An Introduction to Serial Link Manipulators

Frank Dellaert
Center for Robotics and Intelligent Machines
Georgia Institute of Technology

This note is a tutorial on serial link manipulators, giving my own take on the material in the texts by Murray, Li, and Sastry [4] and Peter Corke [1]. I start with simple planar manipulators with only revolute joints, to develop intuition for how forward kinematics can be implemented by concatenating a chain of rigid transforms. I then show how using exponential maps of twists, while an advanced concept, make describing robot arms much easier and more convenient. After that, it is a simple matter to generalize everything to 3D and include prismatic joints. Finally, I also have an extensive section on inverse kinematics and some Jacobian-based cartesian trajectory control.

1 Serial Link Manipulators

This section describes the basic concepts, closely following [4, 1].

A **serial link manipulator** has several **links**, numbered 0 to n, connected by **joints**, numbered 1 to n. Joint j connects link $j-1$ to link j. We will only consider either a **revolute** joint with a **joint angle** θ_j, or a **prismatic** joint with a **link offset** d_j. More complex joints can be described as combinations of these basic joints.

We can treat both revolute and prismatic joins uniformly by introducing the concept of a **generalized joint coordinate** q_j, and specifying the joint type using a string, e.g., the classical Puma robot is RRRRRR, and the SCARA pick and place robot is RRRP. The vector $q \in Q$ of these generalized joint coordinates is also called the **pose** of the manipulator, where Q is called the **joint space** of the manipulator.

There are two more coordinate frames to consider: the **base frame** S, usually attached directly to link 0, and the **tool frame** T, attached to the end-effector of the robot. The tool frame T moves when the joints move.

2 Planar Manipulators

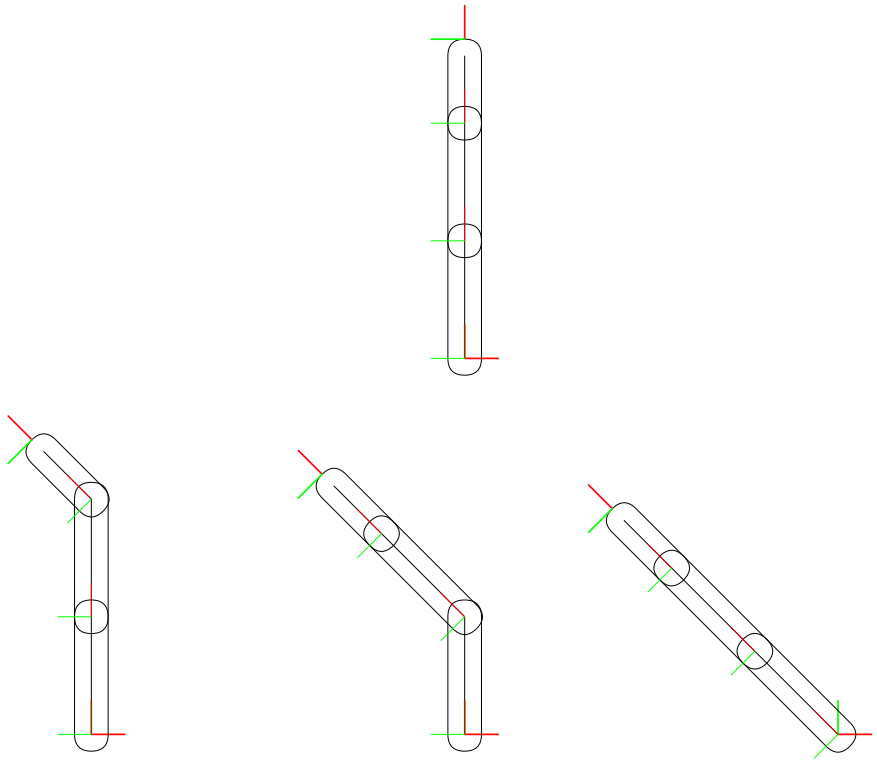

Figure 2.1: Top: the rest state of a planar RRR serial manipulator. Bottom: actuating the three degrees of freedom of this robot, the figure respectively shows θ_3, θ_2, and θ_1.

All essential concepts can be easily developed for 2D or **planar manipulators** with revolute joints only, an example of which is shown in Figure 2.1. The top panel shows the manipulator at rest, along with five 2D coordinate frames: the base frame S, the tool frame T, and one coordinate frame for each of the three links, situated at each joint. Note that at rest, the first link is rotated by $\pi/2$. For this RRR manipulator, the generalized joint coordinates are $q = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T$, and the effect of changing each individual joint angle θ_j is shown at the bottom of the figure.

Later I generalize to 3D and prismatic joints, but for now we only need some plane geometry, which is developed below.

3 Some Geometry for Planar Manipulators

In robot manipulators, the pose of the end-effector or tool is computed by concatenating several rigid transforms. Let p^a be the 2D coordinates of a point in frame A, and p^b the coordinates of the same point in the frame B. We can transform p^a to p^b by a **2D rigid transform** ξ_a^b, which is a rotation followed by a 2D translation,

$$p^b = \xi_a^b \otimes p^a \stackrel{\Delta}{=} R_a^b p^a + t_a^b$$

with $\xi_a^b \stackrel{\Delta}{=} (R_a^b, t_a^b)$, where $R_a^b \in SO(2)$ and $t_a^b \in \mathbb{R}^2$. In all of the above, a subscript A indicates the frame we are transforming *from*, and the superscript B indicates the frame we are transforming *to*. In well-formed equations, subscripts on one symbol match the superscript of the next symbol.

The set of 2D rigid transforms forms a group, where the group operation corresponding to composition of two rigid 2D transforms is be defined as

$$\xi_a^c = \xi_b^c \oplus \xi_a^b = (R_b^c,\, t_b^c) \oplus \left(R_a^b,\, t_a^b\right) \stackrel{\Delta}{=} \left(R_b^c R_a^b,\, R_b^c t_a^b + t_b^c\right) \tag{3.1}$$

The group of rotation-translation pairs ξ with this group operation is called the **special Euclidean group** $SE(2)$. It has an identity element $e = (I, 0)$, and the inverse of a transform $\xi = (R, t)$ is given by $\xi^{-1} = (R^T, -R^T t)$.

2D rigid transforms can be viewed as a subgroup of a general linear group of degree 3, i.e., $SE(2) \subset GL(3)$. This can be done by embedding the rotation and translation into a 3×3 invertible matrix defined as

$$T_a^b = \begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix}$$

With this embedding you can verify that matrix multiplication implements composition, as in Equation 3.1:

$$T_b^c T_a^b = \begin{bmatrix} R_b^c & t_b^c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_b^c R_a^b & R_b^c t_a^b + t_b^c \\ 0 & 1 \end{bmatrix} = T_a^c$$

By similarly embedding 2D points in a three-vector, a 2D rigid transform acting on a point can be implemented by matrix-vector multiplication:

$$\begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^a \\ 1 \end{bmatrix} = \begin{bmatrix} R_a^b p^a + t_a^b \\ 1 \end{bmatrix}$$

In what follows we will work with these transform matrices exclusively.

4 Forward Kinematics

The problem of forward kinematics can now be stated [4]:

> Given a set of generalized joint coordinates $q \in Q$, we wish to determine the pose $T_t^s(q)$ of the tool frame T relative to the base frame S.

We define a link coordinate frame $T_j^s(q)$ for every link j, and we define the link coordinate frame T_0^s to be identical to the base frame S. Since the tool frame T moves with link n, we have

$$T_t^s(q) = T_n^s(q) X_t^n$$

where X_t^n specifies the unchanging pose of the tool T in the frame of link n. The link coordinate frame $T_n^s(q)$ itself can be expressed recursively as

$$T_n^s(q) = T_{n-1}^s(q_1 \dots q_{n-1}) T_n^{n-1}(q_n),$$

finally yielding

$$\boxed{T_t^s(q) = T_1^s(q_1) \dots T_j^{j-1}(q_j) \dots T_n^{n-1}(q_n) X_t^n.} \tag{4.1}$$

5 Describing Serial Manipulators

Equation 4.1 is correct but we need to tie it to the robot's geometry. If we agree on making the link coordinate frame $T_j^s(q)$ coincide with joint axis j and fixed to link j, then the link-to-link transform $T_j^{j-1}(q_j)$ can written as

$$T_j^{j-1}(q_j) = X_j^{j-1} Z_j^j(q_j)$$

where X_j^{j-1} is a fixed transform telling us where joint j is located in the coordinate frame of the previous link, and $Z_j^j(q_j)$ represents the transform at the joint itself, parameterized by the generalized joint coordinate q_j. Substituting this into 4.1 yields

$$\boxed{T_t^s(q) = X_1^s Z_1^1(q_1) \dots X_j^{j-1} Z_j^j(q_j) \dots X_n^{n-1} Z_n^n(q_n) X_t^n} \tag{5.1}$$

For planar revolute joints, the $Z_j^j(q)$ transform can always be made of the form

$$Z_j^j(q) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

possibly with an offset applied to the joint angle. Most often the coordinate frames are chosen such that the X_j^{j-1} transforms are as simple as possible.

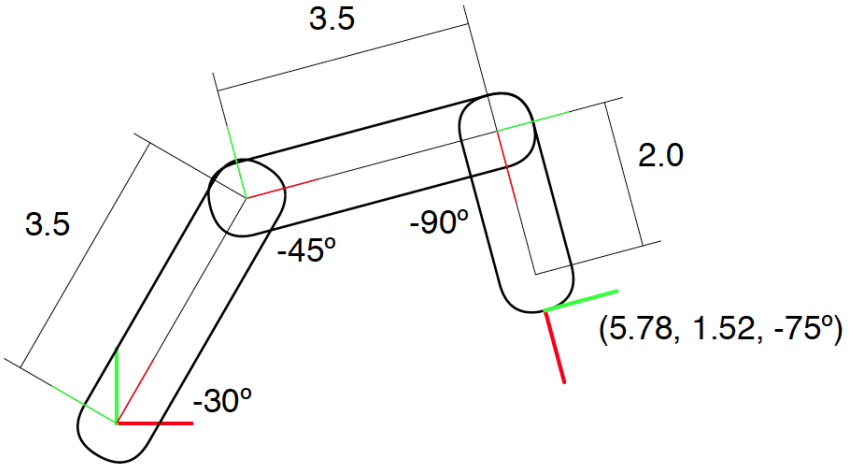

Figure 5.1: Example of simple RRR manipulator with all three joints actuated: $\theta_1 = -30°$, $\theta_2 = -45°$, and $\theta_3 = -90°$, respectively.

As an example, for the simple planar RRR manipulator we have

$$T_t^s(\theta_1, \theta_2, \theta_3) = \left\{X_1^s Z_1^1(\theta_1)\right\} \left\{X_2^1 Z_2^2(\theta_j)\right\} \left\{X_3^2 Z_3^3(\theta_n)\right\} X_t^3$$

I chose the first joint coordinate frames as rotated by 90 degrees, which takes care of the joint angle offset, and chose the tool pose with respect to the third link frame as 2.5 m. along the x-axis:

$$X_1^s = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } X_t^3 = \begin{bmatrix} 1 & 0 & 2.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The next two transforms are simply translations along the link's X axis:

$$X_2^1 = \begin{bmatrix} 1 & 0 & 3.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } X_3^2 = \begin{bmatrix} 1 & 0 & 3.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When multiplied out, we obtain

$$T_t^s(q) = \begin{pmatrix} -\sin\beta & -\cos\beta & -3.5\sin\theta_1 - 3.5\sin\alpha - 2.5\sin\beta \\ \cos\beta & -\sin\beta & 3.5\cos\theta_1 + 3.5\cos\alpha + 3.5\cos\beta \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

with $\alpha = \theta_1 + \theta_2$ and $\beta = \theta_1 + \theta_2 + \theta_3$, the latter being the tool orientation with respect to rest.

6 Product of Exponentials

The above exposition is cumbersome in that it involves a lot of intermediate coordinate frames. Murray et. al. [4] developed a different approach that only involves two coordinate frames: the base frame S and the tool frame T. In addition, it is very easy to determine the parameters corresponding to each joint. The end-result will express the forward kinematics as a **product of exponentials** (POE) of *twists*, which we define below for the planar case.

6.1 2D Twists

The idea behind the product of exponentials formula is to think about the rotation of 2D space around a joint. In particular, we want to find the rigid transform $T_s^s(\theta)$ that takes points in frame S to rotated points in the same frame. If the joint axis happens to be the origin, then the corresponding 3×3 matrix is very easy to write down,

$$T_s^s(\theta) = \begin{bmatrix} R(\theta) & 0 \\ 0 & 1 \end{bmatrix} \tag{6.1}$$

where the 2×2 rotation matrix $R(\theta)$ is given as usual:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

When the joint axis is *not* the origin but some arbitrary point p, let us think of what should happen to a 2D point q. The answer is easy:

$$\begin{bmatrix} q'_x \\ q'_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} q_x - p_x \\ q_y - p_y \end{bmatrix}$$

We can write this elegantly in the language of transform matrices by *conjugating* Equation 6.1 with a translation T_p^s to and from the joint axis p:

$$T_s^s(\theta) = T_p^s \left\{ T_p^p(\theta) \right\} \left(T_p^s \right)^{-1} = \begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix} \tag{6.2}$$

The above is a special case of the **exponential map** $\exp : \mathbb{R}^3 \to SE(2)$, which maps a **2D twist** $\dot{\xi} = (v_x, v_y, \omega)$ to a 2D rigid transform $\exp\left(\dot{\xi}t\right)$. The expression for *arbitrary* twists is is given in the appendix, but it is more complicated than we need. For use in forward kinematics we only work with **unit twists** $\bar{\xi}$, with $\omega = 1$. For revolute joints, the twist that generates the rotation around the joint axis p is the **unit twist** $\bar{\xi} = (p_y, -p_x, 1)$.

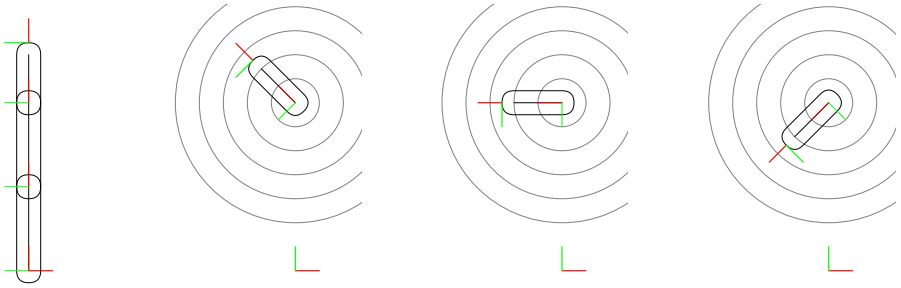

Figure 6.1: The effect of a twist around joint 3 on link 3.

6.2 Example

To illustrate the idea with the planar example, let us look at a single joint, say joint 3. Since the joint axis in rest is at $(0, 7)$, the corresponding twist is $\bar{\xi}_3 = (7, 0, 1)$, with associated transform matrix given by

$$T_s^s(\theta_3) = \exp\left(\bar{\xi}_3\theta_3\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -7 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 7\sin\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 7(1-\cos\theta_3) \\ 0 & 0 & 1 \end{bmatrix}$$

If we view the transform $\exp\left(\bar{\xi}_3\theta_3\right)$ as acting on all points expressed in the base frame S, then it also applies to the entire link 3. This is illustrated in Figure 6.1, where the first panel shows the two last links of the arm in the rest configuration, and the next panels show the effect of the global transform.

In forward kinematics we are mostly interested in the pose of the tool. Hence, if $T_t^s(0)$ is the tool pose for a zero joint angle, then it follows that for a non-zero angle we have

$$T_t^s(\theta_3) = T_s^s\left(\theta_3\right) T_t^s(0) = \exp\left(\bar{\xi}_3\theta_3\right) T_t^s(0)$$

Now we can ask what happens if we move joint 2. Since the joint axis in rest is at $(0, 3.5)$, the corresponding twist is $\bar{\xi}_2 = (3.5, 0, 1)$, with associated transform matrix

$$T_s^s(\theta_2) = \exp\left(\bar{\xi}_2\theta_2\right) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 3.5\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 3.5(1-\cos\theta_2) \\ 0 & 0 & 1 \end{bmatrix}$$

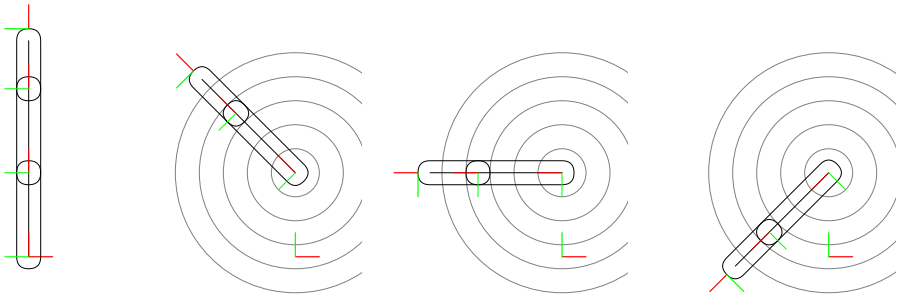

Figure 6.2: The effect of a twist around joint 2, on links 2 and 3.

In Figure 6.2, I show the effect of the twist exponential map $\exp\left(\bar{\xi}_2\theta_2\right)$ on the last two links. Now, since a twist acts on the entire space, if we apply it to the tool frame $T_t^s(\theta_3)$ *after* it has been moved by $\exp\left(\bar{\xi}_3\theta_3\right)$, it stands to reason that the effect of moving the two last joints is given by

$$T_t^s(\theta_2,\theta_3) = \exp\left(\bar{\xi}_2\theta_2\right) T_t^s(\theta_3) = \exp\left(\bar{\xi}_2\theta_2\right)\left\{\exp\left(\bar{\xi}_3\theta_3\right) T_t^s(0)\right\}$$

This formula generalizes in the obvious way for the entire joint configuration,

$$T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right) T_t^s(0) \tag{6.3}$$

where the last remaining transform $\exp\left(\bar{\xi}_1\theta_1\right)$ is of the form 6.1, as the axis of the first joint is the origin. The tool at rest is given by

$$T_t^s(0) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 9.5 \\ 0 & 0 & 1 \end{pmatrix}$$

When multiplied out, we get exactly the same as in Equation 5.3.

6.3 General Case

In general, for any serial manipulator with n joints, we have the following product of exponentials expression for the forward kinematics,

$$\boxed{T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\ldots\exp\left(\bar{\xi}_j\theta_j\right)\ldots\exp\left(\bar{\xi}_n\theta_n\right) T_t^s(0)} \tag{6.4}$$

and, while the left-to-right order has to follow the manipulator structure, the formula above does *not* depend on the order in which the actual joints are actuated.

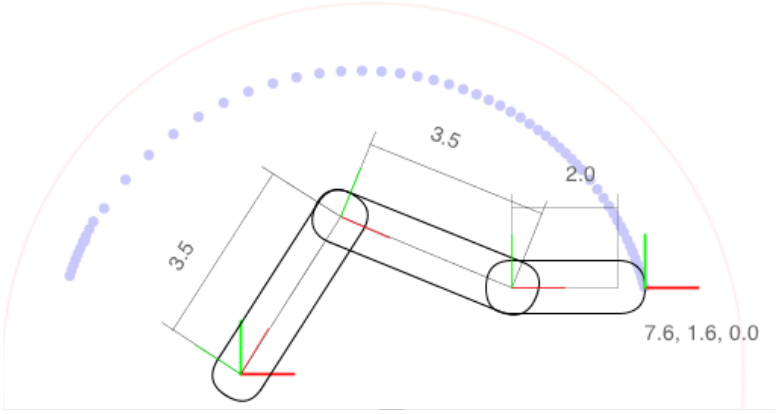

Figure 7.1: In joint-space motion control, the joint angles are linearly interpolated, but this leads to a curved path in Cartesian space.

7 Trajectory Control and the Manipulator Jacobian

Trajectory following is an important capability for manipulator robots, and three main approaches are common: (a) trajectory replay, (b) joint space motion control, and (c) cartesian space motion control. The first relies on an operator to perform the motion first, after which the robot simply replays the sequence, and is akin to motion-capture in movies. Even then, to interpolate between waypoints obtained by robot programming, one of the two other methods is needed.

Joint space motion control is the easiest, and applies linear interpolation or a simple control law in joint space to move from one waypoint to the other, e.g.,

$$q_{t+1} = q_t + K_p(q_d - q_t)$$

where q_t and q_d are the current and desired joint angles, and K_p is a proportional gain. An example of this is shown in Figure 7.1 for the three-link manipulator.

More difficult is **Cartesian motion control**, where we want the robot to follow a well-defined path in Cartesian space, most often a straight line or some interpolating spline. One approach is to calculate an inverse kinematics solution (see Section 8) at many intermediate waypoints and apply joint control again, to get from one to the other. However, there is a method by which we can avoid inverse kinematics altogether. Because we eventually do need to control the joint angles q, the key is to derive a relationship between velocities $\dot{\xi}$ in pose space in response to commanded velocities in joint space $\dot{q}$. This relationship is *locally* linear, and hence we have the following expression at a given configuration q:

$$\dot{\xi} = J(q)\dot{q}$$

The quantity $J(q)$ above is the **manipulator Jacobian**. For planar manipulators, as $\dot{\xi} \in \mathbb{R}^3$, the Jacobian is a $3 \times n$ matrix, with n is the number of joints. Each column of the Jacobian $J(q)$ contains the velocity $\dot{\xi}_j \in \mathbb{R}^3$ corresponding a change in the joint angle q_j only, i.e.,

$$J(q) \triangleq \begin{bmatrix} \dot{\xi}_1 & \dot{\xi}_2 & \dots & \dot{\xi}_n \end{bmatrix} \text{ where } \dot{\xi}_j \triangleq \frac{\partial \xi(q)}{\partial q_j} = \begin{bmatrix} \frac{\partial x(q)}{\partial q_j} \\ \frac{\partial y(q)}{\partial q_j} \\ \frac{\partial \theta(q)}{\partial q_j} \end{bmatrix}$$

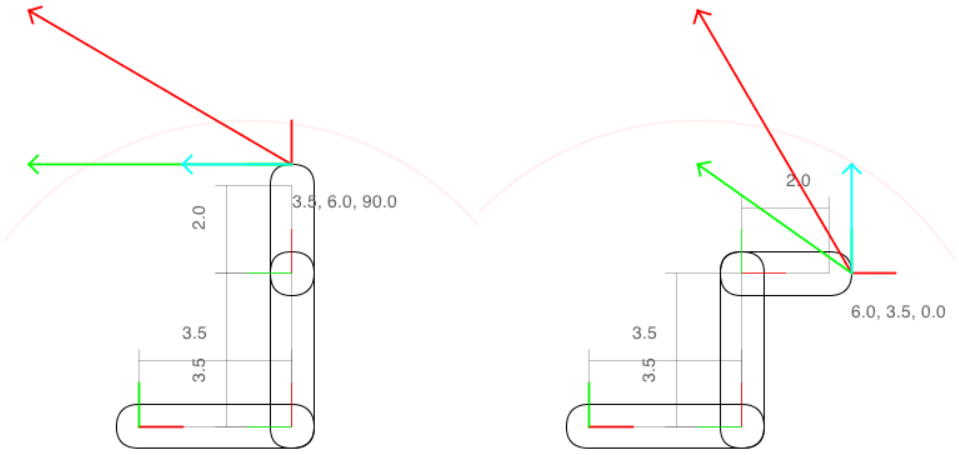

Figure 7.2: The velocities induced by a change in joint angle (red=θ_1, green=θ_2, blue=θ_3), for joint angles $q_{left} = (-90°, 90°, 0°)$ and $q_{right} = (-90°, 90°, -90°)$.

Example. A graphical way to appreciate what a Jacobian means physically is to draw the 2D velocities in Cartesian space. For the three-link planar manipulator example, Figure 7.2 above shows the Jacobian $J(q)$ as a set of three velocities: red for joint 1, green for joint 2, and blue for joint 3. Clearly, these depend on the current joint angles q. The pattern is clear: these velocities are always perpendicular to the vector to the joint axis, and proportional to the distance to the joint axis.
To analytically compute the Jacobian $J(q)$ in this case, we can read off the pose $\xi(q)$ from the forward kinematics equation 5.3 on page 5, yielding

$$\xi(q) = \begin{pmatrix} -3.5 \sin\theta_1 - 3.5 \sin\alpha - 2.5 \sin\beta \\ 3.5 \cos\theta_1 + 3.5 \cos\alpha + 3.5 \cos\beta \\ \beta + \pi/2 \end{pmatrix}$$

where $\alpha = \theta_1 + \theta_2$ and $\beta = \theta_1 + \theta_2 + \theta_3$. Hence, the 3×3 Jacobian $J(q)$ can be computed as

$$\begin{pmatrix} -3.5\cos\theta_1 - 3.5\cos\alpha - 2.5\cos\beta & -3.5\cos\alpha - 2.5\cos\beta & -2.5\cos\beta \\ -3.5\sin\theta_1 - 3.5\sin\alpha - 2.5\sin\beta & -3.5\sin\alpha - 2.5\sin\beta & -2.5\sin\beta \\ 1 & 1 & 1 \end{pmatrix} \tag{7.1}$$

Note that for a planar manipulator, all entries in the third row will always be either 1 or -1: the tool orientation changes at the same rate as any joint you change, possibly in the opposite direction.

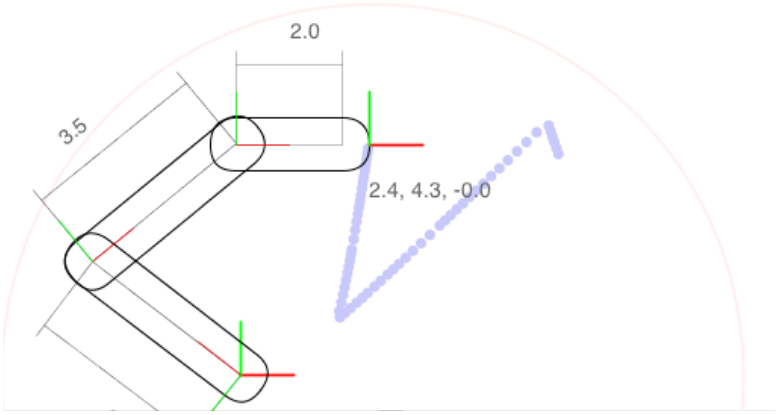

Figure 7.3: Cartesian space motion control, showing the resulting straight trajectories of the end-effector for three successive waypoints.

For a planar manipulator with three joints, i.e., $n = 3$, we can simply invert the 3×3 Jacobian $J(q)$ to calculate the joint space velocities $\dot{q}$ corresponding to a given end-effector velocity $\dot{\xi}$:

$$\dot{q} = J(q)^{-1}\dot{\xi} \tag{7.2}$$

Hence, to achieve a desired trajectory in Cartesian space, we can calculate the desired velocity $\dot{\xi}$ at any given time, calculate the corresponding joint velocities $\dot{q}$ using (7.2), and apply simple proportional control, i.e.,

$$q_{t+1} = q_t + K_p J(q_t)^{-1}(\xi_d - \xi(q_t)),$$

Example. An example of Cartesian motion control with proportional control is shown in Figure 7.3 for the three-link planar robot.

8 Inverse Kinematics

8.1 The IK Problem

Inverse kinematics (IK) is the process of finding joint angles given a desired end-effector pose $T_{desired}$, i.e., solve Equation 10.1 for q:

$$T_t^s(q) = T_{desired}$$

If $T_{desired}$ is outside the workspace of the robot, there is no solution, otherwise there might be a unique solution or multiple solutions.

The essential concepts can be explained by using a two-link planar manipulator. In this simple case, the forward kinematics are given by

$$\begin{cases} x(q) = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ y(q) = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{cases} \tag{8.1}$$

The inverse kinematics problem is then to find the joint angles $q = (\theta_1, \theta_2)$ such that $(x(q), y(q)) = (x_d, y_d)$, the desired end-effector position.

8.2 Closed-Form Solutions

Many industrial manipulators have **closed-form solutions**, and there are several ways to derive these. In this simple 2-link case above, a closed form IK solution is possible, although generally non-unique, within a radius $(l_1 + l_2)$ of the origin. I adapted a solution from John Hollerbach's 2008 lecture notes, for $l_1 = l_2 = L$: we first compute the second joint angle

$$\theta_2(x_d, y_d) = \pm 2 \arctan \sqrt{\frac{(2L)^2}{x_d^2 + y_d^2} - 1} \tag{8.2}$$

after which we compute the first joint angle

$$\theta_1(x_d, y_d, \theta_2) = atan2(y_d, x_d) - atan2\left(L\sin\theta_2, L(1 + \cos\theta_2)\right) - \frac{\pi}{2} \tag{8.3}$$

Note that Equation 8.2 is not defined if $x_d^2 + y_d^2 > (2L)^2$, which corresponds to a desired position that is out-of-range.

To extend this to the three-link example, we add a desired tool orientation, i.e., we have a desired 2D pose $\xi_d = (x'_d, y'_d, \theta'_d)$, where I used primes to distinguish from the 2-link example. Note from the forward kinematics Equation 5.3 that

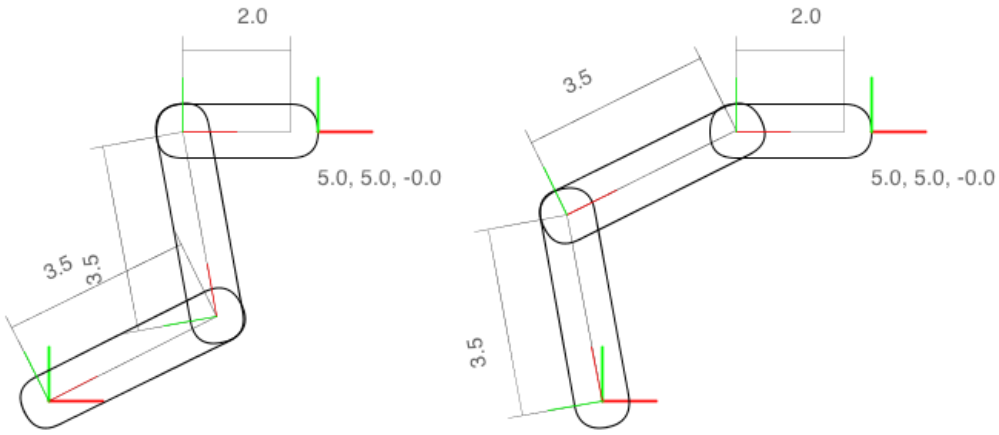

Figure 8.1: Two IK solutions for the planar three-link arm, for tool pose $(x'_d, y'_d, \theta'_d) = (5, 5, 0)$, with joint angles $q = (-63.6°, 74.0°, -100.4°)$ and $q = (10.4°, -74.0°, -26.4°)$, corresponding respectively to choosing a positive and negative sign in Equation 8.2.

$\beta = \theta'_d - \pi/2$, and we can adapt Equations8.3 and 8.2 to accommodate for the joint 2 axis offset:

$$x_d = x'_d + 2.5 \sin\beta \text{ and } y_d = y'_d - 2.5 \cos\beta$$

$$\theta'_2 = (x_d, y_d) \text{ and } \theta'_1 = (x_d, y_d, \theta'_2)$$

Then, the wrist joint angle is easily computed as

$$\theta'_3 = \theta'_d - \theta'_1 - \theta'_2 - \pi/2$$

Figure 8.1 shows two examples corresponding to choosing a different sign in Equation 8.2 for the same desired pose $(x'_d, y'_d, \theta'_d) = (5, 5, 0)$, illustrating the fact that inverse kinematics is not a one-on-one mapping, but that multiple solutions might exist, even for a non-redundant manipulator.

8.3 Iterative Methods

As discussed, many industrial manipulators have closed-form solutions, but they are still an area of active research and general solutions are as yet elusive. There exist, however, iterative methods to solve the IK problem.

One approach is to find the joint angles q that minimize the error between desired end-effector pose and computed end-effector pose. For example, for the

simple 2-link arm we would try to minimize

$$E(q) \triangleq \frac{1}{2}\|p_d - p(q)\|^2 = \frac{1}{2}\left(x_d - x(q)\right)^2 + \frac{1}{2}\left(y_d - y(q)\right)^2 \tag{8.4}$$

where $p_d \in \mathbb{R}^2$ and $p(q) \in \mathbb{R}^2$ are the desired and computed 2D position. However, this is a non-linear minimization problem, as $x(q)$ and $y(q)$ are typically non-linear functions of the joint-angles (at least for rotational joints). Linear least-squares problems are easier to solve, which motivates us to start with an initial guess q for the joint angles, and to linearize the forward kinematics around this pose,

$$p(q + \delta q) \approx p(q) + J(q)\delta q \tag{8.5}$$

where $J(q)$ is once again the manipulator Jacobian, derived in Section 7. For the simple 2-link planar manipulator we can only hope achieve a desired position $p_d \in \mathbb{R}^2$, and hence the Jacobian $J(q)$ is only a 2×2 matrix, easily calculated as

$$\begin{aligned} J(q) &= \begin{bmatrix} \frac{\partial x(q)}{\partial \theta_1} & \frac{\partial x(q)}{\partial \theta_2} \\ \frac{\partial y(q)}{\partial \theta_1} & \frac{\partial y(q)}{\partial \theta_2} \end{bmatrix} \\ &= \begin{bmatrix} -y(q) & -l_2 \sin\left(\theta_1 + \theta_2\right) \\ x(q) & l_2 \cos\left(\theta_1 + \theta_2\right) \end{bmatrix} \end{aligned}$$

Substituting the approximation 8.5 into the objective 8.4 we obtain

$$E(q + \delta q) \approx \frac{1}{2}\| \left(p_d - p(q)\right) - J\delta q\|^2 \tag{8.6}$$

where the dependance of J on q is implied, for notational simplicity. The above says that we can make the position error $e(q) \triangleq p_d - p(q)$ go to zero by calculating a change δq in joint angles, such that

$$J\delta q = p_d - p(q)$$

For a non-redundant manipulator the Jacobian $J(q)$ is square, and its inverse generally exists (except at the boundaries of the workspace). Hence, we could try to invert it immediately:

$$\delta q = J^{-1}\left(p_d - p(q)\right) \tag{8.7}$$

However, because the forward kinematics are non-linear, we might have to iterate this a few times, and the process might in fact diverge.

8.4 Damped Least-Squares

A safer approach is to impose some penalty for taking steps δq that are too large, which can be done by adding a term to the objective function 8.6:

$$E(q+\delta q) \approx \frac{1}{2}\| \left(p_d - p(q)\right) - J\delta q\|^2 + \frac{1}{2}\|\lambda \delta q\|^2 \tag{8.8}$$

This can be solved for δq by setting the derivative of E in Equation 8.8 to 0,

$$-J^T \left(\left(p_d - p(q)\right) - J\delta q\right) + \lambda^2 \delta q = 0$$

which, after simply re-arranging, leads to a damped least-squares iteration:

$$\delta q = \left(J^T J + \lambda^2 I\right)^{-1} J^T \left(p_d - p(q)\right) \tag{8.9}$$

The value of λ is chosen as to make the iterative process converge, and can even be increased automatically when a low value is seen to lead to divergence. On the other hand, too high a value might lead to slow convergence.

One strategy is to proceed very cautiously, i.e., make λ very large, in which case Equation 8.9 essentially becomes gradient descent:

$$\delta q = \lambda^{-2} J^T \left(p_d - p(q)\right) \tag{8.10}$$

Because it only involves J^T, this is also called the **transpose Jacobian method**, but it has the disadvantage of converging very slowly, see Fig. 9.1.

A method that picks λ automatically is the **Levenberg-Marquardt** method: start with a high λ, and then reduce it by a constant factor at every iteration until the error goes up instead of down: in that case undo the change in λ and try again.

8.5 Iterative IK Methods Summary

In summary, all iterative inverse kinematics approaches share the same structure:

Algorithm 1 The basic iterative IK algorithm.

1: **function** ITERATIVEIK(x_d)
2: $q \leftarrow q_0$ ▷ Guess an initial value for joint angles
3: **while** $E(q) \neq 0$ **do** ▷ While not converged
4: $e \leftarrow p_d - p(q)$ ▷ Calculate error between desired and computed pose
5: Calculate $\delta q = f(J(q), e)$ ▷ Using Eqn. (8.7), (8.9), (8.10)
6: $q \leftarrow q + \delta q$ ▷ Update the joint angles q
7: **return** q ▷ Joint angles yielding p_d

9 Redundant Manipulators

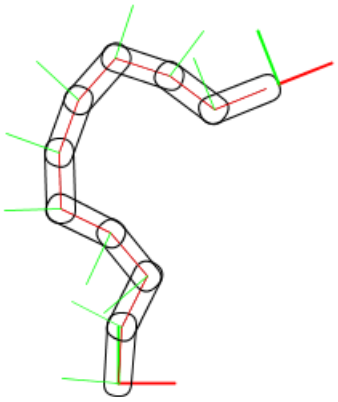

Figure 9.1: A highly redundant planar manipulator.

One advantage of the least-squares approaches above is that they also work for $n > m$, i.e., for **redundant manipulators**. An example of a highly redundant planar manipulator is shown in Figure 9.1. In this case the inverse in Equation 8.7 does not exists (as J is non-square), but we can use the **pseudo-inverse** $J^\dagger$,

$$\delta q = J^\dagger \left(p_d - p(q) \right) \tag{9.1}$$

where $J^\dagger \triangleq J^T \left(J J^T \right)^{-1}$ for $n > m$.

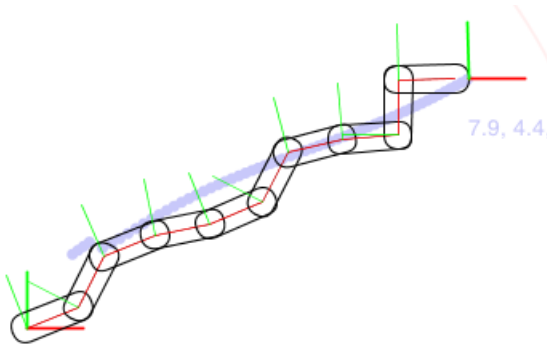

Figure 9.2: Convergence of iterative IK with the transpose Jacobian method.

Note that for redundant manipulators the pseudo-inverse is equivalent to damped least squares with $\lambda = 0$. Of course, even in the redundant case we can take λ to

be non-zero and just apply gradient descent (8.9), or make λ large in which case we recover the transpose method (8.10). An example of the latter is shown as a trajectory in Figure 9.2.

In a redundant manipulator there are typically an infinite number of ways to attain a desired end-effector pose. The methods above arbitrarily pick one of the solutions. However, the extra degrees of freedom might be put to other uses, as well: for example, we might want to favor configurations that require less energy to maintain, or avoid singularities, or even - in the case of using IK for animation - follow a certain style [2]. This can be done by adding an additional, user-defined penalty term $E_{user}(q)$ to the error function that penalizes certain joint configurations and favors others:

$$E(q) \stackrel{\Delta}{=}^2 \frac{1}{2}\|p_d - p(q)\|^2 + E_{user}(q)$$

As long as the derivative of $E_{user}(q)$ is available, it is easy to incorporate this extra information.

A simple example is to make the joint angles as small as possible, i.e., penalizing a deviation from the rest state, leading to

$$E(q) \stackrel{\Delta}{=}^2 \frac{1}{2}\|p_d - p(q)\|^2 + \frac{1}{2}\|\beta q\|^2$$

After linearizing, we have

$$E(q + \delta q) \approx \frac{1}{2}\| \left(p_d - p(q)\right) - J\delta q\|^2 + \frac{1}{2}\|\beta(q + \delta q)\|^2$$

which yields the following update,

$$\delta q = \left(J^T J + \beta^2 I\right)^{-1} \left(J^T \left(p_d - p(q)\right) - \beta^2 q\right)$$

which looks very much like the damped least-squares iteration (8.9), except that now there is an extra error term that drives the joint angles to zero.

Finally, a user could also impose hard equality or inequality constraints. One such constraint is that the robot should never self-intersect or collide with objects in its environment. In dynamic environments, then, these constraints define a **configuration space** that can change over time. We then get into the realm of fully fledged **motion planning**, which is a prolific and active area of research.

10 Spatial Manipulators

The story above generalizes almost entirely to three dimensions. Below we first discuss 3D rotations and rigid transforms, and then generalize the three forward kinematics equations we discussed so far.

10.1 Rotations in 3D

Rotating a point in 3D around the origin from a frame a to a rotated frame b can be done by multiplying with a 3×3 orthonormal rotation matrix

$$p^a = R_a^b p^a$$

where the indices on R_a^b indicate the source and destination frames. The columns of R_a^b represent the axes of frame a in the b coordinate frame:

$$R_a^b = \begin{bmatrix} \hat{x}_a^b & \hat{y}_a^b & \hat{z}_a^b \end{bmatrix}$$

The 3D rotations together with composition constitute the **special orthogonal group** $SO(3)$. It is made up of all 3×3 orthonormal matrices with determinant 1, with matrix multiplication implementing composition. However, *3D rotations do not commute*, i.e., in general $R_2 R_1 \neq R_1 R_2$.

10.2 3D Rigid transforms

A point in 3D can be transformed by a 3D rigid transform, which is a 3D rotation followed by a 3D translation,

$$p^b = R_a^b p^a + t_a^b$$

with $R_a^b \in SO(3)$ and $t_a^b \in \mathbb{R}^3$. We denote this transform by $T_a^b \triangleq \left(R_a^b,\, t_a^b\right)$. The **special Euclidean group** $SE(3)$, with the group operation defined similarly as in Equation 3.1. Moreover, the group $SE(3)$ is a subgroup of a general linear group $GL(4)$ of degree 4, by embedding the rotation and translation into a 4×4 invertible matrix defined as

$$T_a^b = \begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix}$$

Again, by embedding 3D points in a four-vector, a 3D rigid transform acting on a point can be implemented by matrix-vector multiplication:

$$\begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^a \\ 1 \end{bmatrix} = \begin{bmatrix} R_a^b p^a + t_a^b \\ 1 \end{bmatrix}$$

10.3 Kinematic Chains in Three Dimensions

In three dimensions the matrices are now 4×4, but exactly the same expressions are use to describe a kinematic chain:

$$T_t^s(q) = T_1^s(q_1) \dots T_j^{j-1}(q_j) \dots T_n^{n-1}(q_n) X_t^n. \tag{10.1}$$

and to describe any serial manipulator we can again alternate fixed link transforms X_j^{j-1} and parameterized joint transforms $Z_j^j(q_j)$:

$$T_t^s(q) = X_1^s Z_1^1(q_1) \dots X_j^{j-1} Z_j^j(q_j) \dots X_n^{n-1} Z_n^n(q_n) X_t^n \tag{10.2}$$

In 3D we typically obey the convention that the axis of rotation is chosen to be the Z-axis, hence the suggestive naming of the corresponding matrix parameterized by a joint angle. We can now also introduce **prismatic joints**, which are typically parameterized as translations *along* the Z-axis.

10.4 Product of Exponentials in 3D

To generalize the product of exponentials formula, we need the concept of a 3D **unit twist** $\bar{\xi}$, which consist of an **axis of rotation** $\bar{\omega}$ combined with a **linear motion vector** $\bar{v}$, which defines a rotational or translational motion around the joint.

For a *prismatic* joint, the twist is simply $\bar{\xi} = (0, \bar{v})$, with $\bar{v}$ a specifying the direction of motion, and the exponential map is just a translation:

$$\exp\left(\bar{\xi} t\right) = \begin{bmatrix} I & \bar{v}t \\ 0 & 1 \end{bmatrix}$$

For a *revolute* joint, the twist is given by $\bar{\xi} = (\bar{\omega}, p \times \bar{\omega})$, where $\bar{\omega}$ is a unit vector specifying the axis of rotation, and p is *any* point on the joint axis. In this case the exponential map simplifies to

$$\exp\left(\bar{\xi}\theta\right) = \begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\bar{\omega}\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix}$$

The general formula for the rotation matrix $R(\bar{\omega}\theta)$ is given in the appendix, but is easy whenever the rotation axis $\bar{\omega}$ is aligned with a coordinate axis. For example, for a rotation around the Z-axis, i.e., $\omega = [0, 0, 1]^T$ we have

$$R(\bar{\omega}\theta) = R([0, 0, \theta]^T) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and similarly, for respectively the X-axis and Y-axis, we have

$$R([\theta,0,0]^T) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R([0,\theta,0]^T) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Finally, we obtain the same product of exponentials as in the planar case,

$$T_t^s(q) = \exp\left(\bar{\xi}_1 q_1\right) \ldots \exp\left(\bar{\xi}_j q_j\right) \ldots \exp\left(\bar{\xi}_n q_n\right) T_t^s(0) \tag{10.3}$$

10.5 Detailed Example: The Pincher Robot

Figure 10.1: Pincher robot at rest, with all joint angles at 0. The chosen base frame S and tool frame T are shown as RGB coordinate frames.

The Pincher robot is shown in Figure 10.1 in its rest configuration. I chose to put the base frame S at the intersection of the vertical joint 1 axis and the horizontal

joint 2 axis, which makes the transforms for the first two joint axes easy, with twists $\bar{\xi}_1 = (0, 0, 1, 0, 0, 0)$ and $\bar{\xi}_2 = (1, 0, 0, 0, 0, 0)$, respectively:

$$e^{\bar{\xi}_1\theta_1} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } e^{\bar{\xi}_2\theta_2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 & 0 \\ 0 & \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note that the joint angles are measured from the configuration at rest, and positive joint angles will make the arm lean backwards.

The third joint axis, at rest (all servos at 512) is 10.5 cm above the second one. We can just conjugate a rotation around the x-axis, corresponding to the twist $\bar{\xi}_3 = (1, 0, 0, 0, 10.5, 0)$:

$$e^{\bar{\xi}_3\theta_3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_3 & -\sin\theta_3 & 10.5\sin\theta_3 \\ 0 & \sin(\theta_3) & \cos(\theta_3) & 10.5(1-\cos\theta_3) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The fourth joint is the same, except it is higher, with twist $\bar{\xi}_4 = (1, 0, 0, 0, 21, 0)$:

$$e^{\bar{\xi}_4\theta_4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_4 & -\sin\theta_4 & 21\sin\theta_4 \\ 0 & \sin\theta_4 & \cos\theta_4 & 21(1-\cos\theta_4) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finally, the fifth joint is the end-effector and is not included in the inverse kinematics. The tool frame, at rest, is 6.5 cm above the fourth joint:

$$T_t^s(0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 27.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplying all these together, we get the forward kinematics as follows:

$$T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right)\exp\left(\bar{\xi}_4\theta_4\right)T_t^s(0) =$$

$$\begin{pmatrix} \cos\theta_1 & -\sin\theta_1\cos\beta & \sin\theta_1\sin\beta & \frac{1}{2}\sin\theta_1\left(21\sin\theta_2 + 21\sin\alpha + 13\sin\beta\right) \\ \sin\theta_1 & \cos\theta_1\cos\beta & -\cos\theta_1\sin\beta & -\frac{1}{2}\cos\theta_1\left(21\sin\theta_2 + 21\sin\alpha + 13\sin\beta\right) \\ 0 & \sin\beta & \cos\beta & \frac{1}{2}\left(21\cos\theta_2 + 21\cos\alpha + 13\cos\beta\right) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $\alpha = \theta_2 + \theta_3$ and $\beta = \theta_2 + \theta_3 + \theta_4$. It is easy to see that the first joint angle θ_1 rotates the entire arm around the vertical, and that the other angles dictate the kinematics in that rotated frame. In fact, not coincidentally, when setting θ_1 to zero we can recognize exactly the same structure as the three-link kinematics chain from Equation 5.3, in the Y-Z plane:

$$T_t^s(0, \theta_2, \theta_3, \theta_4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & -\frac{1}{2}\left(21\sin\theta_2 + 21\sin\alpha + 13\sin\beta\right) \\ 0 & \sin\beta & \cos\beta & \frac{1}{2}\left(21\cos\theta_2 + 21\cos\alpha + 13\cos\beta\right) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is clear from this that, with three joints in this plane and the rotation around joint 1, we can reach any 3D pose within the workspace. However, we have no way of rotating the tool around its Z-axis.

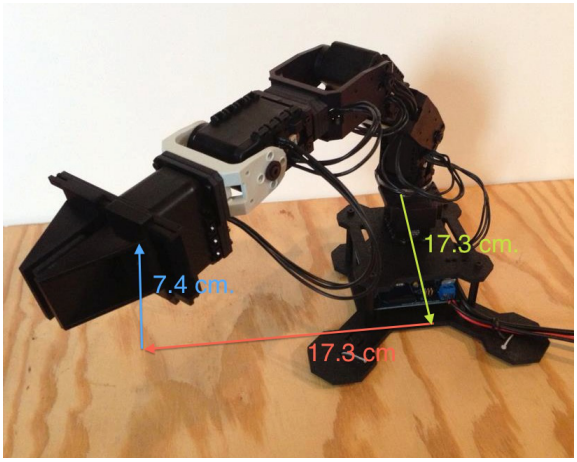

Figure 10.2: The pose of the robot with joint angles $q = (-45°, -45°, -45°, 0)$.

To test the FK, let us plug in some angles:

$$T_t^s(-45°, -45°, -45°, 0) = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 17.3 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 17.3 \\ 0 & -1 & 0 & 7.4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It can be verified in figure 10.2 (and I verified it in reality) that the tool is indeed horizontal now, rotated 45 degrees to the left, and is at position $(17.3, 17.3, 7.4)$ with respect to the chosen base frame.

A Appendix: Denavit-Hartenberg Conventions

No introduction to serial manipulators is complete without mentioning the **Denavit-Hartenberg convention**, which is a particular choice of coordinate frames to make equation 10.2 as simple as possible. In particular, as suggested by the alternation of matrices named X and Z, we ensure that

1. all joint axes are aligned with the Z-axis, and the corresponding transform is parameterized by two parameters, a rotation θ around Z and a displacement d along Z;

2. the X-axis is chosen to be the **common perpendicular** between two successive joint axes, and the link geometry is described by two parameters, a rotation α around X and a displacement a along X.

It might be surprising that only four parameters (θ, d, α, a) are needed to specify the location of one frame relative to another. However, these frames are special since they have two independent conditions imposed: the X-axis intersects the next Z-axis, and is perpendicular to it [3].

Subject to these two constraints, there are two popular variants in use, the proximal and distal variants, that differ on where they put the coordinate frame on the link. In the **distal** variant, the link coordinate frame $T_j^s(q)$ is made to coincide with joint axis $j+1$, and link frame n is defined to be identical to the tool frame. This convention is a bit awkward to work with.

In the simpler, **proximal** variant, also denoted the **modified Denavit-Hartenberg convention** in [1], the link coordinate frame $T_j^s(q)$ is made to coincide with joint axis j, and the transform $T_j^{j-1}(q_j)$ between links is written as:

$$T_j^{j-1}(q_j) = X_j^{j-1}(\alpha_{j-1}, a_{j-1}) Z_j^j(\theta_j, d_j)$$

where q_j is either θ_j for a revolute joint, or to d_j for a prismatic joint, and

$$X_j^{j-1}(\alpha_{j-1}, a_{j-1}) = T_{Rx}(\alpha_{j-1}) T_x(a_{j-1}) \qquad Z_j^j(\theta_j, d_j) = T_{Rz}(\theta_j) T_z(d_j)$$

B Appendix: General Twist Formulas

B.1 2D Twists

A **2D twist**[1] is the derivative of a 2D rigid transform, $\dot{\xi} = \left(\dot{x}, \dot{y}, \dot{\theta}\right)$. The first two components make up the linear velocity $v \in \mathbb{R}^2$, and the last component is the

[1] For purists: in contrast to [4], I define a *twist* $\dot{\xi}$ as any element in the Lie algebra $\mathfrak{se}(2)$ or $\mathfrak{se}(3)$, and use the term *unit twist* $\bar{\xi}$ for the case where $\omega = 1$ (or $\|\omega\| = 1$, in $\mathfrak{se}(3)$).

angular velocity $\omega \in \mathbb{R}$. Hence, we also frequently write $\dot{\xi} = (v, \omega)$. There are two complimentary ways of deriving the **exponential map** $\exp : \mathbb{R}^3 \rightarrow SE(2)$ that maps a **2D twist** $\dot{\xi} = (v, \omega)$ to a 2D rigid transform.

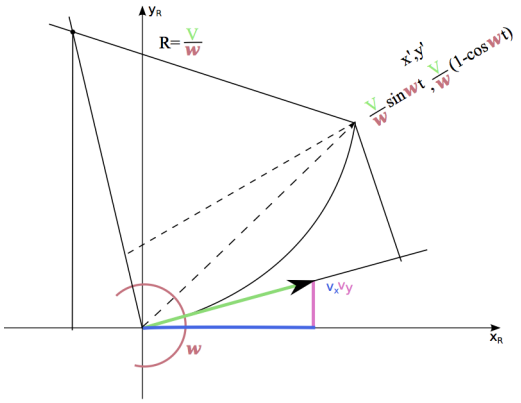

Figure B.1: Integrating a constant twist forward traces out a circular arc.

First, a pose undergoing a constant twist $\dot{\xi}$ traces out a circular trajectory with radius $R = \|v\|/\omega$, as illustrated in Figure B.1. Starting from the identity e, after time t we obtain, as can be worked out from the figure, the 2D pose $\xi(t) \in SE(2)$:

$$\exp\left(\dot{\xi}t\right) = \left(\begin{bmatrix} \cos\omega t & -\sin\omega t \\ \sin\omega t & \cos\omega t \end{bmatrix}, \frac{1}{\omega} \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} \begin{bmatrix} \sin\omega t \\ 1-\cos\omega t \end{bmatrix} \right) \tag{B.1}$$

The second way uses conjugation, as explained in the text, and makes use of the fact that the instantaneous axis of rotation can be found for arbitrary twists $\dot{\xi} = (v, \omega)$ as $p = v^{\perp}/\omega$, where the $\perp$ symbol denotes the orthogonal vector

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix}^{\perp} \triangleq \begin{bmatrix} -v_y \\ v_x \end{bmatrix}$$

We then simply conjugate a rotation at the origin with a translation to p:

$$\exp\left(\dot{\xi}t\right) = \begin{bmatrix} I & v^{\perp}/\omega \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\omega t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -v^{\perp}/\omega \\ 0 & 1 \end{bmatrix} \tag{B.2}$$

where the 2×2 rotation matrix $R(\theta)$ is given as usual:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

You can verify that the above and the circular arc expression B.1 are identical.

B.2 Twists in 3D

In 3D, a **twist**[2] is given by an angular velocity $\omega \in \mathbb{R}^3$ and a linear velocity $v \in \mathbb{R}^3$, collected in a column vector $\dot{\xi} \in \mathbb{R}^6$:

$$\dot{\xi} \stackrel{\Delta}{=} \begin{bmatrix} \omega \\ v \end{bmatrix}$$

For $\|\omega\| \neq 0$, the **exponential map** $\exp : \mathbb{R}^6 \rightarrow SE(3)$ is given by

$$\exp\left(\dot{\xi}t\right) = \begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\omega,t) & \frac{\omega}{\|\omega\|}ht \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix} \tag{B.3}$$

where $p = (\omega \times v)/\|\omega\|$ is a point on the **axis** of the **screw motion** generated by the twist, the scalar $h \stackrel{\Delta}{=} \omega^T v/\|\omega\|$ is its **pitch,** and the rotation matrix $R(t)$ is given by Rodrigues' formula,

$$R(\omega,t) = I + [\omega]_\times \frac{\sin(\|\omega\|t)}{\|\omega\|} + [\omega]_\times^2 \frac{1-\cos(\|\omega\|t)}{\|\omega\|^2}$$

which calculates the 3D rotation matrix associated with spinning for a time t with angular velocity ω, and $[\omega]_\times$ is a the skew symmetric matrix of ω.

The above might look more complicated than the 2D case, but is really but a simple generalization of a circular arc in 2D to a screw motion in 3D.

C Appendix: Jacobians and Exponential Twists

Calculating the Jacobians of complex or highly redundant manipulators can be fairly involved, but once again the product of exponential twists comes to the rescue. Let us examine the forward kinematics of the three-link planar manipulator again, i.e., Equation 6.3:

$$T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right)T_t^s(0)$$

The partial derivative with respect to the first joint angle is

$$\begin{aligned} \frac{\partial T_t^s(q)}{\partial\theta_1} &= \frac{\partial \exp\left(\bar{\xi}_1\theta_1\right)}{\partial\theta_1}\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right)T_t^s(0) \\ &= \hat{\xi}_1 T_t^s(q) \end{aligned}$$

[2]Note we follow a different convention from [4] in that we reserve the first three components for rotation, and the last three for translation.

where $\hat{\xi}$ is a 4×4 matrix (the element of the Lie algebra $\mathfrak{se}(2)$ corresponding to $\bar{\xi}_1$), defined for any twist $\dot{\xi} = (\omega, v)$ as:

$$\hat{\xi} = \widehat{\begin{bmatrix} \omega \\ v \end{bmatrix}} = \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix}$$

Similarly, with respect to the second joint angle we have

$$\begin{aligned} \frac{\partial T_t^s(q)}{\partial \theta_2} &= \exp\left(\bar{\xi}_1 \theta_1\right) \frac{\partial \exp\left(\bar{\xi}_2 \theta_2\right)}{\partial \theta_2} \exp\left(\bar{\xi}_3 \theta_3\right) T_t^s(0) \\ &= \exp\left(\bar{\xi}_1 \theta_1\right) \hat{\xi}_2 \exp\left(\bar{\xi}_2 \theta_2\right) \exp\left(\bar{\xi}_3 \theta_3\right) T_t^s(0) \\ &= \exp\left(\bar{\xi}_1 \theta_1\right) \hat{\xi}_2 \exp\left(-\bar{\xi}_1 \theta_1\right) T_t^s(q) \end{aligned}$$

In the last line we again see a conjugation: the derivative is given by $\hat{\xi}_2$, but acting in the link frame $\exp\left(\bar{\xi}_1 \theta_1\right)$. It is easy to prove that

$$T \hat{\xi} T^{-1} = \widehat{Ad_T \bar{\xi}}$$

where Ad_T is the **adjoint transformation** associated with $T = (R, t)$:

$$Ad_T \bar{\xi} = \begin{bmatrix} R & \\ [t]_\times R & R \end{bmatrix} \bar{\xi}$$

Hence, we can write

$$\frac{\partial T_t^s(q)}{\partial \theta_2} = \widehat{Ad_{T_1^s} \bar{\xi}_2} T_t^s(q), \text{ and also } \frac{\partial T_t^s(q)}{\partial \theta_3} = \widehat{Ad_{T_2^s} \bar{\xi}_3} T_t^s(q).$$

Generalizing, we have

$$\hat{\xi}_i' \triangleq \widehat{Ad_{T_{i-1}^s} \bar{\xi}_i} = \frac{\partial T_t^s(q)}{\partial q_i} T_t^s(q)^{-1}$$

Motivated by this, Murray et. al. [4, p. 116] define a **spatial manipulator Jacobian** which consists of n twists transformed by the appropriate adjoints,

$$J_{st}^s(q) = \begin{bmatrix} \bar{\xi}_1' & \bar{\xi}_2' & \dots & \bar{\xi}_n' \end{bmatrix} = \begin{bmatrix} \bar{\xi}_1 & Ad_{T_1^s} \bar{\xi}_2 & \dots & Ad_{T_{n-1}^s} \bar{\xi}_n \end{bmatrix}$$

These twists are exactly the twists we would use in the products of exponentials formula for forward kinematics if we would define them in the configuration q.

The Jacobian $J^s_{st}(q)$ is related but not identical to the Jacobians $J(q)$ defined in Section 7. In general, the derivative of the end-effector pose transform $T^s_t(q)$ as a result of the twist $\bar{\xi}'_j$ is given by

$$\begin{aligned}\frac{\partial T^s_t(q)}{\partial q_j} = \hat{\xi}'_j T^s_t(q) &= \begin{bmatrix} [\omega'_j]_\times & v'_j \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R(q) & p(q) \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} [\omega'_j]_\times R(q) & \omega'_j \times p(q) + v'_j \\ 0 & 0 \end{bmatrix}\end{aligned}$$

Example. For the planar three-link manipulator example we can simply read off the twists $\bar{\xi}'_j = (p_y, -p_x, 1)$ corresponding to each joint axis p, for any configuration $q = (\theta_1, \theta_2, \theta_3)$:

$$\begin{bmatrix} \bar{\xi}'_1 & \bar{\xi}'_2 & \bar{\xi}'_3 \end{bmatrix} = \begin{pmatrix} 0 & 3.5\cos\theta_1 & 3.5\cos\theta_1 + 3.5\cos\alpha \\ 0 & 3.5\sin\theta_1 & 3.5\sin\theta_1 + 3.5\sin\alpha \\ 1 & 1 & 1 \end{pmatrix}$$

For the planar three-link manipulator example we have

$$\omega'_i \times p(q) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} x(q) \\ y(q) \\ 1 \end{pmatrix} = \begin{pmatrix} -3.5\cos\theta_1 - 3.5\cos\alpha - 3.5\cos\beta \\ -3.5\sin\theta_1 - 3.5\sin\alpha - 2.5\sin\beta \\ 0 \end{pmatrix}$$

and we recover the analytically computed Jacobian $J(q)$. For example , For the second column we have (for the translation part only),

$$\begin{aligned} J_{1:2,2}(q) &= \begin{pmatrix} -3.5\cos\theta_1 - 3.5\cos\alpha - 3.5\cos\beta \\ -3.5\sin\theta_1 - 3.5\sin\alpha - 2.5\sin\beta \end{pmatrix} + \begin{pmatrix} 3.5\cos\theta_1 \\ 3.5\sin\theta_1 \end{pmatrix} \\ &= \begin{pmatrix} -3.5\cos\alpha - 3.5\cos\beta \\ -3.5\sin\alpha - 2.5\sin\beta \end{pmatrix} \end{aligned}$$

which is the linear velocity of the end-effector induced by the second twist $\bar{\xi}'_2$. Compare this with the analytically derived Jacobian in Equation 7.1 on page 11.

References

[1] Peter Corke. *Robotics, Vision, and Control*. Springer, 2011.

[2] K. Grochow, S.L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *SIGGRAPH*, volume 23, pages 522–531. ACM, 2004.

[3] H. Lipkin. A note on Denavit-Hartenberg notation in robotics. In *Proc. ASME IDETC/CIE*, pages 921–926, 2005.

[4] R.M. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.