Manipulator Differential Kinematics

Part I: Kinematics, Velocity, and Applications

By Jesse Haviland and Peter Corke

Kinematics, derived from the Greek word for motion, is the branch of mechanics that studies the motion of a body, or a system of bodies, without considering mass or force. This two-part tutorial is about the kinematics of robot manipulators, and in that context, it is concerned with the relationship between the position of the robot's joints and the pose of its end effector as well as the relationships between various derivatives of those quantities. Kinematics is a fundamental concept in the study or application of robot manipulators, and our audience for Part I is students, practitioners, or researchers encountering this topic for the first time or looking for a concise refresher.

The relationship between end-effector pose and joint coordinates is encapsulated in the forward and inverse kinematics (IK), and these are among the first concepts learned in this field. This tutorial begins with forward kinematics but deliberately avoids the commonly used Denavit-Hartenberg parameters, which we feel confound pedagogy. Instead, we approach the problem using the elementary transform sequence (ETS), which is an intuitive, uncomplicated, and superior method for modeling a kinematic chain. Then, we formulate the first-order differential kinematics, which is the relationship between the velocity of the robot's joints and the end-effector velocity. This relationship is expressed in terms of the manipulator Jacobian and is foundational for many fundamental robotic control algorithms, three of which we cover in this first article.

The first is resolved-rate motion control (RRMC)—a simple algorithm that enables reactive, closed-loop, and sensor-based velocity control of a manipulator. The second is numerical IK—an important planning tool that solves for joint coordinates that correspond to a specific manipulator pose. We introduce various methods that combine first-order differential kinematics and numerical optimization and present a comprehensive experiment that compares the performance and characteristics of each numerical IK method. Third, we introduce performance measures that describe a manipulator's ability to move or exert force depending on its joint configuration.

Part II of this tutorial provides a formulation of second- and higher-order differential kinematics, introduces the manipulator Hessian, and illustrates advanced techniques, many of which improve the performance of techniques demonstrated in Part I. These are useful topics that are not well covered in current textbooks.

We have provided Jupyter Notebooks to accompany this tutorial. The Notebooks are written in Python and use the Robotics Toolbox for Python and the Swift Simulator [1] to provide full implementations of each concept, equation, and algorithm presented in this tutorial. The Notebooks use rich Markdown text and LaTeX equations to document and

Digital Object Identifier 10.1109/MRA.2023.3270228
Date of current version: 16 May 2023

1070-9932/23©2023IEEE

communicate key concepts. While not absolutely essential, for the most engaging and informative experience, we recommend working through the Jupyter Notebooks while reading this article. The Notebooks and setup instructions can be accessed at https://github.com/jhavl/dkt.

A serial-link manipulator, which we refer to as a *manipulator*, is the formal name for a robot that comprises a chain of rigid links and joints; it may contain branches, but it cannot have closed loops. Each joint provides one degree of freedom (1 DoF), which may be a prismatic joint providing translational freedom or a revolute joint providing rotational freedom. The base frame of a manipulator represents the reference frame of the first link in the chain, while the last link is known as the *end effector*.

The ETS, introduced in [1], provides a universal method for describing the kinematics of any manipulator. This intuitive and systematic approach can be applied with a simple walk-through procedure. The resulting sequence comprises a number of elementary transforms—translations and rotations—from the base frame to the robot's end effector. An example of an ETS is displayed in Figure 1 for the Franka-Emika Panda in its zero-angle configuration.

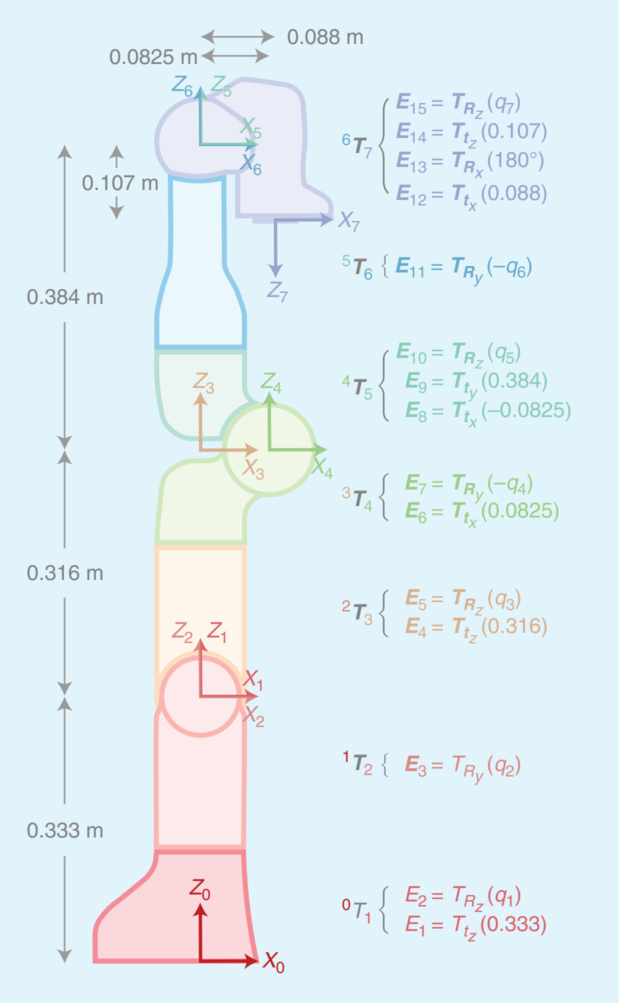
0.088 m
0.0825 m
Z_6
Z_5
X_5
X_6
0.107 m
X_7
Z_7
0.384 m
Z_3
Z_4
X_3
X_4
0.316 m
Z_2
Z_1
X_1
X_2
0.333 m
Z_0
X_0
${}^6T_7 \left\{ \begin{array}{l} E_{15} = T_{R_z}(q_7) \\ E_{14} = T_{t_z}(0.107) \\ E_{13} = T_{R_x}(180°) \\ E_{12} = T_{t_x}(0.088) \end{array} \right.$
${}^5T_6 \left\{ E_{11} = T_{R_y}(-q_6) \right.$
${}^4T_5 \left\{ \begin{array}{l} E_{10} = T_{R_z}(q_5) \\ E_9 = T_{t_y}(0.384) \\ E_8 = T_{t_x}(-0.0825) \end{array} \right.$
${}^3T_4 \left\{ \begin{array}{l} E_7 = T_{R_y}(-q_4) \\ E_6 = T_{t_x}(0.0825) \end{array} \right.$
${}^2T_3 \left\{ \begin{array}{l} E_5 = T_{R_z}(q_3) \\ E_4 = T_{t_z}(0.316) \end{array} \right.$
${}^1T_2 \left\{ E_3 = T_{R_y}(q_2) \right.$
${}^0T_1 \left\{ \begin{array}{l} E_2 = T_{R_z}(q_1) \\ E_1 = T_{t_z}(0.333) \end{array} \right.$

FIGURE 1. The ETS of the 7-DoF Franka-Emika Panda serial-link manipulator in its zero-angle configurations. $\mathbf{E}_i$ represents an elementary transform, while ${}^a\mathbf{T}_b$ represents the pose of link frame b in the reference frame of link a.

The ETS is conceptually easy to grasp, and a reader can intuitively understand the kinematics of a manipulator at a glance. Additionally, the ETS avoids the unnecessary complexity and frame assignment constraints of Denavit and Hartenberg notation [2] and allows joint rotation or translation about or along any axis. This makes the ETS a powerful tool for understanding robot kinematics, and going forward, we believe the ETS method should be considered the standard method for describing robot kinematics.

We use the notation of [3] where $\{a\}$ denotes a coordinate frame, and ${}^a\mathbf{T}_b$ is a relative pose or rigid-body transformation of $\{b\}$ with respect to $\{a\}$.

FORWARD KINEMATICS

The forward kinematics is the first and most fundamental relationship between the link geometry and robot configuration. The forward kinematics of a manipulator provides a nonlinear mapping

$${}^0\mathbf{T}_e(t) = \mathcal{K}(\boldsymbol{q}(t))$$

between the joint space and Cartesian task space, where $\boldsymbol{q}(t) = (q_1(t), q_2(t), \dots q_n(t)) \in \mathbb{R}^n$ is the vector of joint generalized coordinates, n is the number of joints, and ${}^0\mathbf{T}_e \in \mathbf{SE}(3)$ is a homogeneous transformation matrix representing the pose of the robot's end effector in the world-coordinate frame. The ETS model defines $\mathcal{K}(\cdot)$ as the product of M elementary transforms $\mathbf{E}_i \in \mathbf{SE}(3)$.

$$\begin{aligned} {}^0\mathbf{T}_e(t) &= \mathbf{E}_1(\eta_1)\,\mathbf{E}_2(\eta_2)\dots\mathbf{E}_M(\eta_M) \\ &= \prod_{i=1}^{M} \mathbf{E}_i(\eta_i). \end{aligned} \tag{1}$$

Each of the elementary transforms $\mathbf{E}_i$ can be a pure translation along or a pure rotation about the local x-, y-, or z-axis by an amount η_i. Explicitly, each transform is one of the following:

$$\mathbf{E}_i = \begin{cases} \mathbf{T}_{t_x}(\eta_i) \\ \mathbf{T}_{t_y}(\eta_i) \\ \mathbf{T}_{t_z}(\eta_i) \\ \mathbf{T}_{\mathbf{R}_x}(\eta_i) \\ \mathbf{T}_{\mathbf{R}_y}(\eta_i) \\ \mathbf{T}_{\mathbf{R}_z}(\eta_i) \end{cases} \tag{2}$$

where each of the matrices is displayed in Figure 2, and the parameter η_i is either a constant c_i (translational offset or rotation) or a joint variable $q_j(t)$

$$\eta_i = \begin{cases} c_i \\ q_j(t) \end{cases} \tag{3}$$

and the joint variable is

$$T_{t_x}(\eta) = \begin{pmatrix} 1 & 0 & 0 & \eta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{t_y}(\eta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \eta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{t_z}(\eta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \eta \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{R_x}(\eta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\eta) & -\sin(\eta) & 0 \\ 0 & \sin(\eta) & \cos(\eta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{R_y}(\eta) = \begin{pmatrix} \cos(\eta) & 0 & \sin(\eta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\eta) & 0 & \cos(\eta) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T_{R_z}(\eta) = \begin{pmatrix} \cos(\eta) & -\sin(\eta) & 0 & 0 \\ \sin(\eta) & \cos(\eta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

FIGURE 2. The six elementary transforms $\mathbf{E} \in \mathbf{SE}(3)$ from (2), which are the building blocks for ETS notation. Each homogeneous transformation matrix represents a translation along, or a rotation about, a single axis that is parameterized by η as defined in (3) and (4).

$$q_j(t) = \begin{cases} \theta_j(t) & \text{for a revolute joint} \\ d_j(t) & \text{for a prismatic joint} \end{cases} \tag{4}$$

where $\theta_j(t)$ represents a joint angle, and $d_j(t)$ represents a joint translation.

An ETS description does not require intermediate link frames, but it does not preclude their introduction. The convention we adopt is to place the jth frame $\{j\}$ immediately after the ETS term related to q_j, as shown in Figure 1. The relative transform between link frames a and b is simply a subset of the ETS

$${}^{a}\mathbf{T}_b = \prod_{i=\mu(a)}^{\mu(b)} \mathbf{E}_i(\eta_i) \tag{5}$$

where the function $\mu(j)$ returns the index of the factor in the ETS expression, (1) in this case, which is a function of q_j. For example, from Figure 1, for joint variable $j = 5$, $\mu(j) = 10$

DERIVING THE MANIPULATOR JACOBIAN

FIRST DERIVATIVE OF A POSE

In the following sections, we use the skew and inverse skew operation, as defined in Figure 3. Now consider the end-effector pose, which is a function of the joint coordinates given by (1). Its derivative with respect to time is

$$\dot{\mathbf{T}} = \frac{\mathrm{d}\mathbf{T}}{\mathrm{d}t} = \frac{\partial \mathbf{T}}{\partial q_1}\dot{q}_1 + \cdots + \frac{\partial \mathbf{T}}{\partial q_n}\dot{q}_n \in \mathbb{R}^{4\times 4} \tag{6}$$

where each $\partial\mathbf{T}/\partial q_i \in \mathbb{R}^{4\times 4}$.

The information in $\mathbf{T}$ is nonminimal and redundant, as is the information in $\dot{\mathbf{T}}$. We can write these respectively as

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix}, \quad \dot{\mathbf{T}} = \begin{pmatrix} \dot{\mathbf{R}} & \dot{\boldsymbol{t}} \\ 0 & 0 \end{pmatrix} \tag{7}$$

where $\mathbf{R} \in \mathbf{SO}(3)$, $\dot{\mathbf{R}} \in \mathbb{R}^{3\times 3}$, and $\boldsymbol{t}, \dot{\boldsymbol{t}} \in \mathbb{R}^3$.

We will write the partial derivative in partitioned form as

$$\frac{\partial \mathbf{T}}{\partial q_j} = \begin{pmatrix} \mathbf{J}_{R_j} & \mathbf{J}_{t_j} \\ 0 & 0 \end{pmatrix} \tag{8}$$

where $\mathbf{J}_{R_j} \in \mathbb{R}^{3\times 3}$ and $\mathbf{J}_{t_j} \in \mathbb{R}^{3\times 1}$, and then we will rewrite (6) as

$$\boldsymbol{s} = \vee_{\times}(\boldsymbol{S}) = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} \qquad \boldsymbol{S} = [\boldsymbol{s}]_{\times} = \begin{pmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{pmatrix}$$

$$\hat{\boldsymbol{s}} = \vee_{\times}(\hat{\boldsymbol{S}}) = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \qquad \hat{\boldsymbol{S}} = [\hat{\boldsymbol{s}}] = \begin{pmatrix} 0 & -s_6 & s_5 & s_1 \\ s_6 & 0 & -s_4 & s_2 \\ -s_5 & s_4 & 0 & s_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

FIGURE 3. Shown on the left is a vector $\boldsymbol{s} \in \mathbb{R}^3$ along with its corresponding skew-symmetric matrix $\mathbf{S} \in \mathbf{so}(3) \subset \mathbb{R}^{3\times 3}$. Shown on the right is a vector $\hat{\boldsymbol{s}} \in \mathbb{R}^6$ along with its corresponding augmented skew-symmetric matrix $\hat{\mathbf{S}} \in \mathbf{se}(3) \subset \mathbb{R}^{4\times 4}$. The skew functions $[\cdot]_\times : \mathbb{R}^3 \mapsto \mathbf{so}(3)$ maps a vector to a skew-symmetric matrix, and $[\cdot] : \mathbb{R}^6 \mapsto \mathbf{se}(3)$ maps a vector to an augmented skew-symmetric matrix. The inverse skew functions $\vee_\times(\cdot) : \mathbf{so}(3) \mapsto \mathbb{R}^3$ maps a skew-symmetric matrix to a vector, and $\vee(\cdot) : \mathbf{se}(3) \mapsto \mathbb{R}^6$ maps an augmented skew-symmetric matrix to a vector.

$$\begin{pmatrix} \dot{\mathbf{R}} & \dot{\boldsymbol{t}} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{J}_{R_1} & \mathbf{J}_{t_1} \\ 0 & 0 \end{pmatrix} \dot{q}_1 + \cdots + \begin{pmatrix} \mathbf{J}_{R_n} & \mathbf{J}_{t_n} \\ 0 & 0 \end{pmatrix} \dot{q}_n$$

and write a matrix equation for each nonzero partition

$$\dot{\mathbf{R}} = \mathbf{J}_{R_1}\dot{q}_1 + \cdots + \mathbf{J}_{R_n}\dot{q}_n \tag{9}$$

$$\dot{\boldsymbol{t}} = \mathbf{J}_{t_1}\dot{q}_1 + \cdots + \mathbf{J}_{t_n}\dot{q}_n \tag{10}$$

where each term represents the contribution to end-effector velocity due to the motion of the corresponding joint.

Taking (10) first, we can simply write

$$\begin{aligned} \dot{\boldsymbol{t}} &= (\mathbf{J}_{t_1} \ \cdots \ \mathbf{J}_{t_n}) \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix} \\ &= \mathbf{J}_v(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{11}$$

where $\mathbf{J}_v(\boldsymbol{q}) \in \mathbb{R}^{3\times n}$ is the translational part of the manipulator Jacobian matrix.

The rotation rate is slightly more complex, but using the identity $\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times \mathbf{R}$, where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity and $[\boldsymbol{\omega}]_\times \in \mathbf{so(3)}$ is a skew-symmetric matrix, we can rewrite (9) as

$$[\boldsymbol{\omega}]_\times \mathbf{R} = \mathbf{J}_{R_1}\dot{q}_1 + \cdots + \mathbf{J}_{R_n}\dot{q}_n \tag{12}$$

and rearrange it to

$$[\boldsymbol{\omega}]_\times = (\mathbf{J}_{R_1}\mathbf{R}^\top)\dot{q}_1 + \cdots + (\mathbf{J}_{R_n}\mathbf{R}^\top)\dot{q}_n \in \mathbf{so}(3).$$

This 3×3 matrix equation, therefore, has only three unique equations, so applying the inverse skew operator to both sides, we have

$$\begin{aligned} \boldsymbol{\omega} &= \vee_\times(\mathbf{J}_{R_1}\mathbf{R}^\top)\dot{q}_1 + \cdots + \vee_\times(\mathbf{J}_{R_n}\mathbf{R}^\top)\dot{q}_n \\ &= (\vee_\times(\mathbf{J}_{R_1}\mathbf{R}^\top) \ \cdots \ \vee_\times(\mathbf{J}_{R_n}\mathbf{R}^\top)) \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix} \\ &= \mathbf{J}_\omega(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{13}$$

where $\mathbf{J}_\omega(\boldsymbol{q}) \in \mathbb{R}^{3\times n}$ is the rotational part of the manipulator Jacobian.

Combining (11) and (13), we can write

$${}^0\boldsymbol{\nu} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_v(\boldsymbol{q}) \\ \mathbf{J}_\omega(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}} \tag{14}$$

which expresses the end-effector spatial velocity ${}^0\boldsymbol{\nu} = (v_x, \ v_y, \ v_z, \ \omega_x, \ \omega_y, \ \omega_z)$ in the world frame in terms of joint velocity, and

$${}^0\mathbf{J}(\boldsymbol{q}) = \begin{pmatrix} \mathbf{J}_v(\boldsymbol{q}) \\ \mathbf{J}_\omega(\boldsymbol{q}) \end{pmatrix} \in \mathbb{R}^{6\times n} \tag{15}$$

is the manipulator Jacobian matrix expressed in the world-coordinate frame. The Jacobian expressed in the end-effector frame is

$${}^e\mathbf{J}(\boldsymbol{q}) = \begin{pmatrix} {}^0\mathbf{R}_e^\top & \mathbf{0} \\ \mathbf{0} & {}^0\mathbf{R}_e^\top \end{pmatrix} {}^0\mathbf{J}(\boldsymbol{q}) \tag{16}$$

where ${}^0\mathbf{R}_e$ is a rotation matrix representing the orientation of the end effector in the world frame.

Combining (14) and (15), we write

$${}^0\boldsymbol{\nu} = {}^0\mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{17}$$

which provides the derivative of the left side of (1). However, to compute (17), we need to first compute (8), that is $(\partial\mathbf{T}/\partial q_j)$.

FIRST DERIVATIVE OF AN ELEMENTARY TRANSFORM

Before differentiating the ETS to find the manipulator Jacobian, it is useful to consider the derivative of a single elementary transform.

DERIVATIVE OF A PURE ROTATION

The derivative of a rotation matrix with respect to the rotation angle θ is required when considering a revolute joint and can be shown to be

$$\frac{\mathrm{d}\mathbf{R}(\theta)}{\mathrm{d}\theta} = [\hat{\boldsymbol{\omega}}]_\times \mathbf{R}(\theta(t)) \tag{18}$$

where the unit vector $\hat{\boldsymbol{\omega}}$ is the joint rotation axis.

The rotation axis $\hat{\boldsymbol{\omega}}$ can be recovered using the inverse skew operator

$$\hat{\boldsymbol{\omega}} = \vee_\times\left(\frac{\mathrm{d}\mathbf{R}(\theta)}{\mathrm{d}\theta}\mathbf{R}(\theta(t))^\top\right) \tag{19}$$

since $\mathbf{R} \in \mathbf{SO}(3)$, then $\mathbf{R}^{-1} = \mathbf{R}^\top$.

For an ETS, we only need to consider the elementary rotations $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$, as shown in Figure 2, which are embedded within $\mathbf{SE}(3)$, as $\mathbf{T}_{\mathbf{R}_x}, \mathbf{T}_{\mathbf{R}_y}$, and $\mathbf{T}_{\mathbf{R}_z}$ i.e., pure rotations with no translational component. We can show that the derivative of each elementary rotation with respect to a rotation angle is

$$\frac{\mathrm{d}\mathbf{T}_{\mathbf{R}_x}(\theta)}{\mathrm{d}\theta} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{T}_{\mathbf{R}_x}(\theta) = [\hat{\boldsymbol{R}}_x]\,\mathbf{T}_{\mathbf{R}_x}(\theta), \tag{20}$$

$$\frac{\mathrm{d}\mathbf{T}_{\mathbf{R}_y}(\theta)}{\mathrm{d}\theta} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{T}_{\mathbf{R}_y}(\theta) = [\hat{\boldsymbol{R}}_y]\,\mathbf{T}_{\mathbf{R}_y}(\theta) \tag{21}$$

$$\frac{\mathrm{d}\mathbf{T}_{\mathbf{R}_z}(\theta)}{\mathrm{d}\theta} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{T}_{\mathbf{R}_z}(\theta) = [\hat{\boldsymbol{R}}_z]\,\mathbf{T}_{\mathbf{R}_z}(\theta). \tag{22}$$

If a joint's defined positive rotation is a negative rotation about the axis, as is $\mathbf{E}_7$ and $\mathbf{E}_{11}$ in the ETS of the Panda shown in Figure 1, then $[\mathbf{R}_i]^\top\,\mathbf{T}_{\mathbf{R}_i}(\theta)^\top$ is used to calculate the derivative.

DERIVATIVE OF A PURE TRANSLATION

Consider the three elementary translations $\mathbf{T}_t$ shown in Figure 2. The derivative of a homogeneous transformation matrix with respect to translation is required when considering a prismatic joint. For an ETS, these translations are embedded in $\mathbf{SE}(3)$ as $\mathbf{T}_{t_x}$, $\mathbf{T}_{t_y}$, and $\mathbf{T}_{t_z}$, which are pure translations with zero rotational components. We can show that the derivative of each elementary translation with respect to a translation is

$$\frac{\mathrm{d}\mathbf{T}_{t_x}(d)}{\mathrm{d}d} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = [\hat{\boldsymbol{t}}_x] \tag{23}$$

$$\frac{\mathrm{d}\mathbf{T}_{t_y}(d)}{\mathrm{d}d} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = [\hat{\boldsymbol{t}}_y] \tag{24}$$

$$\frac{\mathrm{d}\mathbf{T}_{t_z}(d)}{\mathrm{d}d} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = [\hat{\boldsymbol{t}}_z]. \tag{25}$$

No changes are required if a joint's defined positive translation is a negative translation along the axis.

THE MANIPULATOR JACOBIAN

Now we can calculate the derivative of an ETS. To find out how the jth joint affects the end-effector pose, apply the chain rule to (1)

$$\begin{aligned} \frac{\partial \mathbf{T}(\boldsymbol{q})}{\partial q_j} &= \frac{\partial}{\partial q_j}(\mathbf{E}_1(\eta_1)\mathbf{E}_2(\eta_2)\ldots\mathbf{E}_M(\eta_M)) \\ &= \prod_{i=1}^{\mu(j)-1}\mathbf{E}_i(\eta_i)\ \frac{\mathrm{d}\mathbf{E}_{\mu(j)}(q_j)}{\mathrm{d}q_j} \prod_{i=\mu(j)+1}^{M}\mathbf{E}_i(\eta_i). \end{aligned} \tag{26}$$

The derivative of the elementary transform with respect to a joint coordinate in (26) is obtained using one of (20) and (21); or (22) for a revolute joint; or one of (23), (24), or (25) for a prismatic joint.

Using (13) with (26), we can form the angular velocity component of the jth column of the manipulator Jacobian

$$\mathbf{J}_{\omega_j}(\boldsymbol{q}) = \vee_\times\left(\rho\left(\frac{\partial \mathbf{T}(\boldsymbol{q})}{\partial q_j}\right)\rho(\mathbf{T}(\boldsymbol{q}))^\top\right) \tag{27}$$

where the $\rho(\cdot)$ operation is defined in Figure 4. Using (11) with (26), the translational velocity component of the jth column of the manipulator Jacobian is

$$\mathbf{J}_{v_j}(\boldsymbol{q}) = \tau\left(\frac{\partial \mathbf{T}(\boldsymbol{q})}{\partial q_j}\right) \tag{28}$$

where the $\tau(\cdot)$ operation is defined in Figure 4.

Stacking the translational and angular velocity components, the jth column of the manipulator Jacobian becomes

$$\mathbf{J}_j(\boldsymbol{q}) = \begin{pmatrix} \mathbf{J}_{v_j}(\boldsymbol{q}) \\ \mathbf{J}_{\omega_j}(\boldsymbol{q}) \end{pmatrix} \in \mathbb{R}^6 \tag{29}$$

where the full manipulator Jacobian is

$$\mathbf{J}(\boldsymbol{q}) = (\mathbf{J}_1(\boldsymbol{q}) \ \cdots \ \mathbf{J}_n(\boldsymbol{q})) \in \mathbb{R}^{6\times n} \tag{30}$$

and we display a visualization of the general structure of the manipulator Jacobian in Figure 5.

FAST MANIPULATOR JACOBIAN

Calculating the manipulator Jacobian using (27) and (28) is easy to understand but has $\mathcal{O}(n^2)$ time complexity—we can do better.

Expanding (27) using (26) and simplifying using $\mathbf{RR}^\top = \mathbf{1}$ gives

$$\mathbf{J}_{\omega_j}(\boldsymbol{q}) = \vee_\times(\rho({}^0\mathbf{T}_j)\rho([\hat{\boldsymbol{G}}_{\mu(j)}])(\rho({}^0\mathbf{T}_j)^\top)) \tag{31}$$

where ${}^0\mathbf{T}_j$ represents the transform from the base frame to frame $\{j\}$ as described by (5), and $[\hat{\boldsymbol{G}}_{\mu(j)}]$ corresponds to one of the six elementary transform derivatives from (20), (21), and (22) and from (23), (24), and (25).

In the case of a prismatic joint, $\rho(\hat{\boldsymbol{G}}_{\mu(j)})$ will be a 3×3 matrix of zeros, which results in zero angular velocity. In the

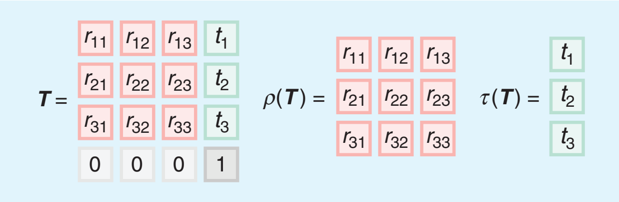
$\boldsymbol{T} =$
r_{11} r_{12} r_{13} t_1
r_{21} r_{22} r_{23} t_2
r_{31} r_{32} r_{33} t_3
0 0 0 1
$\rho(\boldsymbol{T}) =$
r_{11} r_{12} r_{13}
r_{21} r_{22} r_{23}
r_{31} r_{32} r_{33}
$\tau(\boldsymbol{T}) =$
t_1
t_2
t_3

FIGURE 4. A visualization of a homogeneous transformation matrix (the derivatives share the form of $\mathbf{T}$ except that they will have a zero instead of a one located at $\mathbf{T}_{44}$), where the matrix $\rho(\mathbf{T}) \in \mathbb{R}^{3\times3}$ of red boxes forms the rotation component, and the vector $\tau(\mathbf{T}) \in \mathbb{R}^3$ of blue boxes forms the translation component. The rotation component can be extracted through the function $\rho(\cdot): \mathbb{R}^{4\times4} \to \mathbb{R}^{3\times3}$, while the translation component can be extracted through the function $\tau(\cdot): \mathbb{R}^{4\times4} \to \mathbb{R}^3$.

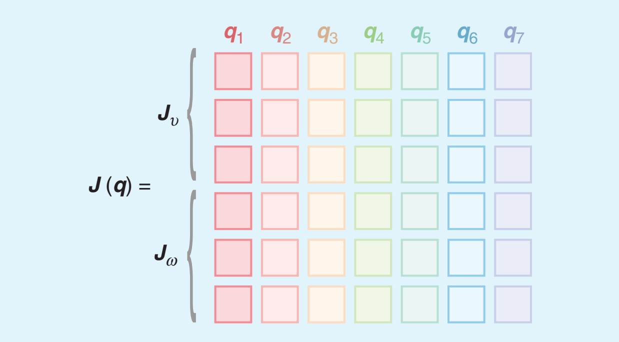
q_1 q_2 q_3 q_4 q_5 q_6 q_7
$\boldsymbol{J}_v$
$\boldsymbol{J}(\boldsymbol{q}) =$
$\boldsymbol{J}_\omega$

FIGURE 5. A visualization of the Jacobian $\mathbf{J}(\boldsymbol{q})$ of a seven-jointed manipulator. Each column describes how the end-effector pose changes due to the motion of the corresponding joint. The top three rows $\mathbf{J}_v$ correspond to the linear velocity of the end effector, while the bottom three rows $\mathbf{J}_\omega$ correspond to the angular velocity of the end effector.

case of a revolute joint, the angular velocity is parallel to the axis of joint rotation

$$\mathbf{J}_{\omega_j}(\boldsymbol{q}) = \begin{cases} \hat{\boldsymbol{n}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_x} \\ \hat{\boldsymbol{o}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_y} \\ \hat{\boldsymbol{a}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_z} \\ (0 \;\; 0 \;\; 0)^\top & \text{if } \mathbf{E}_m = \mathbf{T}_t \end{cases} \tag{32}$$

where $(\hat{\boldsymbol{n}}_j \;\; \hat{\boldsymbol{o}}_j \;\; \hat{\boldsymbol{a}}_j) = \rho({}^0\mathbf{T}_j)$ are the columns of a rotation matrix, as shown in Figure 6.

Expanding (28) using (26) provides

$$\mathbf{J}_{\nu_j}(\boldsymbol{q}) = \tau({}^0\mathbf{T}_j[\hat{\boldsymbol{G}}_{\mu(j)}]\,{}^j\mathbf{T}_e) \tag{33}$$

which reduces to

$$\mathbf{J}_{\nu_j}(\boldsymbol{q}) = \begin{cases} \hat{\boldsymbol{a}}_j y_e - \hat{\boldsymbol{o}}_j z_e & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_x} \\ \hat{\boldsymbol{n}}_j z_e - \hat{\boldsymbol{a}}_j x_e & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_y} \\ \hat{\boldsymbol{o}}_j x_e - \hat{\boldsymbol{y}}_j y_e & \text{if } \mathbf{E}_m = \mathbf{T}_{\mathbf{R}_z} \\ \hat{\boldsymbol{n}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{t_x} \\ \hat{\boldsymbol{o}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{t_y} \\ \hat{\boldsymbol{a}}_j & \text{if } \mathbf{E}_m = \mathbf{T}_{t_z} \end{cases} \tag{34}$$

where $(x_e \;\; y_e \;\; z_e)^\top = \tau({}^j\mathbf{T}_e)$.

This simplification reduces the time complexity of computation of the manipulator Jacobian to $\mathcal{O}(n)$.

MANIPULATOR JACOBIAN APPLICATIONS

The manipulator Jacobian is widely used in robotic control algorithms, and the remainder of this article details several applications of the manipulator Jacobian.

RRMC

RRMC, a direct application of the first-order differential equation we generated in (17), is a simple and elegant method to generate joint velocities that create a desired Cartesian end effector motion [4]. The ability to instantaneously change the end-effector velocity is at the heart of sensor-based reactive control. By "closing the loop" on sensory information, a robot can adapt to changes in its environment as new knowledge is acquired by sensors. A well-known example is visual servoing, a technique that pairs robotic vision with RRMC to guide the manipulator to an image-based goal [5]. RRMC is also foundational for numerical IK, which we will visit in the next section [6], [7]. Of course, this introductory technique, while still highly relevant and powerful, can be extended and enhanced to improve robustness and consider constraints such as joint limits and environmental collisions [8], [9], [10], [11], [12], [13].

We first rearrange (17)

$$\dot{\boldsymbol{q}} = {}^0\mathbf{J}(\boldsymbol{q})^{-1}\,{}^0\boldsymbol{\nu} \tag{35}$$

which can be solved only when $\mathbf{J}(\boldsymbol{q})$ is square (and nonsingular), which is when the robot has 6 DoF.

For redundant robots, there is no unique solution for (35). Consequently, the most common approach is to use the Moore-Penrose pseudoinverse

$$\dot{\boldsymbol{q}} = {}^0\mathbf{J}(\boldsymbol{q})^{+}\,{}^0\boldsymbol{\nu}. \tag{36}$$

The pseudoinverse will return joint velocities with the minimum velocity norm of the possible solutions, although note an important caveat on mixed units described in Excurse 1. Immediately from this, we can construct a primitive open-loop velocity controller. At each time step, we must calculate the manipulator Jacobian $\mathbf{J}(\boldsymbol{q})$, which is a function of the robot's current configuration $\boldsymbol{q}$. Then, we set ${}^0\boldsymbol{\nu}$ to the desired spatial velocity of the end effector.

A more useful application of RRMC is to employ it in a closed-loop pose controller, which we denote as *position-based servoing* (*PBS*). Using this method, we can make the end effector travel in a straight line, in the robot's task space, toward some desired end-effector pose, as displayed in Figure 7. The PBS scheme relies on an error vector that represents the translation and rotation from the end effector's current pose to the desired pose

$$\boldsymbol{e} = \begin{pmatrix} \tau({}^0\mathbf{T}_{e^*}) - \tau({}^0\mathbf{T}_e) \\ \alpha(\rho({}^0\mathbf{T}_{e^*})\rho({}^0\mathbf{T}_e)^\top) \end{pmatrix} \in \mathbb{R}^6 \tag{37}$$

where the top three rows correspond to the translational error in the world frame, the bottom three rows correspond to the

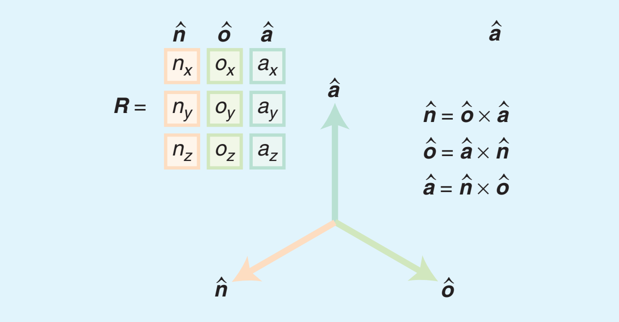
$\hat{n}$ $\hat{o}$ $\hat{a}$
$R =$
n_x o_x a_x
n_y o_y a_y
n_z o_z a_z
$\hat{a}$
$\hat{n} = \hat{o} \times \hat{a}$
$\hat{o} = \hat{a} \times \hat{n}$
$\hat{a} = \hat{n} \times \hat{o}$
$\hat{a}$
$\hat{n}$
$\hat{o}$

FIGURE 6. The two vector representation of a rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$. The rotation matrix $\mathbf{R}$ describes the coordinate frame in terms of three orthogonal vectors, $\hat{\boldsymbol{n}}, \hat{\boldsymbol{o}}$, and $\hat{\boldsymbol{a}}$, which are the axes of the rotated frame expressed in the reference coordinate frame $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$. As shown previously, each of the vectors $\hat{\boldsymbol{n}}, \hat{\boldsymbol{o}}$, and $\hat{\boldsymbol{a}}$ can be calculated using the cross-product of the other two.

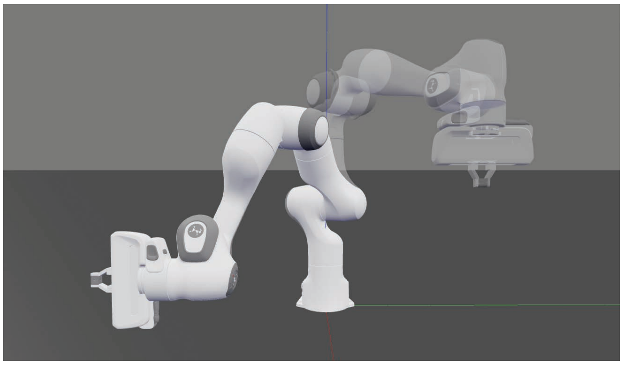

FIGURE 7. A visualization of a Panda robot that has been controlled by a PBS control scheme utilizing RRMC. The end-effector's pose has both been translated and rotated to reach the desired pose.

Excurse 1. Inhomogeneous Units

Vector norms can be problematic when they contain mixed units. For example, $\| \boldsymbol{e} \|$ and $\| \nu \|$ are nonhomogeneous as they both combine translational and rotational units. For manipulators with mixed prismatic and revolute joints, so, too, does $\| \dot{\boldsymbol{q}} \|$. Solutions include scaling [14] to ensure that the translational and rotational values have a commensurate magnitude or to consider translational and rotational components separately.

rotational error in the world frame, ${}^0\mathbf{T}_e = \mathcal{K}(\boldsymbol{q})$ is the forward kinematics of the robot, which represents the end-effector pose in the base frame of the robot, ${}^0\mathbf{T}_{e^*}$ is the desired end-effector pose in the base frame of the robot ($\cdot^*$ denotes desired), and $\alpha(\cdot): \mathbf{SO} \longmapsto \mathbb{R}^3$ transforms a rotation matrix to its Euler vector equivalent [7]. The Euler vector is a form of angle-axis representation and specifies an axis of rotation and the angle of rotation about that axis.

If $\mathbf{R}$ is not a diagonal matrix, then the Euler vector equivalent of $\mathbf{R}$ is calculated as

$$\alpha(\mathbf{R}) = \frac{\text{atan2}(\| \boldsymbol{l} \|, r_{11} + r_{22} + r_{33} - 1)}{\| \boldsymbol{l} \|} \boldsymbol{l}. \tag{38}$$

where, using the convention from Figure 4, we define $\rho(\mathbf{T}) = \mathbf{R} = \{r_{ij}\}$ and

$$\boldsymbol{l} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}. \tag{39}$$

If $\mathbf{R}$ is a diagonal matrix then we use different formulas. For the case where $(r_{11} \;\; r_{22} \;\; r_{33}) = (1 \;\; 1 \;\; 1)$, then $\alpha(\mathbf{R}) = (0 \;\; 0 \;\; 0)^\top$; otherwise,

$$\alpha(\mathbf{R}) = \frac{\pi}{2} \begin{pmatrix} r_{11} + 1 \\ r_{22} + 1 \\ r_{33} + 1 \end{pmatrix}. \tag{40}$$

To construct the PBS scheme, we take the error term from (37) to set $\boldsymbol{\nu}$ in (35) [or (36) for robots with 7+ DoF] at each time step

$$\boldsymbol{\nu} = \mathbf{k}\boldsymbol{e} \tag{41}$$

where $\mathbf{k}$ is a proportional gain with units s^{-1} that controls the rate of convergence to the goal and is typically a diagonal matrix to set gains for each task-space DoF

$$\mathbf{k} = \text{diag}(k_t, k_t, k_t, k_r, k_r, k_r) \tag{42}$$

where k_t is the translational motion gain, and k_r is the rotational motion gain. The inverse of the gain, $\mathbf{k}^{-1}$, corresponds to the time constant of the scheme. Note that since the error $\boldsymbol{e}$ is in the base frame of the robot, we must use the base-frame manipulator Jacobian ${}^0\mathbf{J}(\boldsymbol{q})$ in (35) or (36).

The control scheme we have just described will cause the error to asymptotically decrease to zero. For real applications, this is slow and impractical. We can improve this by increasing k_t and k_r and capping the end-effector velocity norm $\| \boldsymbol{\nu} \|$ at some value ν_m before stopping when the error norm $\| \boldsymbol{e} \|$ drops below some value e_m

$$\boldsymbol{\nu} = \begin{cases} \mathbf{k}\boldsymbol{e} \dfrac{\nu_m}{\| \mathbf{k}\boldsymbol{e} \|} & \text{if } \| \mathbf{k}\boldsymbol{e} \| > \nu_m \\ \mathbf{k}\boldsymbol{e} & \text{if } \nu_m \geq \| \mathbf{k}\boldsymbol{e} \| > e_m \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{43}$$

This will cause $\| \boldsymbol{\nu} \|$ to be consistent until e_m is reached, and subsequently, $\| \boldsymbol{\nu} \|$ asymptotically decreases to safely stop the robot. The effect of this is displayed in Figure 8, where ν_m has been set to 2.0. This approach involves combining mixed units; see Excurse 1.

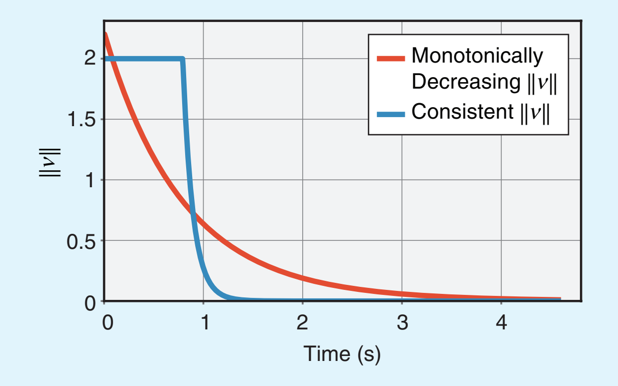

FIGURE 8. A comparison of the Euclidean distance error from the end-effectors' pose to the desired pose using the monotonically decreasing $\| \boldsymbol{\nu} \|$ method from (41) and the consistent $\| \boldsymbol{\nu} \|$ method from (43).

This particular approach may cause unintuitive results as both $\| \boldsymbol{e} \|$ and $\| \boldsymbol{\nu} \|$ are nonhomogeneous—they combine translational and rotational units. Therefore, a more intuitive approach would be to modify (43) to adjust the translational and angular velocity profiles separately. Alternate velocity profiles, such as a linearly decreasing velocity norm, are possible through the modification of (43).

Although we have the safety mechanism of capping $\| \boldsymbol{\nu} \|$ at ν_m, care must still be taken to not increase k_t and k_r without due consideration. Well-known problems caused by an excessive gain value, such as overshoot and system instability, may become apparent, especially on a real-world robot (which has nonmodeled dynamics, including flexibility, communication delay, and nonlinearities).

For a reactive task, where a dynamic environment is causing the goal pose to change, simply update the desired end-effector pose ${}^0\mathbf{T}_{e^*}$ in (37) at each time step, based on sensory information. A technique such as position-based visual servoing could be used to achieve this [5]. Refer to Notebook 3, where we use the Swift simulator to show this in action.

NUMERICAL IK

IK is the problem of determining the corresponding joint coordinates given some end-effector pose. There are two approaches to solving IK: analytical and numerical.

Analytical formulas must be pregenerated for a given manipulator and, in some cases, may not exist. The IKFast program, provided as a part of OpenRAVE, precompiles analytic solutions for a given manipulator in optimized C++ code [15]. After this initial step, IK can be computed rapidly, taking as little as 5 ms. However, analytic solutions generally cannot optimize for additional criteria such as joint limits.

Numerical IK uses an iterative technique and can additionally consider extra constraints such as collision avoidance, joint limit avoidance, or manipulability [6], [7], [16]. In this section, we first construct a primitive numerical IK solver before showing how it can be improved using advanced optimization techniques.

The Newton-Raphson (NR) method for IK is remarkably similar to RRMC in a PBS scheme. However, instead of sending the joint velocities to the manipulator, the joint velocities update the configuration of a virtual manipulator until the goal pose is reached. To find the joint coordinates that correspond to some end-effector pose ${}^{0}\mathbf{T}_{e^*}$, the NR method seeks to minimize an error function

$$E = \frac{1}{2} e^{\top} \mathbf{W}_e \boldsymbol{e} \tag{44}$$

where $\boldsymbol{e}$ is defined in (37), and $\mathbf{W}_e = \operatorname{diag}(\boldsymbol{w}_e)(\boldsymbol{w}_e \in (\mathbb{R}^+)^n)$ is a diagonal weighting matrix that prioritizes the corresponding error term. To achieve this, we iterate upon the following

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + {}^{0}\mathbf{J}(\boldsymbol{q}_k)^{-1} \boldsymbol{e}_k. \tag{45}$$

With this approach, ${}^{0}\mathbf{J}(\boldsymbol{q})$ must be square (and nonsingular) to be invertible and is therefore limited to manipulators with six joints.

When using the NR method, the initial joint coordinates q_0 should correspond to a nonsingular manipulator pose since it uses the manipulator Jacobian. When the problem is solvable, it converges very quickly. However, this method frequently fails to converge on the goal. We can improve the solvability of the problem and add compatibility with manipulators with greater than six joints by using the Gauss-Newton (GN) method

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \left(\mathbf{J}(\boldsymbol{q}_k)^{\top} \mathbf{W}_e \, \mathbf{J}(\boldsymbol{q}_k)\right)^{-1} \boldsymbol{g}_k \tag{46}$$

$$\boldsymbol{g}_k = \mathbf{J}(\boldsymbol{q}_k)^{\top} \mathbf{W}_e \boldsymbol{e}_k \tag{47}$$

where $\mathbf{J} = {}^{0}\mathbf{J}$ is the base-frame manipulator Jacobian. If $\mathbf{J}(\boldsymbol{q}_k)$ is nonsingular, and $\mathbf{W}_e = \mathbf{1}_n$, then (46) provides the pseudoinverse solution to (45). However, if $\mathbf{J}(\boldsymbol{q}_k)$ is singular, (46) cannot be computed, and the GN solution is infeasible.

Most linear algebra libraries (including the Python NumPy library) implement the pseudoinverse using singular value decomposition, which is robust to singular matrices. Therefore, we can make both solvers more robust by using the pseudoinverse instead of the normal inverse in (45) and (46).

However, the computation is still unstable near singular points. We can further improve the solvability through the Levenberg-Marquardt (LM) method

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + (\boldsymbol{A}_k)^{-1} \boldsymbol{g}_k \tag{48}$$

$$\mathbf{A}_k = \mathbf{J}(\boldsymbol{q}_k)^{\top} \mathbf{W}_e \, \mathbf{J}(\boldsymbol{q}_k) + \mathbf{W}_n \tag{49}$$

where $\mathbf{W}_n = \operatorname{diag}(\boldsymbol{w}_n)(\boldsymbol{w}_n \in (\mathbb{R}^+)^n)$ is a diagonal damping matrix. The damping matrix ensures that $\mathbf{A}_k$ is nonsingular and positive definite. The performance of the LM method largely depends on the choice of $\mathbf{W}_n$. Wampler [17] proposed $\boldsymbol{w}_n$ to be a constant; Chan and Lawrence [18] proposed a damped least-squares method with

$$\mathbf{W}_n = \lambda E_k \mathbf{1}_n \tag{50}$$

where λ is a constant that does not have much influence on performance, and E_k is the error value from (44) at time step k. Sugihara [7] proposed

$$\mathbf{W}_n = E_k \mathbf{1}_n + \operatorname{diag}(\tilde{\boldsymbol{w}}_n) \tag{51}$$

where $\tilde{\boldsymbol{w}}_n \in \mathbb{R}^n$, $\hat{w}_{n_i} = l^2 \sim 0.01 l^2$, and l is the length of a typical link within the manipulator.

An important point to note is that the aforementioned methods are subject to local minima and, in some cases, will fail to converge on the solution. The choice of the initial joint configuration $\boldsymbol{q}_0$ is important. An alternative approach is to restart an IK problem with a new random $\boldsymbol{q}_0$ after a few 20~50 iterations rather than persist with a single attempt with 500~5000 iterations. This is a simple but effective method of performing a global search for the IK solution.

We display a comparison of IK methods presented in this tutorial in Table 1. Table 1 shows the results for several IK algorithms trying to solve for 10000 randomly generated reachable end-effector poses using a 6-DoF UR5 manipulator. We show each method initialized with a random valid q_0 and a maximum of 500 iterations to reach the goal before being declared infeasible. We also show each of the methods with 100 searches to reach the goal where a new search is initialized with a new random valid $\boldsymbol{q}_0$ after 30 iterations. The iterations recorded for this method include the iterations from failed searches.

We have presented some useful IK methods, but it is by no means an exhaustive list. Sugihara [7] provides a good comparison of many different numerical IK methods. In Part II of this tutorial, we explore advanced IK techniques that incorporate additional constraints into the optimization.

MANIPULATOR PERFORMANCE METRICS

Manipulator performance metrics seek to quantify the performance of a manipulator in a given configuration. In this section, we explore two common manipulator performance metrics based on the manipulator Jacobian. A full survey of performance metrics can be found in [19]. It is important to note several considerations on how manipulator-based performance metrics should be used in practice. First, the metrics are unitless, and the upper bound of a metric depends on the manipulator kinematic model (i.e., joint types and link lengths) and the units chosen to represent translation and orientation. Consequently, metrics computed for different manipulators are not directly comparable. Second, the manipulator Jacobian contains three rows corresponding to translational rates and three rows corresponding to angular rates. Therefore, any metric using the whole Jacobian will produce a nonhomogeneous result due to the mixed units. Depending on the manipulator scale, this can cause either the translational or rotational component to dominate the result. This problem also arises in manipulators with mixed prismatic and revolute joints. In general, the most intuitive use of performance metrics comes from using only the translational or rotational rows of the manipulator Jacobian (where the choice of which depends on the use case) and using only the metric on a manipulator comprising a single joint type [19].

MANIPULABILITY INDEX

The Yoshikawa manipulability index [20] is the most widely used and accepted performance metric [19]. The index is calculated as

$$m(\boldsymbol{q}) = \sqrt{\det(\hat{\mathbf{J}}(q)\hat{\mathbf{J}}(q)^{\top})} \tag{52}$$

where $\hat{\mathbf{J}}(\boldsymbol{q}) \in \mathbb{R}^{3 \times n}$ is either the translational or rotational rows of $\mathbf{J}(\boldsymbol{q})$ causing $m(\boldsymbol{q})$ to describe the corresponding component of manipulability. Note that Yoshikawa used $\mathbf{J}(\boldsymbol{q})$ instead of $\hat{\mathbf{J}}(\boldsymbol{q})$ in (52), but we describe it so due to the limitations of Jacobian-based measures previously discussed. The scalar $m(\boldsymbol{q})$ describes the volume of a 3D ellipsoid—if this ellipsoid is close to spherical, then the manipulator can achieve any arbitrary end-effector (translational or rotational depending on $\hat{\mathbf{J}}(\boldsymbol{q})$) velocity. The ellipsoid is described by three radii aligned with its principal axes. A small radius indicates the robot's inability to achieve a velocity in the corresponding direction. At a singularity, the ellipsoid's radius becomes zero along the corresponding axis, and the volume becomes zero. If the manipulator's configuration is well conditioned, these ellipsoids will have a larger volume. Therefore, the manipulability index is essentially a measure of how easily a manipulator can achieve an arbitrary velocity.

CONDITION NUMBER

The condition number of the manipulator Jacobian was proposed as a performance measure in [21]. The condition number is

$$\kappa = \frac{\sigma_{\max}}{\sigma_{\min}} \in [1, \infty] \tag{53}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular values of $\hat{\mathbf{J}}(\boldsymbol{q})$, respectively. The condition number is a

TABLE 1. Numerical IK methods compared on 10,000 problems with a 6-DoF UR5 manipulator.

METHOD	SEARCHES ALLOWED	ITER. ALLOWED	MEAN ITER.	MEDIAN ITER.	INFEASIBLE COUNT	INFEASIBLE %	MEAN SEARCHES	MAX SEARCHES
NR	1	500	21.34	16	1,093	10.93%	1	1
GN	1	500	21.6	16	1,078	10.78%	1	1
NR pseudoinverse	1	500	21.24	16	1,100	11%	1	1
GN pseudoinverse	1	500	21.72	16	1,090	10.9%	1	1
LM (Wampler $\lambda = 1e-4$)	1	500	20.1	14	934	9.34%	1	1
LM (Wampler $\lambda = 1e-6$)	1	500	29.84	17	529	5.29%	1	1
LM (Chan $\lambda = 1.0$)	1	500	16.58	14	1,011	10.11%	1	1
LM (Chan $\lambda = 0.1$)	1	500	9.43	9	963	9.63%	1	1
LM (Sugihara $w_n = 1e-3$)	1	500	20.54	15	1,024	10.24%	1	1
LM (Sugihara $w_n = 1e-4$)	1	500	17.01	14	1,011	10.11%	1	1
NR	100	30	30.16	18	0	0%	1.47	25
GN	100	30	30.33	18	0	0%	1.48	23
NR pseudoinverse	100	30	30.27	18	0	0%	1.47	20
GN pseudoinverse	100	30	30.65	18	0	0%	1.49	20
LM (Wampler $\lambda = 1e-4$)	100	30	25.23	15	0	0%	1.35	17
LM (Wampler $\lambda = 1e-6$)	100	30	29.3	18	0	0%	1.45	24
LM (Chan $\lambda = 1.0$)	100	30	22.6	15	0	0%	1.25	18
LM (Chan $\lambda = 0.1$)	100	30	15.33	9	0	0%	1.2	18
LM (Sugihara $w_n = 1e-3$)	100	30	26.49	16	0	0%	1.35	18
LM (Sugihara $w_n = 1e-4$)	100	30	23.04	15	0	0%	1.26	18

ITER.: iterations.

measure of velocity isotropy. A condition number close to one means that the manipulator can achieve a velocity in a direction equally as easily as any other direction. However, a high condition number does not guarantee a high manipulability index where the manipulator may struggle to move in all directions.

CONCLUSIONS

In Part I of this tutorial, we have covered foundational aspects of manipulator differential kinematics. We first detailed a procedure for describing the kinematics of any manipulator and used this model to derive formulas for calculating the forward and first-order differential kinematics. We then detailed some applications unlocked by these formulas, including reactive motion control, IK, and methods that describe the performance of a manipulator at a given configuration. In Part II, we explore second-order differential kinematics and detail how it can improve applications detailed in Part I while also unlocking new applications.

ACKNOWLEDGMENT

We are grateful to the anonymous reviewers whose detailed and insightful comments have improved this article. This research was conducted by the Australian Research Council Project CE140100016 and supported by the QUT Centre for Robotics (QCR). We would also like to thank the members of QCR who provided valuable feedback and insights while testing this tutorial and associated Jupyter Notebooks.

AUTHORS

Jesse Haviland, Centre for Robotics, Queensland University of Technology, Brisbane, Queensland 4000, Australia. E-mail: j.haviland@qut.edu.au.

Peter Corke, Centre for Robotics, Queensland University of Technology, Brisbane, Queensland 4000, Australia. E-mail: peter.corke@qut.edu.au.

REFERENCES

[1] P. Corke, “A simple and systematic approach to assigning Denavit–Hartenberg parameters,” *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 590–594, Jun. 2007, doi: 10.1109/TRO.2007.896765.

[2] R. S. Hartenberg and J. Denavit, “A kinematic notation for lower-pair mechanisms based on matrices,” *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, Jun 1955, doi: 10.1115/1.4011045.

[3] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in Python*, 3rd ed. Berlin, Germany: Springer Nature, 2023.

[4] D. E. Whitney, “Resolved motion rate control of manipulators and human prostheses,” *IEEE Trans. Man-Mach. Syst.*, vol. 10, no. 2, pp. 47–53, Jun. 1969, doi: 10.1109/TMMS.1969.299896.

[5] S. Hutchinson, G. D. Hager, and P. Corke, “A tutorial on visual servo control,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996, doi: 10.1109/70.538972.

[6] P. Chang, “A closed-form solution for inverse kinematics of robot manipulators with redundancy,” *IEEE J. Robot. Autom.*, vol. 3, no. 5, pp. 393–403, Oct. 1987, doi: 10.1109/JRA.1987.1087114.

[7] T. Sugihara, “Solvability-unconcerned inverse kinematics by the Levenberg–Marquardt method,” *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 984–991, Oct. 2011, doi: 10.1109/TRO.2011.2148230.

[8] J. Haviland and P. Corke, “Maximising manipulability during resolved-rate motion control,” 2020, *arXiv:2002.11901.*

[9] D. Guo and Y. Zhang, “Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators,” *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6903–6914, Dec. 2014, doi: 10.1109/TIE.2014.2331036.

[10] J. Haviland and P. Corke, “NEO: A novel expeditious optimisation algorithm for reactive motion control of manipulators,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1043–1050, Apr. 2021, doi: 10.1109/LRA.2021.3056060.

[11] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY, USA: Springer-Verlag, 1986, pp. 396–404.

[12] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, “Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields,” in *Proc. IEEE Int. Conf. Humanoid Robots*, 2008, pp. 91–98, doi: 10.1109/ICHR.2008.4755937.

[13] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, “A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks,” in *Proc. IEEE Int. Conf. Adv. Robot.*, 2009, pp. 1–6.

[14] L. J. Stocco, S. E. Salcudean, and F. Sassani, “On the use of scaling matrices for task-specific robot design,” *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 958–965, Oct. 1999, doi: 10.1109/70.795800.

[15] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon Univ., Robot. Inst., Pittsburgh, PA, USA, Aug. 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[16] A. S. Deo and I. D. Walker, “Adaptive non-linear least squares for inverse kinematics,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 1993, pp. 186–193, doi: 10.1109/ROBOT.1993.291981.

[17] C. W. Wampler, “Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods,” *IEEE Trans. Syst., Man, Cybern.*, vol. 16, no. 1, pp. 93–101, Jan. 1986, doi: 10.1109/TSMC.1986.289285.

[18] S. K. Chan and P. D. Lawrence, “General inverse kinematics with the error damped pseudoinverse,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 1988, pp. 834–839, doi: 10.1109/ROBOT.1988.12164.

[19] S. Patel and T. Sobh, “Manipulator performance measures-a comprehensive literature survey,” *J. Intell. Robot. Syst.*, vol. 77, no. 3, pp. 547–570, Mar. 2015, doi: 10.1007/s10846-014-0024-y.

[20] T. Yoshikawa, “Manipulability of robotic mechanisms,” *Int. J. Robot. Res.*, vol. 4, no. 2, pp. 3–9, 1985, doi: 10.1177/027836498500400201.

[21] J. K. Salisbury and J. J. Craig, “Articulated hands: Force control and kinematic issues,” *Int. J. Robot. Res.*, vol. 1, no. 1, pp. 4–17, 1982, doi: 10.1177/027836498200100102.