, then the new guess $\theta^1 = \theta^0 + \Delta\theta$ exactly satisfies $x_d = f(\theta^1)$. If the forward kinematics is not linear in θ, as is usually the case, the new guess θ^1 should still be closer to the root than θ^0, and the process is then repeated, producing a sequence $\{\theta^0, \theta^1, \theta^2, \ldots\}$ converging to θ_d (Figure 6.7).

As indicated in Figure 6.7, if there are multiple inverse kinematics solutions, the iterative process tends to converge to the solution that is "closest" to the initial guess θ^0. You can think of each solution as having its own basin of attraction. If the initial guess is not in one of these basins (e.g., the initial guess is not sufficiently close to a solution), the iterative process may not converge.

In practice, for computational efficiency reasons, Equation (6.4) is often solved without directly calculating the inverse $J^{-1}(\theta^0)$. More efficient techniques exist for solving a set of linear equations $Ax = b$ for x. For example, for invertible square matrices A, the LU decomposition of A can be used to solve for x with fewer operations. In MATLAB, for example, the syntax

```
x = A\b
```

solves $Ax = b$ for x without computing A^{-1}.

If J is not invertible, either because it is not square or because it is singular, then J^{-1} in Equation (6.5) does not exist. Equation (6.4) can still be solved (or approximately solved) for $\Delta\theta$ by replacing J^{-1} in Equation (6.5) with the Moore–Penrose **pseudoinverse** $J^\dagger$. For any equation of the form $Jy = z$, where $J \in \mathbb{R}^{m\times n}$, $y \in \mathbb{R}^n$, and $z \in \mathbb{R}^m$, the solution

$$y^* = J^\dagger z$$

falls into one of two categories:

- The solution y^* exactly satisfies $Jy^* = z$ and, for any solution y exactly satisfying $Jy = z$, we have $\|y^*\| \leq \|y\|$. In other words, among all solutions, y^* minimizes the two-norm. There can be an infinite number of solutions y to $Jy = z$ if the robot has more joints n than end-effector coordinates m, i.e., the Jacobian J is "fat."

- If there is no y that exactly satisfies $Jy = z$ then y^* minimizes the two-norm of the error, i.e., $\|Jy^* - z\| \leq \|Jy - z\|$ for any $y \in \mathbb{R}^n$. This case corresponds to $\operatorname{rank} J < m$, i.e., the robot has fewer joints n than end-effector coordinates m (a "tall" Jacobian J) or it is at a singularity.

Many programming languages provide functions to calculate the pseudoinverse; for example, the usage in MATLAB is

```
y = pinv(J) * z
```

In the case where J is full rank (rank m for $n > m$ or rank n for $n < m$), i.e., the robot is not at a singularity, the pseudoinverse can be calculated as

$$\begin{array}{ll} J^\dagger = J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1} & \text{if } J \text{ is fat, } n > m \text{ (called a right inverse since } JJ^\dagger = I) \\ J^\dagger = (J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}} & \text{if } J \text{ is tall, } n < m \text{ (called a left inverse since } J^\dagger J = I). \end{array}$$

Replacing the Jacobian inverse with the pseudoinverse, Equation (6.5) becomes

$$\Delta\theta = J^{\dagger}(\theta^0)\left(x_d - f(\theta^0)\right). \tag{6.6}$$

If $\text{rank}(J) < m$ then the solution $\Delta\theta$ calculated in Equation (6.6) may not exactly satisfy Equation (6.4), but it satisfies this condition as closely as possible in a least-squares sense. If $n > m$ then the solution is the smallest joint variable change (in the two-norm sense) that exactly satisfies Equation (6.4).

Equation (6.6) suggests using the Newton–Raphson iterative algorithm for finding θ_d:

(a) **Initialization**: Given $x_d \in \mathbb{R}^m$ and an initial guess $\theta^0 \in \mathbb{R}^n$, set $i = 0$.

(b) Set $e = x_d - f(\theta^i)$. While $\|e\| > \epsilon$ for some small ϵ:

- Set $\theta^{i+1} = \theta^i + J^{\dagger}(\theta^i)e$.
- Increment i.

To modify this algorithm to work with a desired end-effector configuration represented as $T_{sd} \in SE(3)$ instead of as a coordinate vector x_d, we can replace the coordinate Jacobian J with the end-effector body Jacobian $J_b \in \mathbb{R}^{6\times n}$. Note, however, that the vector $e = x_d - f(\theta^i)$, representing the direction from the current guess (evaluated through the forward kinematics) to the desired end-effector configuration, cannot simply be replaced by $T_{sd} - T_{sb}(\theta^i)$; the pseudoinverse of J_b should act on a body twist $\mathcal{V}_b \in \mathbb{R}^6$. To find the right analogy, we should think of $e = x_d - f(\theta^i)$ as a velocity vector which, if followed for unit time, would cause a motion from $f(\theta^i)$ to x_d. Similarly, we should look for a body twist $\mathcal{V}_b$ which, if followed for unit time, would cause a motion from $T_{sb}(\theta^i)$ to the desired configuration T_{sd}.

To find this $\mathcal{V}_b$, we first calculate the desired configuration in the body frame,

$$T_{bd}(\theta^i) = T_{sb}^{-1}(\theta^i)T_{sd} = T_{bs}(\theta^i)T_{sd}.$$

Then $\mathcal{V}_b$ is determined using the matrix logarithm,

$$[\mathcal{V}_b] = \log T_{bd}(\theta^i).$$

This leads to the following inverse kinematics algorithm, which is analogous to the above coordinate-vector algorithm:

(a) **Initialization**: Given T_{sd} and an initial guess $\theta^0 \in \mathbb{R}^n$, set $i = 0$.

(b) Set $[\mathcal{V}_b] = \log\left(T_{sb}^{-1}(\theta^i)T_{sd}\right)$. While $\|\omega_b\| > \epsilon_\omega$ or $\|v_b\| > \epsilon_v$ for small $\epsilon_\omega, \epsilon_v$:

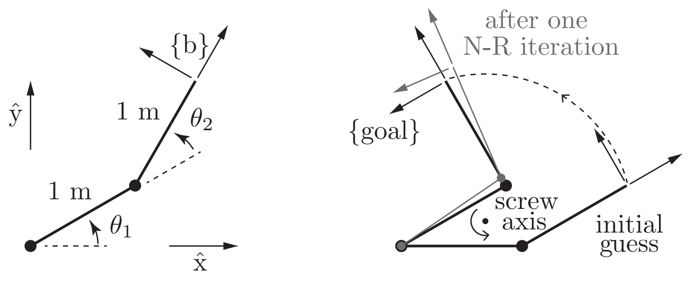


Figure 6.8: (Left) A 2R robot. (Right) The goal is to find the joint angles yielding the end-effector frame {goal} corresponding to $\theta_1 = 30°$ and $\theta_2 = 90°$. The initial guess is $(0°, 30°)$. After one Newton–Raphson iteration, the calculated joint angles are $(34.23°, 79.18°)$. The screw axis that takes the initial frame to the goal frame (by means of the curved dashed line) is also indicated.

- Set $\theta^{i+1} = \theta^i + J_b^\dagger(\theta^i)\mathcal{V}_b$.
- Increment i.

An equivalent form can be derived in the space frame, using the space Jacobian $J_s(\theta)$ and the spatial twist $\mathcal{V}_s = [\mathrm{Ad}_{T_{sb}}]\mathcal{V}_b$.

For this numerical inverse kinematics method to converge, the initial guess θ^0 should be sufficiently close to a solution θ_d. This condition can be satisfied by starting the robot from an initial home configuration where both the actual end-effector configuration and the joint angles are known and ensuring that the requested end-effector position T_{sd} changes slowly relative to the frequency of the calculation of the inverse kinematics. Then, for the rest of the robot's run, the calculated θ_d at the previous timestep serves as the initial guess θ^0 for the new T_{sd} at the next timestep.

Example 6.1 (Planar 2R robot)**.** Now we apply the body Jacobian Newton–Raphson inverse kinematics algorithm to the 2R robot in Figure 6.8. Each link is 1 m in length, and we would like to find the joint angles that place the tip of the robot at $(x, y) = (0.366 \text{ m}, 1.366 \text{ m})$, which corresponds to $\theta_d = (30°, 90°)$ and

$$T_{sd} = \begin{bmatrix} -0.5 & -0.866 & 0 & 0.366 \\ 0.866 & -0.5 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

as shown by the frame {goal} in Figure 6.8. The forward kinematics, expressed in the end-effector frame, is given by

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathcal{B}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \quad \mathcal{B}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Our initial guess at the solution is $\theta^0 = (0, 30°)$, and we specify an error tolerance of $\epsilon_\omega = 0.001$ rad (or 0.057°) and $\epsilon_v = 10^{-4}$ m (100 microns). The progress of the Newton–Raphson method is illustrated in the table below, where only the $(\omega_{zb}, v_{xb}, v_{yb})$-components of the body twist $\mathcal{V}_b$ are given since the robot's motion is restricted to the x–y-plane:

i	(θ_1, θ_2)	(x, y)	$\mathcal{V}_b = (\omega_{zb}, v_{xb}, v_{yb})$	$\|\omega_b\|$	$\|v_b\|$
0	$(0.00, 30.00°)$	$(1.866, 0.500)$	$(1.571, 0.498, 1.858)$	1.571	1.924
1	$(34.23°, 79.18°)$	$(0.429, 1.480)$	$(0.115, -0.074, 0.108)$	0.115	0.131
2	$(29.98°, 90.22°)$	$(0.363, 1.364)$	$(-0.004, 0.000, -0.004)$	0.004	0.004
3	$(30.00°, 90.00°)$	$(0.366, 1.366)$	$(0.000, 0.000, 0.000)$	0.000	0.000

The iterative procedure converges to within the tolerances after three iterations. Figure 6.8 shows the initial guess, the goal configuration, and the configuration after one iteration. Notice that the first v_{xb} calculated is positive, even though the origin of the goal frame is in the $-\hat{\mathrm{x}}_\mathrm{b}$-direction of the initial guess. The reason is that the constant body velocity $\mathcal{V}_b$ that takes the initial guess to {goal} in one second is a rotation about the screw axis indicated in the figure.

6.3 Inverse Velocity Kinematics

One solution for controlling a robot so that it follows a desired end-effector trajectory $T_{sd}(t)$ is to calculate the inverse kinematics $\theta_d(k\Delta t)$ at each discrete timestep k, then control the joint velocities $\dot{\theta}$ as follows

$$\dot{\theta} = \big(\theta_d(k\Delta t) - \theta((k-1)\Delta t)\big)/\Delta t$$

during the time interval $[(k-1)\Delta t, k\Delta t]$. This amounts to a feedback controller since the desired new joint angles $\theta_d(k\Delta t)$ are being compared with the most recently measured actual joint angles $\theta((k-1)\Delta t)$ in order to calculate the required joint velocities.

Another option that avoids the computation of inverse kinematics is to calculate the required joint velocities $\dot{\theta}$ directly from the relationship $J\dot{\theta} = \mathcal{V}_d$, where the desired end-effector twist $\mathcal{V}_d$ and J are expressed with respect to the same frame:

$$\dot{\theta} = J^\dagger(\theta)\mathcal{V}_d. \tag{6.7}$$

The desired twist $\mathcal{V}_d(t)$ can be chosen to be $T_{sd}^{-1}(t)\dot{T}_{sd}(t)$ (the body twist of the desired trajectory at time t) or $\dot{T}_{sd}(t)T_{sd}^{-1}(t)$ (the spatial twist), depending on whether the body Jacobian or space Jacobian is used; however small velocity errors are likely to accumulate over time, resulting in increasing position error. Thus, a position feedback controller should choose $\mathcal{V}_d(t)$ so as to keep the end-effector following $T_{sd}(t)$ with little position error. Feedback control is discussed in Chapter 11.

In the case of a redundant robot with $n > 6$ joints, of the $(n-6)$-dimensional set of joint velocities satisfying Equation (6.7), the use of the pseudoinverse $J^\dagger(\theta)$ returns joint velocities $\dot{\theta}$ minimizing the two-norm $\|\dot{\theta}\| = \sqrt{\dot{\theta}^{\mathrm{T}}\dot{\theta}}$.

The use of the pseudoinverse in Equation (6.7) implicitly weights the cost of each joint velocity identically. We could instead give the joint velocities different weights; for example, the velocity at the first joint, which moves a lot of the robot's mass, could be weighted more heavily than the velocity at the last joint, which moves little of the robot's mass. As we will see later, the kinetic energy of a robot can be written

$$\frac{1}{2}\dot{\theta}^{\mathrm{T}}M(\theta)\dot{\theta},$$

where $M(\theta)$ is the symmetric, positive-definite, configuration-dependent mass matrix of the robot. The mass matrix $M(\theta)$ can be used as a weighting function in the inverse velocity kinematics, and the goal is to find the $\dot{\theta}$ that minimizes the kinetic energy while also satisfying $J(\theta)\dot{\theta} = \mathcal{V}_d$.

Another possibility is to find the $\dot{\theta}$ that causes the robot to minimize a configuration-dependent potential energy function $h(\theta)$ while satisfying $J(\theta)\dot{\theta} = \mathcal{V}_d$. For example, $h(\theta)$ could be the gravitational potential energy, or an artificial potential function whose value increases as the robot approaches an obstacle. Then the rate of change of $h(\theta)$ is

$$\frac{d}{dt}h(\theta) = \frac{dh(\theta)}{d\theta}\frac{d\theta}{dt} = \nabla h(\theta)^{\mathrm{T}}\dot{\theta},$$

where $\nabla h(\theta)$ points in the direction of maximum ascent of $h(\theta)$.

More generally, we may wish to minimize the sum of the kinetic energy and the rate of change of the potential energy:

$$\min_{\dot{\theta}} \frac{1}{2}\dot{\theta}^{\mathrm{T}}M(\theta)\dot{\theta} + \nabla h(\theta)^{\mathrm{T}}\dot{\theta},$$

subject to the constraint $J(\theta)\dot{\theta} = \mathcal{V}_d$. From the first-order necessary conditions for optimality (Appendix D)

$$\begin{aligned} J^{\mathrm{T}}\lambda &= M\dot{\theta} + \nabla h, \\ \mathcal{V}_d &= J\dot{\theta}, \end{aligned}$$

the optimal $\dot{\theta}$ and λ can be derived as follows:

$$\begin{aligned} \dot{\theta} &= G\mathcal{V}_d + (I - GJ)M^{-1}\nabla h, \\ \lambda &= B\mathcal{V}_d + BJM^{-1}\nabla h, \end{aligned}$$

where $B \in \mathbb{R}^{m\times m}$ and $G \in \mathbb{R}^{n\times m}$ are defined by

$$\begin{aligned} B &= (JM^{-1}J^{\mathrm{T}})^{-1}, \\ G &= M^{-1}J^{\mathrm{T}}(JM^{-1}J^{\mathrm{T}})^{-1} = M^{-1}J^{\mathrm{T}}B. \end{aligned}$$

Recalling the static relation $\tau = J^{\mathrm{T}}\mathcal{F}$ from the previous chapter, the Lagrange multiplier λ (see Appendix D) can be interpreted as a wrench in task space. Moreover, in the expression $\lambda = B\mathcal{V}_d + BJM^{-1}\nabla h$, the first term, $B\mathcal{V}_d$, can be interpreted as a dynamic force generating the end-effector velocity $\mathcal{V}_d$ while the second term, $BJM^{-1}\nabla h$, can be interpreted as the static wrench counteracting gravity.

If the potential function $h(\theta)$ is zero or unspecified, the kinetic-energy-minimizing solution is

$$\dot{\theta} = M^{-1}J^{\mathrm{T}}(JM^{-1}J^{\mathrm{T}})^{-1}\mathcal{V}_d,$$

where $M^{-1}J^{\mathrm{T}}(JM^{-1}J^{\mathrm{T}})^{-1}$ is the weighted pseudoinverse according to the mass matrix $M(\theta)$.

6.4 A Note on Closed Loops

A desired end-effector trajectory over a time interval $[0, t_f]$ is a closed loop if $T_{sd}(0) = T_{sd}(t_f)$. It should be noted that numerical methods for calculating inverse kinematics for redundant robots, at either the configuration or velocity levels, are likely to yield motions that are not closed loops in the joint space, i.e., $\theta(0) \neq \theta(t_f)$. If closed-loop motions in joint space are required, an extra set of conditions on the inverse kinematics must be satisfied.

6.5 Summary

- Given a spatial open chain with forward kinematics $T(\theta)$, $\theta \in \mathbb{R}^n$, in the inverse kinematics problem one seeks to find, for a desired end-effector configuration $X \in SE(3)$, solutions θ that satisfy $X = T(\theta)$. Unlike the forward kinematics problem, the inverse kinematics problem can possess multiple solutions, or no solutions in the event that X lies outside the workspace. For a spatial open chain with n joints and an X in the workspace, $n = 6$ typically leads to a finite number of inverse kinematic solutions while $n > 6$ leads to an infinite number of solutions.

- The inverse kinematics can be solved analytically for the six-dof PUMA-type robot arm, a popular 6R design consisting of a 3R orthogonal axis wrist connected to a 2R orthogonal axis shoulder by an elbow joint.

- Stanford-type arms also admit analytic inverse kinematics solutions. These arms are obtained by replacing the elbow joint in the generalized 6R PUMA-type arm by a prismatic joint. Geometric inverse kinematic algorithms similar to those for PUMA-type arms have been developed.

- Iterative numerical methods are used in cases where analytic inverse kinematic solutions are unavailable. These methods typically involve solving the inverse kinematics equations using an iterative procedure like the Newton–Raphson method, and they require an initial guess at the joint variables. The performance of the iterative procedure depends to a large extent on the quality of the initial guess and, in the case where there are several possible inverse kinematic solutions, the method finds the solution that is "closest" to the initial guess. Each iteration is of the form

$$\dot{\theta}^{i+1} = \theta^i + J^\dagger(\theta^i)\mathcal{V},$$

  where $J^\dagger(\theta)$ is the pseudoinverse of the Jacobian $J(\theta)$ and $\mathcal{V}$ is the twist that takes $T(\theta^i)$ to T_{sd} in one second.

6.6 Software

Software functions associated with this chapter are listed below.

`[thetalist,success] = IKinBody(Blist,M,T,thetalist0,eomg,ev)`
This function uses iterative Newton–Raphson to calculate the inverse kinematics given the list of joint screws $\mathcal{B}_i$ expressed in the end-effector frame, the end-effector home configuration M, the desired end-effector configuration T, an

initial guess at the joint angles θ^0, and the tolerances ϵ_ω and ϵ_v on the final error. If a solution is not found within a maximum number of iterations, then `success` is false.

`[thetalist,success] = IKinSpace(Slist,M,T,thetalist0,eomg,ev)`
This is similar to `IKinBody`, except that the joint screws $\mathcal{S}_i$ are expressed in the space frame and the tolerances are interpreted in the space frame, also.

6.7 Notes and References

The inverse kinematics of the most general 6R open chain is known to have up to 16 solutions; this result was proved by Lee and Liang [87] and by Raghavan and Roth [143]. Procedures for finding closed-form inverse kinematics solutions to somewhat more general six-dof open chains than those treated in this chapter are described in [129, 122]; these procedures use solutions to a collection of some basic screw-theoretic subproblems, called the Paden–Kahan subproblems, e.g., finding the angle of rotation for a zero-pitch screw motion between a pair of given points. Iterative numerical procedures for finding all 16 solutions of a general 6R open chain are reported in [104].

A comprehensive summary of inverse kinematics methods for kinematically redundant robot arms are discussed in [26]. Many of these methods rely on results and solution techniques from least-squares optimization, and for this reason we provide a brief review of the basics of optimization in Appendix D; a classic reference for optimization is [98]. The repeatability (or cyclicity) conditions for a general class of inverse kinematic redundancy resolution schemes are examined in [162].

6.8 Exercises

Exercise 6.1 Write a program that solves the analytical inverse kinematics for a planar 3R robot with link lengths $L_1 = 3$, $L_2 = 2$, and $L_3 = 1$, given the desired position (x, y) and orientation θ of a frame fixed to the tip of the robot. Each joint has no joint limits. Your program should find all the solutions (how many are there in the general case?), give the joint angles for each, and draw the robot in these configurations. Test the code for the case of $(x, y, \theta) = (4, 2, 0)$.

Exercise 6.2 Solve the inverse position kinematics (you do not need to solve the orientation kinematics) of the 6R open chain shown in Figure 6.9.

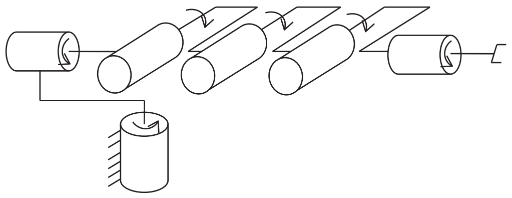


Figure 6.9: A 6R open chain.

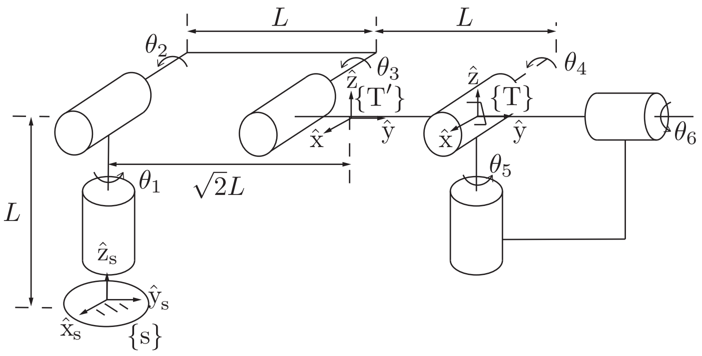


Figure 6.10: A 6R open chain.

Exercise 6.3 Find the inverse kinematics solutions when the end-effector frame $\{T\}$ of the 6R open chain shown in Figure 6.10 is set to $\{T'\}$ as shown. The orientation of $\{T\}$ at the zero position is the same as that of the fixed frame $\{s\}$, and $\{T'\}$ is the result of a pure translation of $\{T\}$ along the $\hat{y}_s$-axis.

Exercise 6.4 The RRP open chain of Figure 6.11 is shown in its zero position. Joint axes 1 and 2 intersect at the fixed frame origin, and the end-effector frame origin p is located at $(0, 1, 0)$ when the robot is in its zero position.

(a) Suppose that $\theta_1 = 0$. Solve for θ_2 and θ_3 when the end-effector frame origin p is at $(-6, 5, \sqrt{3})$.

(b) If joint 1 is not fixed to zero but instead allowed to vary, find all the inverse kinematic solutions $(\theta_1, \theta_2, \theta_3)$ for the p given in (a).

Exercise 6.5 The four-dof robot of Figure 6.12 is shown in its zero position.

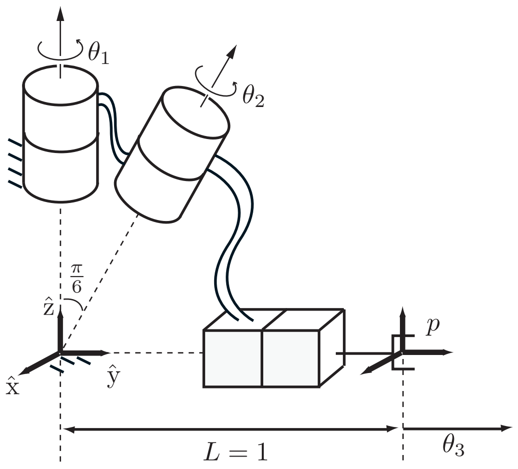


Figure 6.11: An RRP open chain.

Joint 1 is a screw joint of pitch h. Given the end-effector position $p = (p_x, p_y, p_z)$ and orientation $R = e^{[\hat{z}]\alpha}$, where $\hat{z} = (0, 0, 1)$ and $\alpha \in [0, 2\pi]$, find the inverse kinematics solution $(\theta_1, \theta_2, \theta_3, \theta_4)$ as a function of p and α.

Exercise 6.6 Figure 6.13(a) shows a surgical robot, which can be modeled as an RRPRRP open chain as shown in Figure 6.13(b).

(a) In the general case, how many inverse kinematic solutions will exist for a given end-effector frame?

(b) Consider points A and B on the surgical robot shown in Figure 6.13(b). Given coordinates (x_A, y_A, z_A) and (x_B, y_B, z_B) for the points A and B in the fixed frame, find the joint variables θ_1, θ_2, θ_3, θ_4, and θ_5. You should find an explicit formula for $(\theta_1, \theta_2, \theta_3)$ while for (θ_4, θ_5) you can just describe the procedure.

Exercise 6.7 In this exercise you are asked to draw a plot of a scalar $x_d - f(\theta)$ versus a scalar θ (similar to Figure 6.7) with two roots. Draw it so that, for some initial guess θ^0, the iterative process actually jumps over the closest root and eventually converges to the further root. Hand-draw the plot and show the iteration process that results in convergence to the further root. Comment on the basins of attraction of the two roots in your plot.

Exercise 6.8 Use Newton–Raphson iterative numerical root finding to per-

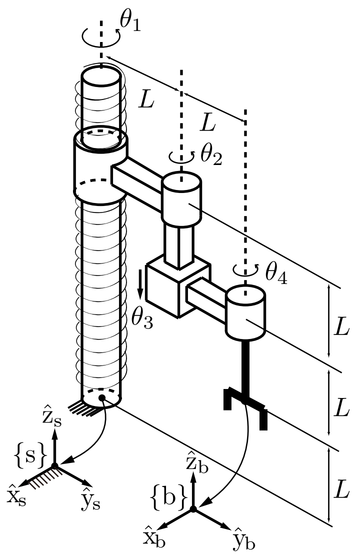


Figure 6.12: An open chain with a screw joint.

form two steps in finding the root of

$$g(x, y) = \begin{bmatrix} x^2 - 4 \\ y^2 - 9 \end{bmatrix}$$

when your initial guess is $(x^0, y^0) = (1, 1)$. Write the general form of the gradient (for any guess (x, y)) and compute the results of the first two iterations. You can do this by hand or write a program. Also, give all the correct roots, not just the one that would be found from your initial guess. How many are there?

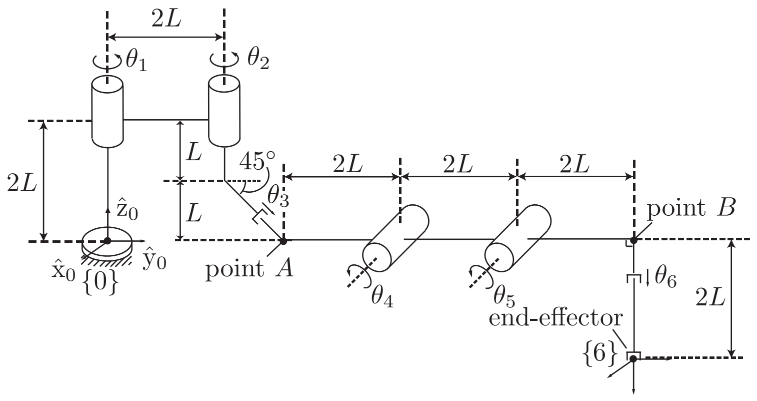
Exercise 6.9 Modify the function `IKinBody` to print out the results of each Newton–Raphson iteration, in a table similar to that for the 2R robot example in Section 6.2. Show the table produced when the initial guess for the 2R robot of Figure 6.8 is $(0, 30°)$ and the goal configuration corresponds to $(90°, 120°)$.

Exercise 6.10 The 3R orthogonal axis wrist mechanism of Figure 6.14 is shown in its zero position, with joint axes 1 and 3 collinear.

(a) Given a desired wrist orientation $R \in SO(3)$, derive an iterative numerical procedure for solving its inverse kinematics.

(b) Perform a single iteration of Newton–Raphson root-finding using body-

(a) da Vinci S Surgical System instrument arm, © 2016 Intuitive Surgical, Inc.



(b) RRPRRP robot at zero position.

Figure 6.13: Surgical robot and kinematic model.

frame numerical inverse kinematics. First write down the forward kinematics and Jacobian for general configurations of the wrist. Then apply your results for the specific case of an initial guess of $\theta_1 = \theta_3 = 0$, $\theta_2 = \pi/6$, with a desired end-effector frame at

$$R = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3).$$

Exercise 6.11 The 3R nonorthogonal chain of Figure 6.15 is shown in its zero position.

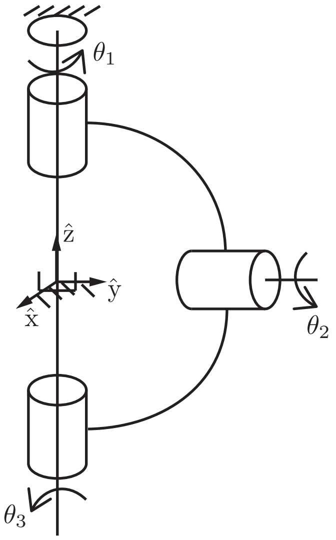


Figure 6.14: A 3R wrist.

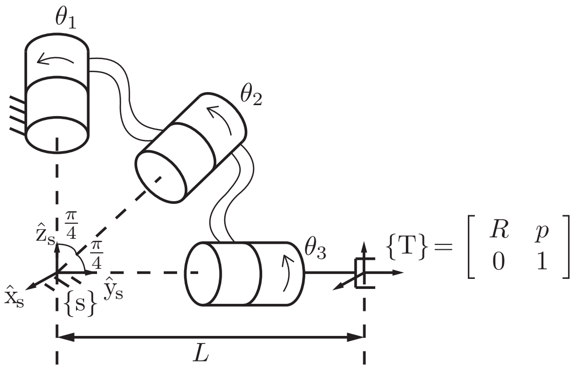


Figure 6.15: A 3R nonorthogonal chain.

(a) Derive a numerical procedure for solving the inverse position kinematics; that is, given some end-effector position p as indicated in the figure, find $(\theta_1, \theta_2, \theta_3)$.

(b) Given an end-effector orientation $R \in SO(3)$, find all inverse kinematic solutions $(\theta_1, \theta_2, \theta_3)$.

Exercise 6.12 Use the function `IKinSpace` to find joint variables θ_d of the UR5 (Section 4.1.2) satisfying

$$T(\theta_d) = T_{sd} = \begin{bmatrix} 0 & 1 & 0 & -0.5 \\ 0 & 0 & -1 & 0.1 \\ -1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The distances are in meters. Use $\epsilon_\omega = 0.001$ rad (0.057°) and $\epsilon_v = 0.0001$ (0.1 mm). For your initial guess θ^0, choose all joint angles as 0.1 rad. If T_{sd} is outside the workspace, or if you find that your initial guess is too far from a solution to converge, you may demonstrate `IKinBody` using another T_{sd}.

Note that numerical inverse kinematics is intended to find a solution close to the initial guess. Since your initial guess is not close to a solution (and remember that there are generally multiple solutions), the procedure may thrash about before finding a solution far from the initial guess. This solution may not respect joint limits. You can post-process the solution so that all joint angles are in the range $[0, 2\pi)$.

Exercise 6.13 Use the function `IKinBody` to find joint variables θ_d of the WAM (Section 4.1.3) satisfying

$$T(\theta_d) = T_{sd} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Distances are in meters. Use $\epsilon_\omega = 0.001$ rad (0.057°) and $\epsilon_v = 0.0001$ (0.1 mm). For your initial guess θ^0, choose all joint angles as 0.1 rad. If T_{sd} is outside the workspace, or if you find that your initial guess is too far from a solution to converge, you may demonstrate `IKinBody` using another T_{sd}.

Note that numerical inverse kinematics is intended to find a solution close to the initial guess. Since your initial guess is not close to a solution (and remember that there are generally multiple solutions), the procedure may thrash about before finding a solution far away from the initial guess. This solution may not respect joint limits. You can post-process the solution so that all joint angles are in the range $[0, 2\pi)$.

Exercise 6.14 The fundamental theorem of linear algebra (FTLA) states that, given a matrix $A \in \mathbb{R}^{m \times n}$,

$$\begin{aligned} \text{null}(A) &= \text{range}(A^{\mathrm{T}})^{\perp}, \\ \text{null}(A^{\mathrm{T}}) &= \text{range}(A)^{\perp}, \end{aligned}$$

where null(A) denotes the null space of A (i.e., the subspace of $\mathbb{R}^n$ of vectors x that satisfy $Ax = 0$), range(A) denotes the range or column space of A (i.e., the subspace of $\mathbb{R}^m$ spanned by the columns of A), and range$(A)^{\perp}$ denotes the orthogonal complement to range(A) (i.e., the set of all vectors in $\mathbb{R}^m$ that are orthogonal to every vector in range(A)).

In this problem you are asked to use the FTLA to prove the existence of Lagrange multipliers (see Appendix D) for the equality-constrained optimization problem. Let $f : \mathbb{R}^n \to \mathbb{R}$, assumed differentiable, be the objective function to be minimized. The vector x must satisfy the equality constraint $g(x) = 0$ for given differentiable $g : \mathbb{R}^n \to \mathbb{R}^m$.

Suppose that x^* is a local minimum. Let $x(t)$ be any arbitrary curve on the surface parametrized implicitly by $g(x) = 0$ (implying that $g(x(t)) = 0$ for all t) such that $x(0) = x^*$. Further, assume that x^* is a regular point of the surface. Taking the time derivative of both sides of $g(x(t)) = 0$ at $t = 0$ then leads to

$$\frac{\partial g}{\partial x}(x^*)\dot{x}(0) = 0. \tag{6.8}$$

At the same time, because $x(0) = x^*$ is a local minimum it follows that $f(x(t))$ (viewed as an objective function in t) has a local minimum at $t = 0$, implying that

$$\frac{d}{dt} f(x(t))\bigg|_{t=0} = \frac{\partial f}{\partial x}(x^*)\dot{x}(0) = 0. \tag{6.9}$$

Since (6.8) and (6.9) must hold for all arbitrary curves $x(t)$ on the surface defined by $g(x) = 0$, use the FTLA to prove the existence of a Lagrange multiplier $\lambda^* \in \mathbb{R}^m$ such that the first-order necessary condition,

$$\nabla f(x^*) + \frac{\partial g}{\partial x}(x^*)^{\mathrm{T}}\lambda^* = 0,$$

holds.

Exercise 6.15

(a) For matrices A, B, C, and D, if A^{-1} exists show that

$$\begin{bmatrix} A & D \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + EG^{-1}F & -EG^{-1} \\ -G^{-1}F & G^{-1} \end{bmatrix},$$

where $G = B - CA^{-1}D$, $E = A^{-1}D$, and $F = CA^{-1}$.

(b) Use the above result to solve for the first-order necessary conditions for the equality-constrained optimization problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^{\mathrm{T}} Q x + c^{\mathrm{T}} x$$

subject to $Hx = b$, where $Q \in \mathbb{R}^{n \times n}$ is symmetric and positive definite and $H \in \mathbb{R}^{m \times n}$ is some matrix of maximal rank m. See Appendix D.