June 12, 2021

On the understanding of rotations and inverse kinematics

Mattias Fredriksson

Contents

1 Introduction

Understanding the mathematics behind rotations is fundamental when looking onto the topic of rigid body kinematics. The purpose for this document is therefore to formulate key mathematical components for solving the inverse kinematics problem analytically, using numerical optimization methods.

Foundation for the document lies primarily in [5]. However, as the book provides an intuitive approach to robotics focusing on the concepts of twist and exponential coordinates in $\mathbb{R}^6$. The focus here lies on the use of independent coordinates for solving positioning and orientation of the end effectors. And uses an analytical rather than geometrical Jacobian to facilitate testing using numerical differentiation methods.

Lastly, the goal was also to derive a few algorithms applicable to solving different inverse kinematics problems. And each algorithm presented in [*section*.4] is designed for simplicity and straight forward implementation in Numpy. Even if the goal was for me to understand the mathematics behind inverse kinematics, hopefully you will find some use in it.

2 Rotations

2.1 Begin

Understanding rotations are a fundamental part in understanding rigid body movement. For many rigid body systems, rotational movement in joints are significant compared to translational movement. Considering the case of modeling the human body as a system of rigid body links. Joints connecting the links modeling the underlying skeletal structure exhibit significant rotational degrees of freedom (DoF) but involve limited translational movement.

Due to the fundamental importance, understanding rotational movement are key for understanding rigid body kinematics. As such, following sections are not concerned about different Euler or quaternion parameterizations, but of separating rotations in parallel and orthogonal parts, expressing rotations on exponential and logarithm form, and derivatives of rotations.

2.2 Rotations in 2D

Understanding rotations within a 2-dimensional plane is important not only as a steppingstone for understanding the 3-dimensional case, but it also shares fundamental properties which translate into the higher dimension. All rotations are equivalent to a basis change in which distances between points are preserved, and the linear transformation matrix for a planar rotation of angle α is

$$R = \begin{vmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{vmatrix} \tag{1}$$

A rotation matrix then define a reorientation of the basis as shown in *fig.* 1a, where the rotation R moves all points in a circular motion around the origin. By taking the derivative of each matrix element with respect to change in angle, one gets the derivative

$$\frac{\delta R}{\delta \alpha} = \begin{vmatrix} -\sin(\alpha) & -\cos(\alpha) \\ \cos(\alpha) & -\sin(\alpha) \end{vmatrix} \tag{2}$$

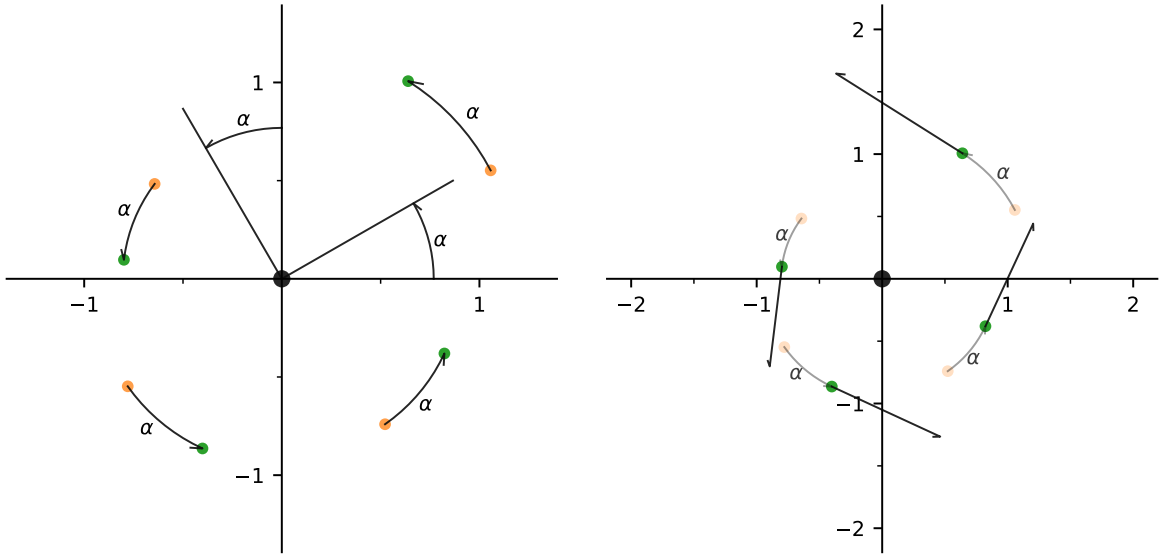
Derivative of R with respect to α is then the tangent of the circle as shown in *fig.* 1b. Reiterating definitions of a planar rotation might not be of interest, the description do however provide an overview in *fig.* 1 when moving into the third dimension.

2.3 Rotations in 3D

A common representation for rotations is the rotation composition around the unit axis parameterized using Euler angles. While Euler angles are useful, a better definition of a rotational transform is as the rotation by an angle α around an axis $\hat{\boldsymbol{k}}$. The axis-angle view of rotations provide a simpler geometrical perspective, and has the implication that when decomposing a vector $\boldsymbol{v}$ into a parallel part $\boldsymbol{v}_{\parallel}$ and orthogonal part $\boldsymbol{v}_{\perp}$ relative to the axis $\hat{\boldsymbol{k}}$, only the orthogonal part $\boldsymbol{v}_{\perp}$ change during rotation around the axis. Some of the geometrical relationships for the decomposition of $\boldsymbol{v}$ can then be summarized by the following equations

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{v}_{\parallel} + \boldsymbol{v}_{\perp} \\ ||\hat{\boldsymbol{k}}|| &= 1 \\ \boldsymbol{v}_{\parallel} &= \hat{\boldsymbol{k}}\langle\hat{\boldsymbol{k}}, \boldsymbol{v}\rangle \\ \boldsymbol{v}_{\perp} &= \boldsymbol{v} - \hat{\boldsymbol{k}}\langle\hat{\boldsymbol{k}}, \boldsymbol{v}\rangle \\ R\boldsymbol{v}_{\parallel} &= \boldsymbol{v}_{\parallel} \end{aligned} \tag{3}$$

Here $\langle \cdot, \cdot \rangle$ represents the vector dot product, and $||\boldsymbol{x}|| = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$ the vector norm. With the perpendicular vector part projected onto the subspace orthogonal to the rotation axis, which is



(a) $R(\alpha)$ (b) $\delta R(\alpha)/\delta\alpha$

Figure 1: Depiction of change for a set of points $\boldsymbol{x} \in \mathbb{R}^2$ rotating around origin by the angle α. The left image highlights the circular movement of the rotation from $\boldsymbol{x}$ (yellow) to $R(\alpha)\boldsymbol{x}$ (green), as well as the movement of the unit axes. The right image shows the derivative, or tangent vectors (black lines), for the rotating points as offsets from the position $R(\alpha)\boldsymbol{x}$.

the plane consisting of all $\boldsymbol{v} \in \mathbb{R}^3$ satisfying the equation $\langle \hat{\boldsymbol{k}}, \boldsymbol{v} \rangle = 0$, the problem of rotating $\boldsymbol{v}_\perp$ are reduced to the 2-dimensional problem discussed in the previous section.

The only remaining issue is to find a basis within the plane in which to solve the 2-dimensional problem. Since any orthogonal pair of vectors $|\boldsymbol{b}_x, \boldsymbol{b}_y|$ within the plane will do, the simplest solution is to set $\boldsymbol{b}_x = \boldsymbol{v}_\perp$ and use the cross product to get $\boldsymbol{b}_y = \hat{\boldsymbol{k}} \times \boldsymbol{v}_\perp$. With $\boldsymbol{v}_\perp$ represented by $|1, 0|^T$ within the plane basis, utilization of *Eq.* 2 from previous section and moving the solution back into $\mathbb{R}^3$ gives

$$R\boldsymbol{v} = \cos(\alpha)\boldsymbol{b}_x + \sin(\alpha)\boldsymbol{b}_y = \cos(\alpha)\boldsymbol{v}_\perp + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v}_\perp \tag{4}$$

and since

$$R\boldsymbol{v}_\perp = R(\boldsymbol{v}_{\parallel} + \boldsymbol{v}_\perp) = R\boldsymbol{v}_{\parallel} + R\boldsymbol{v}_\perp \tag{5}$$

the geometrical solution to rotating a vector using axis angle parameters can be summarized as

$$R\boldsymbol{v} = \boldsymbol{v}_{\parallel} + \cos(\alpha)\boldsymbol{v}_\perp + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v}_\perp \tag{6}$$

Rotation of a vector as described above are shown in *fig.* 2, and it can be worth to point out that in the equations above $\boldsymbol{v}_\perp$ is not of unit norm, but the relationships hold due to the equivalence of $\|\boldsymbol{v}_\perp\| = \|\hat{\boldsymbol{k}} \times \boldsymbol{v}_\perp\|$. While the simple geometrical solution might be elegant, finding a general solution on matrix form has more applications and will be discussed in the next section.

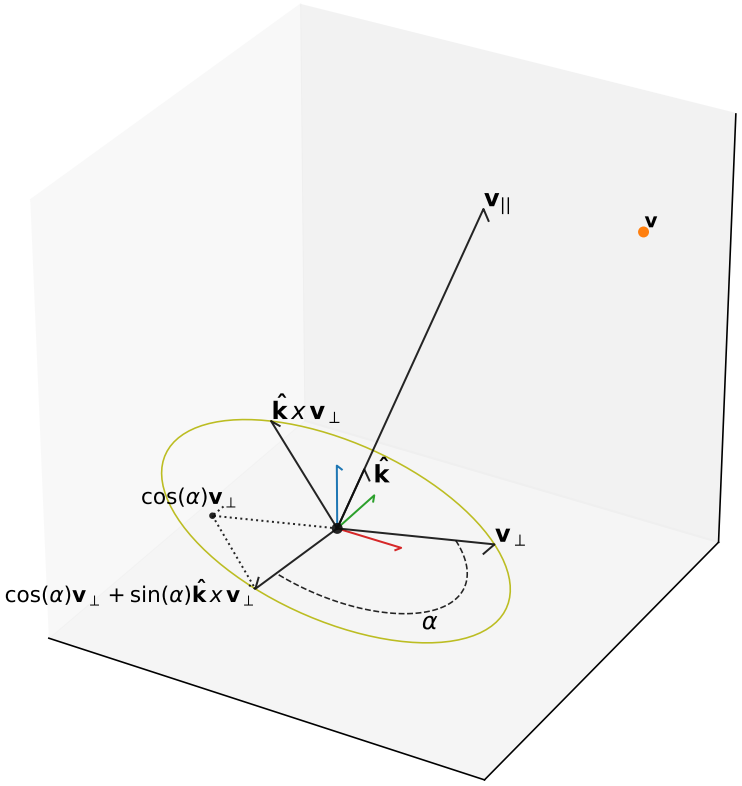


Figure 2: Geometrical relationships for rotating vector $\boldsymbol{v}$ around the axis $\hat{\boldsymbol{k}}$ with angle $\alpha = -\frac{7}{9}\pi$.

2.4 Rodrigues' rotation formula

Finding the rotation matrix associated with an axis-angle rotation can be achieved by reformulating *Eq.* 6 and finding the matrix associated with each term. The solution is the matrix form of the equation associated with Rodrigues' rotation formula

$$R(\alpha) = I + \sin(\alpha)K + (1 - \cos(\alpha))K^2 \tag{7}$$

where K is the skew-symmetric matrix $[\hat{\boldsymbol{k}}]_\times$ defined by the cross product satisfying $K\boldsymbol{v} = \hat{\boldsymbol{k}} \times \boldsymbol{v}$ and has the form

$$K = [\hat{\boldsymbol{k}}]_\times = \begin{vmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{vmatrix} \tag{8}$$

Deriving the rotation matrix for the linear transform associated with the axis-angle rotation is now straight forward. Consider *Eq.* 7 for rotating a vector $\boldsymbol{v}$

$$R\boldsymbol{v} = (I + \sin(\alpha)K + (1 - \cos(\alpha))K^2)\boldsymbol{v} = \boldsymbol{v} + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v} + \cos(\alpha)\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) \tag{9}$$

The right hand side of the equation is similar to *Eq.* 6 except to the fact that every occurrence of $\boldsymbol{v}$ is moved to the right to allow factorization, and $\boldsymbol{v}$ is no longer separated in a parallel and perpendicular part. To find the steps involved for converting the formulation in *Eq.* 6, a few geometrical identities for $\boldsymbol{v}_{\parallel}$ and $\boldsymbol{v}_{\perp}$ are helpful:

$$\hat{\boldsymbol{k}} \times \boldsymbol{v}_{\parallel} = 0 \implies \hat{\boldsymbol{k}} \times \boldsymbol{v} = \hat{\boldsymbol{k}} \times (\boldsymbol{v}_{\perp} + \boldsymbol{v}_{\parallel}) = \hat{\boldsymbol{k}} \times \boldsymbol{v}_{\perp} \tag{10}$$

which gives

$$\boldsymbol{v}_{\perp} = \hat{\boldsymbol{k}} \times (\hat{\boldsymbol{v}}_{\perp} \times \hat{\boldsymbol{k}}) = -\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}_{\perp}) = -\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) \tag{11}$$

and in turn

$$\boldsymbol{v}_{\parallel} = \boldsymbol{v} - \boldsymbol{v}_{\perp} = \boldsymbol{v} + \hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) \tag{12}$$

The equations are not controversial in any form, but inserting the results into *Eq.* 6 gives

$$\begin{aligned} R\boldsymbol{v} &= \boldsymbol{v}_{\parallel} + \cos(\alpha)\boldsymbol{v}_{\perp} + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v}_{\perp} \\ &= \boldsymbol{v} + \hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) - \cos(\alpha)\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v} \\ &= \boldsymbol{v} + \sin(\alpha)\hat{\boldsymbol{k}} \times \boldsymbol{v} + (1 - \cos(\alpha))\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) \end{aligned} \tag{13}$$

The rightmost side of *Eq.* 13 is now identical to *Eq.* 9, and is the geometrical solution for deriving the rotation matrix associated with an axis-angle rotation. The geometrical relationships involved are highlighted in *fig.* 3.

2.5 Derivative

Differentiating *Eq.* 7 with respect to angle yields

$$\frac{\delta R}{\delta \alpha} = \cos(\alpha)K + \sin(\alpha)K^2 \tag{14}$$

However, the rotation derivative in *Eq.* 14 can also be written as a function of the cross-product due to the equivalence of

$$\begin{aligned} \frac{\delta R}{\delta \alpha} = KR &= K(I + \sin(\alpha)K + (1 - \cos(\alpha))K^2) \\ &= K + \sin(\alpha)K^2 + (1 - \cos(\alpha))K^3 \\ &= K + \sin(\alpha)K^2 - K + \cos(\alpha)K \\ &= \cos(\alpha)K + \sin(\alpha)K^2 \end{aligned} \tag{15}$$

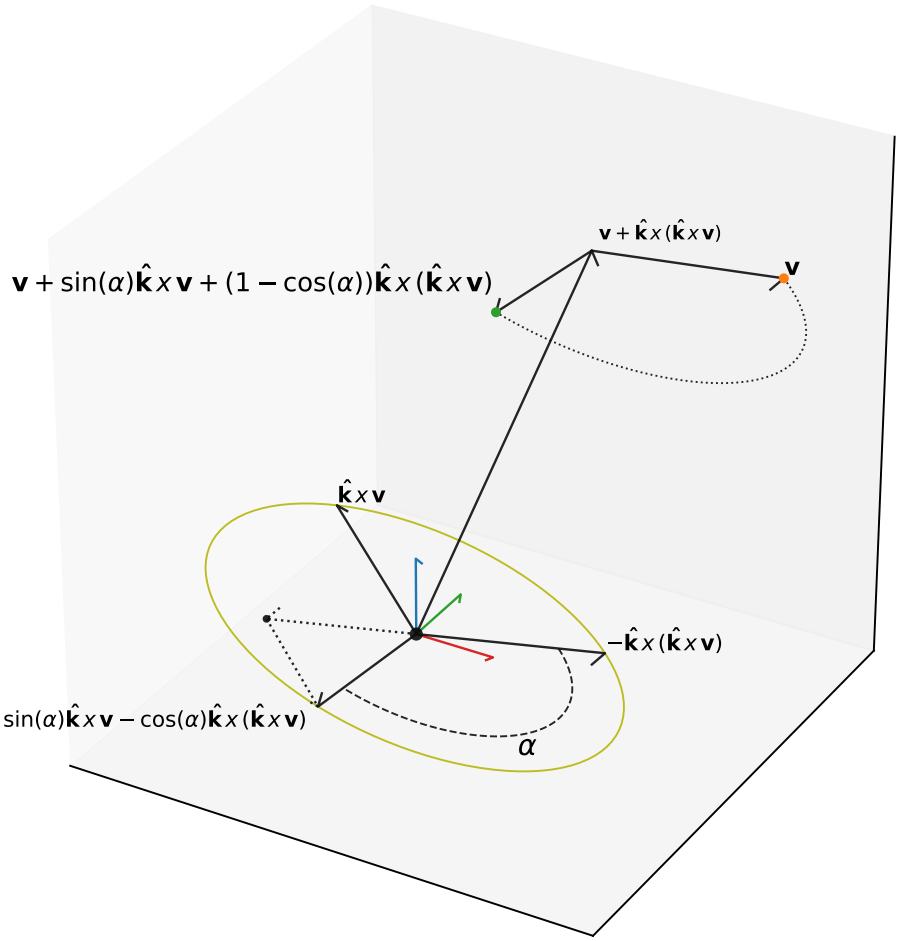


Figure 3: Relationships used in the factorization of $\boldsymbol{v}$ when rotating around the axis $\hat{\boldsymbol{k}}$.

Eq. 15 further highlight the derivative as the tangent line to the circle as shown for the 2-dimensional case in *fig.* 1b, the only difference is that the circle lies within the plane orthogonal to the rotation axis. Following up on *Eq.* 15, as the cross-product forms cyclical patterns for powers of K, K^3 can be simplified with

$$\begin{aligned} K^3 &= -K \\ K^4 &= -K^2 \end{aligned} \tag{16}$$

which follows from the association to the cross-product. In particular, with $\|\hat{\boldsymbol{k}}\| = 1$ vectors perpendicular to $\hat{\boldsymbol{k}}$ are reflected as shown by

$$K^3\boldsymbol{v} = \hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v})) = -\hat{\boldsymbol{k}} \times \boldsymbol{v} \tag{17}$$

It is important to not forget that only the perpendicular part is reflected since

$$\hat{\boldsymbol{k}} \times (\hat{\boldsymbol{k}} \times \boldsymbol{v}) = -\boldsymbol{v} \implies \langle \hat{\boldsymbol{k}}, \boldsymbol{v} \rangle = 0 \tag{18}$$

which is true for any vector transformed by the cross-product as it is orthogonal to both $\boldsymbol{v}$ and $\hat{\boldsymbol{k}}$.

2.6 Exponential

Expressing rotation matrices on exponential form, or rather as a product of infinitesimal generators, might not be intuitive. However, a good introduction to the subject is given in [5]. Essentially, matrix exponentials are a natural expression for axis-angle rotations using the special case of linear differential equations associated with the natural number e. Consider a function $y(x) \in \mathbb{R}$ where the derivative with respect to x is equivalent to

$$\frac{\delta y}{\delta x} = ay(x) = ae^{ax} \tag{19}$$

The matrix exponential e^{Ax} is a generalization for $A \in \mathbb{R}^{n \times n}$ where the derivative of the function $Y(x) \in \mathbb{R}^N$ can be expressed on the form

$$\frac{\delta Y}{\delta x} = AY(x) = Ae^{Ax} \tag{20}$$

Applying these concepts to rotation matrices one can identify that the derivative with respect to angle in *Eq.* 15 can be expressed as an exponential

$$\frac{\delta R}{\delta \alpha} = KR(\alpha) = Ke^{K\alpha} = e^{K\alpha}K \tag{21}$$

where the rotation matrix on exponential form is defined by

$$e^{K\alpha} = I + K\alpha + \frac{(K\alpha)^2}{2!} + \frac{(K\alpha)^3}{3!} + \cdots = I + \sin(\alpha)K + (1 - \cos(\alpha))K^2 \tag{22}$$

Explanation for the mathematics involved are better approached in sources such as [5]. To quickly summarize, definition of the rotation matrix can be derived from the exponential form as a sum of an infinitive series, and it can be shown that the series converge to the closed form solution in the Rodrigues' formula as shown in *Eq.* 22. It is also important to note that the rotation exponential does not share all properties of the natural number but share the properties for rotation compositions. One important property for the exponential occurs when exponentials commute

$$e^{A\alpha}e^{B\theta} = e^{A\alpha + B\theta} \iff e^{A\alpha}e^{B\theta} = e^{B\theta}e^{A\alpha} \tag{23}$$

While rotations seldom commute except when rotating around the same axis, the property is important when exponentials commute. Another important property to remember is the rotation inverse

$$R(\alpha)^T = R(\alpha)^{-1} = (e^{A\alpha})^{-1} = e^{-A(\alpha)} \tag{24}$$

Finally, the exponential defines the rotation matrix as a function of the axis-angle parameters and are interchangeable. While simplifying the form for expressing axis-angle rotations, it also provide a parameterization for rotations with minimal number of independent parameters in the *exponential coordinates*. Since exponential coordinates are determined through the logarithmic function acting as the inverse to the exponential, it will be covered in the next section.

2.7 Logarithm

The exponential provide the mathematics for parameterizing rotations using *exponential coordinates* which is equivalent to the rotation vector/axis-angle representation

$$\hat{\boldsymbol{\omega}}\alpha \in \mathbb{R}^3 \implies e^{[\hat{\boldsymbol{\omega}}]_\times \alpha} = R(\hat{\boldsymbol{\omega}}, \alpha) \tag{25}$$

Exponential coordinates, in similarity with Euler angles, define rotations using a minimal number of independent parameters (implying there are no dependent or 'redundant' parameters). The elegance of the coordinate representation is in the uniqueness of axis-angle rotations in the range $0 \leq \alpha < \pi$, with discontinuity occurring at $\alpha = \pi$ due to the equivalence of rotating π radians around either $\pm\hat{\boldsymbol{k}} \in \mathbb{R}^3$. Combined with the geometrical interpretation, exponential coordinates provide an intrinsic parameterization relative to Euler angles.

Importance of a matrix logarithm as the inverse to the exponential then become apparent, both for finding the derivative in *Eq.* 15, but also for extracting the exponential coordinates for composite rotations. To find a function for the logarithm one can approach the problem by analyzing Rodrigues' formula in [*section.*2.4], and the skew-symmetric matrices involved

$$K = [\hat{\boldsymbol{k}}\alpha] = \begin{vmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{vmatrix} \text{ and } K^2 = [\hat{\boldsymbol{k}}\alpha]^2 = \begin{vmatrix} k_x^2 - 1 & k_y k_x & k_z k_x \\ k_x k_y & k_y^2 - 1 & k_z k_y \\ k_x k_z & k_y k_z & k_z^2 - 1 \end{vmatrix} \tag{26}$$

Utilizing the anti-symmetry of K and the symmetry of K^2 for some rotation $R = R(\hat{\boldsymbol{k}}, \alpha)$ with unknown axis-angle parameters yield

$$\begin{aligned} R - R^T &= (I + \sin(\alpha)K + (1 - \cos(\alpha))K^2) - (I - \sin(\alpha)K + (1 - \cos(\alpha))K^2) \\ &= 2\sin(\alpha)K \end{aligned} \tag{27}$$

leaving the skew-matrix as a function of the angle. Finding the angle is then a problem of analyzing the diagonal as the traces for the first parts are equivalent to $tr(I) = 3$ and $tr(K) = 0$. Trace for R is then determined by K^2 and is equivalent to

$$tr(R) = 3 + (1 - \cos(\alpha))(k_x^2 + k_y^2 + k_z^2 - 3) = 3 - 2(1 - \cos(\alpha)) = 1 - 2\cos(\alpha) \tag{28}$$

where the equation can be simplified due to the unit norm. The angle α is then

$$\alpha = \cos^{-1}(\frac{tr(R) - 1}{2}) \tag{29}$$

and the logarithm in the general case is

$$[\hat{\boldsymbol{k}}\alpha] = \frac{1}{2 - \sin(\alpha)}(R - R^T) \tag{30}$$

Edge cases for the logarithm occur for the identify rotation where $e^0 = I \iff \alpha = 0$ and for an angle of π follows $\sin(\pi)[\hat{\boldsymbol{k}}\alpha] = 0$. The first case is simple to solve since $[\hat{\boldsymbol{k}}\alpha] = 0$ for which $\hat{\boldsymbol{k}}$ can be satisfied by any $\hat{\boldsymbol{k}} \in \mathbb{R}^3$. The second edge case occurring at $\alpha = \pi$ is not as simple, but cancelation for the cosine and sine for an angle π yield

$$R(\hat{\boldsymbol{k}}, \pi) = I + 2K^2 = \begin{vmatrix} 2k_x^2 - 1 & 2k_y k_x & 2k_z k_x \\ 2k_x k_y & 2k_y^2 - 1 & 2k_z k_y \\ 2k_x k_z & 2k_y k_z & 2k_z^2 - 1 \end{vmatrix} \tag{31}$$

Each column R_i in the matrix $I + 2K^2$ then give an equation for finding $\hat{\boldsymbol{k}}$ since

$$\frac{1}{2}(R_i + \boldsymbol{e}_i) = \hat{\boldsymbol{k}}_i\hat{\boldsymbol{k}} \implies \hat{\boldsymbol{k}} = \frac{1}{\sqrt{2(R_{i,i} + 1)}}(R_i + \boldsymbol{e}_i) \tag{32}$$

Optimally, the column for finding $\hat{\boldsymbol{k}}$ is chosen as the one with largest diagonal entry to avoid cases where $\hat{\boldsymbol{k}}_i = 0$. Particularly, as the column approach one of the negative unit axis $-\boldsymbol{e}_i$, the diagonal entry approaches -1, and the columns contain no solution for finding $\hat{\boldsymbol{k}}$ using *Eq.* 32. Instead, the case represents a rotation around one of the other unit axes, and can only be determined using *Eq.* 32 for the column containing the rotation axis (which is equivalent to one of the positive unit axes).

2.8 Properties of rotation compositions

An important consideration in application, is the rotation defined within a context and composed from a rotation series. Two fundamental equalities for a rotation composition is

- First, expressing the rotation derivative as a product of a skew and rotation composition when the derivative is computed for a rotation occurring within an intermediate frame in the composition.
- Second, expressing the rotation series as a product of a rotation and remaining rotation composition, for any rotation within the composition.

Both equalities are discussed in [5], but the importance can't be understated. Before diving into the equalities, the rotation composition is a rotation series on the form

$$R = R_0 R_1 \cdots R_n = e^{[\boldsymbol{\omega}_0]_\times} e^{[\boldsymbol{\omega}_1]_\times} \cdots e^{[\boldsymbol{\omega}_n]_\times} \tag{33}$$

2.8.1 Derivative in the fixed frame

Consider the partial derivative for a rotation within the composition

$$\frac{\delta R}{\delta \alpha} = e^{[\boldsymbol{\omega}_0]_\times} e^{[\boldsymbol{\omega}_1]_\times} \cdots [\hat{\boldsymbol{k}}]_\times e^{[\hat{\boldsymbol{k}}]_\times \alpha} \cdots e^{[\boldsymbol{\omega}_n]_\times} = R_0 [\hat{\boldsymbol{k}}]_\times e^{[\hat{\boldsymbol{k}}]_\times \alpha} R_e \tag{34}$$

The skew-matrix has an equivalent representation in the fixed frame which satisfies the equivalence

$$\begin{aligned} R_0 [\hat{\boldsymbol{k}}]_\times e^{[\hat{\boldsymbol{k}}]_\times \alpha} R_e = [\hat{\boldsymbol{\omega}}]_\times R_0 e^{[\hat{\boldsymbol{k}}]_\times \alpha} R_e \implies \\ R_0 [\hat{\boldsymbol{k}}]_\times = [\hat{\boldsymbol{\omega}}]_\times R_0 \end{aligned} \tag{35}$$

Equation for the skew-symmetric matrix satisfying the equivalence in the right-most equation is

$$\begin{aligned} [\hat{\boldsymbol{\omega}}]_\times = R_0 [\hat{\boldsymbol{k}}]_\times R_0^T &= R_0 \begin{vmatrix} \hat{\boldsymbol{k}} \times (R_0^T)_x & \hat{\boldsymbol{k}} \times (R_0^T)_y & \hat{\boldsymbol{k}} \times (R_0^T)_z \end{vmatrix} \\ &= \begin{vmatrix} \langle \boldsymbol{v}_x, \hat{\boldsymbol{k}} \times \boldsymbol{v}_x \rangle & \langle \boldsymbol{v}_x, \hat{\boldsymbol{k}} \times \boldsymbol{v}_y \rangle & \langle \boldsymbol{v}_x, \hat{\boldsymbol{k}} \times \boldsymbol{v}_z \rangle \\ \langle \boldsymbol{v}_y, \hat{\boldsymbol{k}} \times \boldsymbol{v}_x \rangle & \langle \boldsymbol{v}_y, \hat{\boldsymbol{k}} \times \boldsymbol{v}_y \rangle & \langle \boldsymbol{v}_y, \hat{\boldsymbol{k}} \times \boldsymbol{v}_z \rangle \\ \langle \boldsymbol{v}_z, \hat{\boldsymbol{k}} \times \boldsymbol{v}_x \rangle & \langle \boldsymbol{v}_z, \hat{\boldsymbol{k}} \times \boldsymbol{v}_y \rangle & \langle \boldsymbol{v}_z, \hat{\boldsymbol{k}} \times \boldsymbol{v}_z \rangle \end{vmatrix} \end{aligned} \tag{36}$$

where $\boldsymbol{v}_x = (R_0^T)_x$ is the first row in R_0 and is equivalent to the first column of R_0^T. Using the scalar triple-product equivalence $\langle \boldsymbol{a}, \boldsymbol{b} \times \boldsymbol{c} \rangle = \langle \boldsymbol{b}, \boldsymbol{a} \times \boldsymbol{c} \rangle$, *Eq.* 36 can be simplified to

$$[\hat{\boldsymbol{\omega}}]_\times = \begin{vmatrix} 0 & -\langle \hat{\boldsymbol{k}}, \boldsymbol{v}_z \rangle & \langle \hat{\boldsymbol{k}}, \boldsymbol{v}_y \rangle \\ \langle \hat{\boldsymbol{k}}, \boldsymbol{v}_z \rangle & 0 & -\langle \hat{\boldsymbol{k}}, \boldsymbol{v}_x \rangle \\ -\langle \hat{\boldsymbol{k}}, \boldsymbol{v}_y \rangle & \langle \hat{\boldsymbol{k}}, \boldsymbol{v}_x \rangle & 0 \end{vmatrix} = [R_0 \hat{\boldsymbol{k}}] \tag{37}$$

Implication from the equivalence in *Eq.* 37 is then that the partial derivative in a composition can always be expressed using the skew-symmetric matrix with the rotation axis moved into the fixed frame.

2.8.2 Rotation in the fixed frame

Starting of in a similar fashion to the last section, for a rotation composition there exists some rotation satisfying the equivalence

$$e^{[\hat{\omega}]_\times \alpha} R_0 R_e = e^{\omega_0} e^{\omega_1} \cdots e^{[\hat{k}]_\times \alpha} \cdots e^{\omega_n} = R_0 e^{[\hat{k}]_\times \alpha} R_e \tag{38}$$

To find a solution, replacement of $e^{[\hat{k}]_\times \alpha}$ with the Rodriguez's formula yield

$$\begin{aligned} e^{[\hat{\omega}]_\times \alpha} &= R_0 e^{[\hat{k}]_\times \alpha} R_0^T \\ &= R_0 (I + \sin(\alpha)[\hat{k}]_\times + (1 - \cos(\alpha))[\hat{k}]_\times^2) R_0^T \\ &= R_0 I R_0^T + \sin(\alpha) R_0 [\hat{k}]_\times R_0^T + (1 - \cos(\alpha)) R_0 [\hat{k}]_\times^2 R_0^T \\ &= I + \sin(\alpha)[R_0 \hat{k}]_\times + (1 - \cos(\alpha))[R_0 \hat{k} R_0^T]_\times^2 \\ &= e^{[R_0 \hat{\omega}]_\times \alpha} \end{aligned} \tag{39}$$

With the equivalence for $R_0[\hat{k}]_\times R_0^T = [R_0 \hat{k}]_\times$ derived in the previous section. Equivalence to $R_0[\hat{k}]_\times^2 R_0^T$ can be determined through substitution with the final expression

$$[R_0 \hat{k}]_\times^2 = (R_0 [\hat{k}]_\times R_0^T)^2 = (R_0 [\hat{k}]_\times R_0^T)(R_0 [\hat{k}]_\times R_0^T) = R_0 [\hat{k}]_\times^2 R_0^T \tag{40}$$

Thus, the right-hand side is equivalent to the left-most side as the inner product terms cancels out. The fixed frame equivalent for a rotation is then the rotation around the rotation axis moved into the fixed frame as it satifies

$$e^{[R_0 \hat{\omega}]_\times \alpha} R_0 = R_0 e^{[\hat{\omega}]_\times \alpha} \tag{41}$$

2.8.3 Angular velocity

A crucial part in defining the motion for a rigid body system is in understanding angular velocity, or the instantaneous rate of rotation around an axis. Considering the rotation derivative in [*section*.2.5]

$$\frac{\delta R}{\delta \alpha} = [\hat{k}\alpha]_\times R \tag{42}$$

the angular velocity with respect to time is

$$[\hat{k}\alpha]_\times = \frac{\delta R}{\delta t} R \tag{43}$$

Equivalently the angular velocity with respect to angle is

$$[\hat{k}]_\times = \frac{\delta R}{\delta \alpha} R \tag{44}$$

and can be considered as a system rotating 1 radian per unit of change or as the system response with respect to its independent parameters. While the left-hand side seems like an association to the skew-matrix $[\hat{k}]_\times$, the concepts are distinct. Angular velocity represents a unit of change while the skew-matrix is simply its quantity. One of the most important aspects in the distinction is additivity for velocities expressed in the same reference frame. Considering a system defined by a rotation composition where the rotation at time t is

$$R(t) = e^{[\hat{k}_1]_\times \alpha_1 t} e^{[\hat{k}_2]_\times \alpha_2 t} \cdots e^{[\hat{k}_n]_\times \alpha_n t} \tag{45}$$

Additivity yield the angular velocity for the composition as

$$\hat{\boldsymbol{k}}_1\alpha_1 + \hat{\boldsymbol{k}}_2\alpha_2 + \cdots + \hat{\boldsymbol{k}}_n\alpha_n \tag{46}$$

2.8.4 Derivation additivity for simple angular velocities

Deriving additivity for angular velocities highlights the importance of the basis in which the velocity is defined in. Consider a rotation composition as a system at configuration $\boldsymbol{\alpha}$ under the change δ, factoring δ yield

$$\begin{aligned} R(\boldsymbol{\alpha}+\delta) &= e^{[\hat{\boldsymbol{k}}_1]_\times(\alpha_1+\delta)}e^{[\hat{\boldsymbol{k}}_2]_\times(\alpha_2+\delta)}\cdots e^{[\hat{\boldsymbol{k}}_n]_\times(\alpha_n+\delta)} \\ &= e^{[\hat{\boldsymbol{\omega}}_1]_\times\delta}e^{[\hat{\boldsymbol{\omega}}_2]_\times\delta}\cdots e^{[\hat{\boldsymbol{\omega}}_n]_\times\delta}e^{[\hat{\boldsymbol{k}}_1]_\times\alpha_1}e^{[\hat{\boldsymbol{k}}_2]_\times\alpha_2}\cdots e^{[\hat{\boldsymbol{k}}_n]_\times\alpha_n} \end{aligned} \tag{47}$$

where the rotation axis $\boldsymbol{\omega}_i$ is the rotation axis moved into the spatial frame in accordance with *Eq.* 41 in [*section.*2.8.2]. Consider our instantaneous system configuration at the limit when δ approaches 0, rotations represented by $e^{\hat{\boldsymbol{\omega}}_i\delta}$ become infinitesimal rotations and commute. Thus as δ approaches 0, *Eq.* 47 approach

$$\begin{aligned} \lim_{\delta\to 0} R(\boldsymbol{\alpha}+\delta) &= \lim_{\delta\to 0} e^{[\hat{\boldsymbol{\omega}}_1+\hat{\boldsymbol{\omega}}_2+\cdots+\hat{\boldsymbol{\omega}}_n]_\times\delta}R(\boldsymbol{\alpha}) \\ &= \lim_{\delta\to 0} e^{[\boldsymbol{\omega}]_\times\delta}R_0 \end{aligned} \tag{48}$$

where R_0 is used to simplify the system expression for the instantaneous system configuration $\boldsymbol{\alpha}$. Angular velocity for the system can be expressed in accordance with *Eq.* 44 as

$$\frac{R(\boldsymbol{\alpha}+\delta)-R(\boldsymbol{\alpha})}{\delta}R(\boldsymbol{\alpha})_0^T = \frac{e^{[\boldsymbol{\omega}]_\times\delta}R_0R_0^T - R_0R_0^T}{\delta} = \frac{e^{[\boldsymbol{\omega}]_\times\delta}I - I}{\delta} \tag{49}$$

And at the limit $\delta \to 0$, the rotation $e^{[\boldsymbol{\omega}]_\times\delta}$ is the infinitesimal rotation equivalent to the first order Taylor polynomial for the exponential

$$\begin{aligned} \lim_{\delta\to 0} e^{[\boldsymbol{\omega}]_\times\delta} &= \lim_{\delta\to 0} I + [\boldsymbol{\omega}\delta]_\times + [\boldsymbol{\omega}\delta]_\times^2 \\ &\approx I + [\boldsymbol{\omega}\delta]_\times = \begin{vmatrix} 1 & -\omega_z\delta & \omega_y\delta \\ \omega_z\delta & 1 & -\omega_x\delta \\ -\omega_y\delta & \omega_x\delta & 1 \end{vmatrix} \end{aligned} \tag{50}$$

with the approximate equivalence to the infinitesimal rotation expanded using the Rodrigues' formula follow from ignoring second order terms when δ approaches 0 [6]. Replacement of the exponential in *Eq.* 49 with the infinitesimal rotation gives

$$\lim_{\delta\to 0}\frac{e^{[\boldsymbol{\omega}]_\times\delta}-I}{\delta} = \lim_{\delta\to 0}\frac{\delta[\boldsymbol{\omega}]}{\delta} = [\boldsymbol{\omega}]_\times = \left[\sum_{i=0}^{n}\boldsymbol{\omega}_i\right]_\times \tag{51}$$

Thus, the total angular velocity in the fixed frame can be expressed as a sum of the angular velocities moved into the same frame. Finally, commutation for infinitesimal rotations follows from ignoring second order terms [6], as the product of infinitesimal rotations approaches the sum

$$\begin{aligned} \lim_{\delta\to 0} e^{\hat{\boldsymbol{\omega}}_1\delta}e^{\hat{\boldsymbol{\omega}}_2\delta} &= \lim_{\delta\to 0}(I + [\boldsymbol{\omega}_1]_\times\delta)(I + [\boldsymbol{\omega}_2]_\times\delta) \\ &= \lim_{\delta\to 0}(I + [\boldsymbol{\omega}_2]_\times\delta)(I + [\boldsymbol{\omega}_1]_\times\delta) \\ &\approx I + [\boldsymbol{\omega}_1]_\times\delta + [\boldsymbol{\omega}_2]_\times\delta \end{aligned} \tag{52}$$

Approach for deriving additivity of angular velocities provides fundamental properties for rotations. One could however also consider the case of substituting the system using some scalar $\alpha = 0$. Thus, the instantaneous system configuration could be expressed as

$$R(\alpha) = R(\boldsymbol{\alpha} + \mathbb{1}\alpha) = \prod_{i=0}^{n} e^{[\hat{\boldsymbol{k}}_i]_\times (\alpha_i + \alpha)} = \prod_{i=0}^{n} e^{[\hat{\boldsymbol{k}}_i]_\times \alpha_i} e^{[\hat{\boldsymbol{k}}_i]_\times \alpha} \tag{53}$$

Derivative with respect to the scalar variable α can then be determined through the product rule. And due to additivity of the skew matrix at the identity element, the approach yield the same result as above.

3 Rigid body systems

A system of rigid bodies with no closed loops can be expressed as a kinematic chain formed by a tree hierarchy of links and joints. Each link in the hierarchy is represented by a rigid body, while joints define connections between links constraining the rigid body motion in the system. Mathematical formulation for a kinematic chain is then the function of its independent system parameters: the generalized coordinates $\boldsymbol{q}$. The forward kinematic solution to the system is then the end effector state $\boldsymbol{e}$ expressed as a function of the system parameters

$$f(\boldsymbol{q}) = \boldsymbol{e} \tag{54}$$

Finding the system parameters as a function of the end effector state then involve solving the inverse kinematic problem for the system

$$f^{-1}(\boldsymbol{e}) = \boldsymbol{q} \tag{55}$$

Equations above act as abstract definitions for expressing the state of a rigid body system. Intention of following sections is then to define equations involved and derive the algorithms for rigid body systems of open-ended serial link chains constrained by revolute and prismatic joints.

3.1 Rigid body displacement

Rigid body motion involves combinations of translation of the rigid body frame or rotations of the frame basis and amounts to a displacement of the rigid body frame. Transformation for any point $\boldsymbol{x}_b \in \Re^3$ displacing a point from the rigid body frame is then

$$\boldsymbol{x}_s = R\boldsymbol{x}_b + \boldsymbol{p} \tag{56}$$

with the displacement transform represented in the 4×4 homogenous form

$$\begin{vmatrix} \boldsymbol{x}_s \\ 1 \end{vmatrix} = \begin{vmatrix} R & \boldsymbol{p} \\ 0 & 1 \end{vmatrix} \begin{vmatrix} \boldsymbol{x}_b \\ 1 \end{vmatrix} = T \begin{vmatrix} \boldsymbol{x}_b \\ 1 \end{vmatrix} \tag{57}$$

Considering the system of a single rigid body link, notation for a transformation from the body frame to the fixed frame (task space) using subscript b and s for the body and fixed frame respectively is

$$\begin{vmatrix} \boldsymbol{x}_s \\ 1 \end{vmatrix} = T_{sb} \begin{vmatrix} \boldsymbol{x}_b \\ 1 \end{vmatrix} \tag{58}$$

For a rigid body hierarchy, parameterization of the i:th link in the reference frame of the previous link $i-1$ using the notation above yield

$$\begin{vmatrix} \boldsymbol{x}_{b_{i-1}} \\ 1 \end{vmatrix} = T_{b_{i-1}b_i} \begin{vmatrix} \boldsymbol{x}_{b_i} \\ 1 \end{vmatrix} \tag{59}$$

For a link series forming an open-ended chain, the combined fixed space transform for the i:th joint is then the product of the transforms for all previous joints in the chain

$$T_{sb_i} = \prod_{j=0}^{i} T_{b_{j-1}b_j} = T_{sb_1} T_{b_1 b_2} \cdots T_{b_{i-1}b_i} \tag{60}$$

With *Eq.* 60 transforming points defined in the i:th rigid body frame to the fixed space frame. The problem of computing the transform for a link in a rigid body hierarchy can then be reduced

to the simplified problem of an open-ended partial serial chain. Allowing *Eq.* 60 to be used for computing the rigid body transform for a link in the partial system chain in the hierarchy. With rigid body links only dependent on the partial chain, system equations can be reduced to a function of the partial chain, and only serial link chains will be considered for remaining equations.

3.2 Forward kinematics

To describe a serial link chain as a function of the independent parameters it is important to distinguish the concept of the modeled chain as a set of both rigid bodies and joints, from the concept of a serial link chain with joints defining the links in which rigid bodies can be realized. Implication from using the later concept is that the chain can be modeled as a serial chain of 1 DoF joints with intermediate reference frames. Joints with multiple DoF can then be formulated as a series of simpler joints rather than a single entity, and ball and socket joints as an example can be constructed from 3 revolute joints rotating around the unit axes. Forward kinematics for the i:th body frame in a serial link chain given state $\boldsymbol{q}$ is then

$$T_{sb_i}(\boldsymbol{q}) = \prod_{j=0}^{i} T_{b_{j-1}b_j}(\boldsymbol{q}_j) = \prod_{j=0}^{i} T_{b_{j-1}c_j} \begin{vmatrix} e^{\boldsymbol{\omega}_j \boldsymbol{q}_j} & \boldsymbol{d}_j \boldsymbol{q}_j \\ 0 & 1 \end{vmatrix} \tag{61}$$

with $T_{b_{j-1}c_j}$ representing a constant offset from the parent frame in which the joint is parameterized. Vectors $\boldsymbol{\omega}_j$ and $\boldsymbol{d}_j$ is the velocity of the joint displacement, or twist, with respect to changes in the independent parameter $\boldsymbol{q}_j$. Twist in the joint frame then generate a rotation around the axis $\hat{\boldsymbol{k}}_j = \boldsymbol{\omega}_j / \|\boldsymbol{\omega}_j\|$ with angular velocity $\|\boldsymbol{\omega}_j\|$, and a translation in direction $\hat{\boldsymbol{d}}_j = \boldsymbol{d}_j / \|\boldsymbol{d}_j\|$ with velocity $\|\boldsymbol{d}_j\|$. Computing the position in the fixed frame given some state $\boldsymbol{q}$ and position $\boldsymbol{v}$ in the i:th rigid body frame is now simply

$$\begin{vmatrix} \boldsymbol{v}_s \\ 1 \end{vmatrix} = T_{sb_i}(\boldsymbol{q}) \begin{vmatrix} \boldsymbol{v}_b \\ 1 \end{vmatrix} \tag{62}$$

Formulation for rigid body motion can be expanded further as discussed in[5], it can for example be shown that the twist can be reformulated as a screw motion around a screw axis. Homogenous transformations can also be expressed on the form of matrix exponentials with the 6-dimensional twist as the exponential coordinates in similarity to rotations (see [*section*.2.6]). However, expanding definitions further is not crucial for defining revolute and prismatic joints as we will see later.

3.3 End-effectors

End-effectors in the rigid body system can be defined as spatial rigid body states with a known displacement in task space. Parameterizations for a rigid body displacement follows from *Eq.* 57 and can either be a partial state as the position $\boldsymbol{p}$ in the fixed frame, or a displacement defined by both position and orientation $\boldsymbol{\omega}$. While a displacement involving both translation and rotation can be parameterized as a screw motion using the exponential coordinate representation in $\mathbb{R}^6$, the independent coordinate representation will be used here

$$\boldsymbol{e}_s = \begin{vmatrix} \boldsymbol{p}_{\boldsymbol{e}_s} \\ \boldsymbol{\omega}_{\boldsymbol{e}_s} \end{vmatrix} \tag{63}$$

Parameterizations in $\mathbb{R}^6$ for a rigid displacement can then be associated with an end-effector of a robotic arm performing some task, and partial states can be positional data measured to reconstruct the link chain parameterizations (with the example of motion capture data). Modeling an end effector as the displacement T_e relative to some rigid body (or joint frame) denoted b within the system, the forward kinematics *Eq.* 61 for the displacement T_{se} in fixed space is

$$T_{se} = T_{sb}(\boldsymbol{q})T_e \tag{64}$$

and the matrix has the form

$$T_{se} = \begin{vmatrix} R_{sb} & \boldsymbol{p}_{sb} \\ 0 & 1 \end{vmatrix} \begin{vmatrix} e^{[\boldsymbol{\omega}_e]_\times} & \boldsymbol{p}_e \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} R_{sb}e^{[\boldsymbol{\omega}_e]_\times} & R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ 0 & 1 \end{vmatrix} \tag{65}$$

Highlighting independence of $\boldsymbol{\omega}_{\boldsymbol{e}_s}$ from the translation $\boldsymbol{p}_{\boldsymbol{e}_s}$, the end effector as a function of the system configuration can be expressed using the logarithm from [*section.*2.7] as

$$\boldsymbol{e}_s(\boldsymbol{q}) = \begin{vmatrix} R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ log(R_{se}) \end{vmatrix} \tag{66}$$

Taking in consideration the derivative of the exponential coordinate $\boldsymbol{\omega}_s$ with respect to some system coordinate q_j causing rotation in the end effector, by leaving the remainder of the system constant one gets the partial derivative

$$\frac{\delta \boldsymbol{\omega}_{\boldsymbol{e}_s}}{\delta q_j} = \frac{\boldsymbol{\omega}_{\boldsymbol{e}_s}(q_j + \delta) - \boldsymbol{\omega}_{\boldsymbol{e}_s(q_j)}}{\delta} = \frac{log(e^{[R_{sb}\hat{\boldsymbol{k}}_j]_\times \delta}R_{se}) - log(R_{se})}{\delta} \tag{67}$$

And while there are solutions to *Eq.* 67, reformulating the end-effector as a function of the instantaneous configuration will simplify the derivative as can be seen in the next section.

3.3.1 End effector with respect to the instantaneous configuration

Additivity for rotations is only possible when rotations commute, but commutation only occur for rotations around the same axis. However, angular velocities are additive at the identity element due to commutative property of infinitesimal rotations (as shown in [*section.*2.8.4]). Reformulating the system as a function of the 'instantaneous configuration in the fixed frame' using a rotation constant R_{ei} one gets

$$T_{si}(\boldsymbol{q}) = T_{se}T_{ei} = \begin{vmatrix} R_{se} & R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ 0 & 1 \end{vmatrix} \begin{vmatrix} R_{ei} & 0 \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} R_{se}R_{ei} & R_{sb}\boldsymbol{p}_e + p_{sb} \\ 0 & 1 \end{vmatrix} \tag{68}$$

Expression for the end effector as the vector $\boldsymbol{i}$ for a configuration $\boldsymbol{q}$ becomes

$$\boldsymbol{i}_s(\boldsymbol{q}) = \begin{vmatrix} \boldsymbol{p}_{\boldsymbol{i}_s} \\ \boldsymbol{\omega}_{\boldsymbol{i}_s} \end{vmatrix} = \begin{vmatrix} R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ log(R_{se}R_{si}) \end{vmatrix} \tag{69}$$

To simplify the partial derivative for the end effector $\boldsymbol{i}$ one can now set the (independent) constant R_{ei} using the equivalence $R_{ei} = R_{se}(\boldsymbol{q}_0)$. And for some instantaneous configuration $\boldsymbol{q}_0 = \boldsymbol{q}$ the coordinate representation is equivalent to

$$\boldsymbol{i}_s(\boldsymbol{q}_0) = \begin{vmatrix} R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ log(R_{se}R_{se}^T) \end{vmatrix} = \begin{vmatrix} R_{sb}\boldsymbol{p}_e + \boldsymbol{p}_{sb} \\ 0 \end{vmatrix} \tag{70}$$

Exponential coordinate $\boldsymbol{\omega_i}$ for the end effector $\boldsymbol{i}$ is now analogous to an angular velocity. Partial derivative for some rotation dependent on q_j relative to the instantaneous configuration $\boldsymbol{q}_0$ is now

$$\frac{\delta \boldsymbol{\omega_{i_s}}}{\delta q_j} = \frac{log(e^{[R_{sb_j}\hat{\boldsymbol{k}}_j]_\times \delta} R_{se} R_{se}^T) - log(R_{se}R_{se}^T)}{\delta} = \frac{R_{sb_j}\hat{\boldsymbol{k}}_j\delta - 0}{\delta} = R_{sb_j}\hat{\boldsymbol{k}}_j \tag{71}$$

As can be seen above, reformulating the system equation on the form $T_{si}(\boldsymbol{q})$ simplifies the partial derivative for the end effector, and will be useful for finding a solution to the inverse kinematics problem. Determining the displacement T_{se} for some $\boldsymbol{i}$ can then be resolved through $T_{si}T_{ei}^T$. However, determining the transform associated with $\boldsymbol{i}$ will not be necessary as we will see later.

3.4 Prismatic joint

To compute a general solution to the inverse kinematics problem for reaching a given end-effector state, it will be necessary to compute the change in the end-effector state with respect to changes in the system configuration $\boldsymbol{q}$. And by using the system as a construct of 1-DoF joints defined in [*section*.3.2], prismatic joints generate translative motion in the joint frame along some axis $\hat{\boldsymbol{d}}$. Fixed frame expression for an end effector as a function of the system parameter q_j associated with the prismatic joint j, while leaving the rest of the system constant, yield

$$T_{se}(q_j) = T_{sc_j}\begin{vmatrix} e^0 & q_j\hat{\boldsymbol{d}}_j \\ 0 & 1 \end{vmatrix} T_{c_{j+1}e} = \begin{vmatrix} I & q_j R_{sc_j}\hat{\boldsymbol{d}}_i \\ 0 & 1 \end{vmatrix} T_{sc_j}T_{c_{j+1}e} \tag{72}$$

And is equivalent since moving in the direction $\hat{\boldsymbol{d}}_j$ in the joint frame is equivalent to moving in the direction $R_{sc_j}\hat{\boldsymbol{d}}_j$ in the fixed frame. Formulating prismatic displacement in *Eq.* 72 as a function for displacing the independent coordinates in *Eq.* 69 one gets

$$\boldsymbol{i}_s(q_j) = \begin{vmatrix} R_{sc_j}(q_j\hat{\boldsymbol{d}}_j + \boldsymbol{p}_{c_{j+1}e}) + \boldsymbol{p}_{sc_j} \\ log(IR_{se}R_{se}^T) \end{vmatrix} = \begin{vmatrix} q_j R_{sc_j}\hat{\boldsymbol{d}}_j + \boldsymbol{p}_{s_c} \\ 0 \end{vmatrix} \tag{73}$$

where the constant $\boldsymbol{p}_{s_c}$ is used to factor the position of the end effector defined by the constant system when $q_j = 0$. Differentiating with respect to the joint parameter q_j gives the body twist for the end-effector in the fixed frame basis

$$\frac{\delta \boldsymbol{i}_s}{\delta q_j} = \begin{vmatrix} \delta \boldsymbol{p}_{i_s}/\delta q_j \\ \delta \boldsymbol{\omega}_{i_s}/\delta q_j \end{vmatrix} = \begin{vmatrix} R_{sc_j}\hat{\boldsymbol{d}}_j \\ 0 \end{vmatrix} \tag{74}$$

which represents the change in the end effector $\boldsymbol{i}$ in relation to some instantaneous configuration $\boldsymbol{q}$.

3.5 Revolute joint

Revolute/hinge joints constitute another atomic construct allowing rotation of the joint frame around some rotation axis $\hat{\boldsymbol{k}}$. Displacement in fixed space for a revolute joint with regard to its independent parameter q_j is

$$T_{se}(q_j) = T_{sc_j}\begin{vmatrix} e^{[\hat{\boldsymbol{k}}]_\times q_j} & 0 \\ 0 & 1 \end{vmatrix} T_{c_{j+1}e} = \begin{vmatrix} e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j} & (I - e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j})\boldsymbol{p}_{sc_j} \\ 0 & 1 \end{vmatrix} T_{sc_j}T_{c_{j+1}e} \tag{75}$$

where $\boldsymbol{p}_{sc_j}$ is the fixed-space position of the revolute joint for the current configuration. The end-effector displacement as a function of the angle can then be summarized by the product

$$T_{se}(q_j) = \begin{vmatrix} e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j} R_{sc_j} R_{sc_{j+1}} & e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j}(\boldsymbol{p}_{s_c} - \boldsymbol{p}_{sc_j}) + \boldsymbol{p}_{sc_j} \\ 0 & 1 \end{vmatrix} \tag{76}$$

where $\boldsymbol{p}_{s_c}$ used to represent the current position of the end-effector for the constant configuration $q_j = 0$. The difference $\boldsymbol{p}_{s_c} - \boldsymbol{p}_{sc_j}$ is then the vector from the joint origin to the end effector when the joint is at its identity configuration. Thus, the equation can be summarized as a fixed space expression for the product of the constant joint transform with a rotation of the remaining system.

Formulating *Eq.* 76 as a displacement of the independent coordinates in *Eq.* 69 with regard to some configuration q_{0_j}, one gets

$$\boldsymbol{i}_s(q_j) = \begin{vmatrix} e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j}(\boldsymbol{p}_{s_c} - \boldsymbol{p}_{sc_j}) + \boldsymbol{p}_{sc_j} \\ log(e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j} R_{sc_j} R_{sc_{j+1}} R_{se}^T) \end{vmatrix} = \begin{vmatrix} e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j - q_{0_j}}(\boldsymbol{p}_{s_0} - \boldsymbol{p}_{sc_j}) + \boldsymbol{p}_{sc_j} \\ log(e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j - q_{0_j}}) \end{vmatrix} \tag{77}$$

Here, the rightmost side use the commutative and linear property for rotations around the same axis, replacing $\boldsymbol{p}_{sc_j}$ with the end effector position $\boldsymbol{p}_{s_0}$ determined by the configuration $\boldsymbol{q}_0$. Derivative for $\boldsymbol{\omega}_{\boldsymbol{i}_s}$ is now identical to *Eq.* 71 as determined in [*section*.3.3.1], and derivative for $\boldsymbol{p}_{\boldsymbol{i}_s}$ can be computed using the current end effector position $\boldsymbol{p}_s$ as

$$\frac{\delta \boldsymbol{p}_{\boldsymbol{i}_s}}{\delta q_j} = [R_{sc_j}\hat{\boldsymbol{k}}]_\times e^{[R_{sc_j}\hat{\boldsymbol{k}}]_\times q_j}(\boldsymbol{p}_{s_c} - \boldsymbol{p}_{sc_j}) = [R_{sc_j}\hat{\boldsymbol{k}}]_\times (\boldsymbol{p}_s - \boldsymbol{p}_{sc_j}) \tag{78}$$

Derivative for the end effector relative to the instantaneous configuration can then be summarized by

$$\frac{\delta \boldsymbol{i}_s}{\delta q_j} = \begin{vmatrix} [R_{sc_j}\hat{\boldsymbol{k}}]_\times (\boldsymbol{p}_s - \boldsymbol{p}_{sc_j}) \\ R_{sb_j}\hat{\boldsymbol{k}}_j \end{vmatrix} \tag{79}$$

3.6 Inverse kinematics

Finding the inverse solution to the forward kinematic equation discussed in section [*section*.3.2] can in many cases be solved analytically. Iterative methods can on the other hand solve both over- and under-determined systems without imposing fictious constraints. Through stepwise optimization of a given initial condition, the error residual toward the given end effector state can be reduced until a minimum is reached (either globally or locally). The optimization problem for an open-link chain can then be formalized as

$$\min_{\boldsymbol{q}\in\mathbb{R}^n} \|\boldsymbol{e}_s(\boldsymbol{q}) - \boldsymbol{e}_T\| \tag{80}$$

where the function $f : \mathbb{R}^n \to \mathbb{R}^m$ is the forward kinematic function with respect to some intended end effector configuration $\boldsymbol{e}_T \in \mathbb{R}^m$. Solution to the optimization problem can then be determined through iterative methods such as the Gauss-Newton method.

3.7 Gauss-Newton

Gauss-Newtons method is a numerical approach for solving non-linear optimization problems and converge to a local minimum given some initial condition $\boldsymbol{q}_0$. To find a configuration closer to intended state $\boldsymbol{e}_T$, linearization of the residual function $E(\boldsymbol{q}) = \boldsymbol{e}_s(\boldsymbol{q}) - \boldsymbol{e}_T$ at current state $\boldsymbol{q}_n$ using the Jacobian $J(\boldsymbol{q}_n)$ yield

$$E(\boldsymbol{q}_{n+1}) \approx E(\boldsymbol{q}_n) + J(\boldsymbol{q}_n)\boldsymbol{\delta} \tag{81}$$

The linearized problem can then be solved by finding a root to the equation

$$E(\boldsymbol{q}_n) + J(\boldsymbol{q}_n)\boldsymbol{\delta} = 0 \tag{82}$$

where the solution to *Eq.* 82 minimizing the error in a least square sense can be found by solving the system of linear equations

$$J(\boldsymbol{q}_n)\boldsymbol{\delta} = -E(\boldsymbol{q}_n) \tag{83}$$

Solution $\boldsymbol{\delta}$ to *Eq.* 83 then represent a step from the current configuration

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n + \boldsymbol{\delta} \tag{84}$$

Configuration $\boldsymbol{q}_{n+1}$ then generally reduce the error residual relative to the intended end effector configuration $\boldsymbol{e}_T$. By repeatedly computing the Jacobian and solving *Eq.* 83, the optimization problem in *Eq.* 80 can be minimized locally in a least square sense [4].

3.7.1 Jacobian matrix

Before going further and defining the rigid body system residual, the Jacobian for the residual function E is the matrix containing the coordinate derivatives with respect to the system configuration $\boldsymbol{q}$

$$J(\boldsymbol{q}) = \begin{vmatrix} \frac{\delta E_0}{\delta q_0}(\boldsymbol{q}) & \frac{\delta E_0}{\delta q_1}(\boldsymbol{q}) & \cdots & \frac{\delta E_0}{\delta q_n}(\boldsymbol{q}) \\ \frac{\delta E_1}{\delta q_0}(\boldsymbol{q}) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\delta E_m}{\delta q_0}(\boldsymbol{q}) & \cdots & \cdots & \frac{\delta E_m}{\delta q_n}(\boldsymbol{q}) \end{vmatrix} \tag{85}$$

3.8 System residual

Residual toward an optimal end-effector configuration is the error between the intended and current configuration. Considering the optimization problem expressed in *Eq.* 80, the residual define the error for the current configuration relative the intended configuration as defined by $\boldsymbol{e}_T$. However, since the change is determined by solving *Eq.* 82 using the Jacobian. Rather than to directly use the forward-kinematic equation for computing the residual, coordinates are computed relative to the instantaneous configuration as discussed in [*section*.3.3.1]. Residual $\boldsymbol{r}$ is then the column vector

$$\boldsymbol{r} = E(\boldsymbol{q}) = \boldsymbol{i}_s(\boldsymbol{q}) - \boldsymbol{i}_s(\boldsymbol{q}_T) = \begin{vmatrix} \boldsymbol{p}_{i_s}(\boldsymbol{q}) - \boldsymbol{p}_{i_s}(\boldsymbol{q}_T) \\ log(R_{se}R_{se}^T - log(R_{sT}R_{se}^T)) \end{vmatrix} = \begin{vmatrix} \boldsymbol{p}_{se} - \boldsymbol{p}_{sT} \\ -log(R_{sT}R_{se}^T) \end{vmatrix} \tag{86}$$

for the current configuration $\boldsymbol{q} = \boldsymbol{q}_0$ and intended configuration defined by some unknown configuration $\boldsymbol{q}_T$. One can then see the benefit of using the relative representation. Particularly as $-\boldsymbol{r}$ can be determined using the original coordinate equation from the displacement difference T_{iT} defined by

$$T_{sT} = T_{si}T_{iT} \tag{87}$$

where the displacement difference T_{iT} is equivalent to

$$T_{iT} = T_{si}^{-1}T_{sT} = \begin{vmatrix} R_{se}R_{se}^T & -I\boldsymbol{p}_{se} \\ 0 & 1 \end{vmatrix} \begin{vmatrix} R_{sT}R_{se}^T & \boldsymbol{p}_{sT} \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} IR_{sT}R_{se}^T & I\boldsymbol{p}_{sT} - \boldsymbol{p}_{se} \\ 0 & 1 \end{vmatrix} \tag{88}$$

The right hand side is then equivalent to the negative residual when combined with *Eq.* 66. Solving the optimization problem in *Eq.* 83 is then possible by computing the residual $\boldsymbol{r}$, and the Jacobian J using the derivates defined in [*section*.3.4] and [*section*.3.5].

Finally, one can then see that the Jacobian for the representation is equivalent to the body Jacobian for the body twist [5] moved into the fixed frame. Thus, the approach is equivalent to solving the optimization problem as defined in the body frame, using coordinate vectors rotated into the fixed frame. Since the configuration change is resolved by updating the system parameters $\boldsymbol{q}$, solving the problem in another basis has no other implication.

3.8.1 System under unit of change

To better understand the optimization process, a good idea is to look at it as a system under change. Linearization reduces the complicated system to a simpler representation, which moved over a unit of time t, moves the current end effector configuration toward some intended configuration $\boldsymbol{e}_{iT}$. In particular, solving the linear equation from [*section*.3.7]

$$J(\boldsymbol{q}_n)\boldsymbol{\delta} = E(\boldsymbol{q}_n) \tag{89}$$

One finds the velocity which followed from a unit of time corresponds to the displacement

$$T(t) = \begin{vmatrix} e^{[\omega_{\boldsymbol{e}_{iT}}]_\times t} & \boldsymbol{p}_{\boldsymbol{e}_{iT}t} \\ 0 & 1 \end{vmatrix} \tag{90}$$

where $T(1) = T_{iT}$. Thus, by solving the system of linear equations, one finds the change $\boldsymbol{\delta}$ in the system coordinates which correspond to the displacement T_{iT}. And the change represent a solution to *Eq.* 87 for the simpler system.

To define the simpler system, one can consider it to be a system with 2 degrees of freedom which translates and rotates from the current configuration in the direction of the intended configuration. And by solving the linear system, one finds the change in the system coordinates corresponding to the necessary angular velocity and translative velocity correcting the 2 DoF system. However, since the system is not equivalent to the simple system, the velocities only correspond to a motion in the correct direction with respect to the instantaneous configuration. And by moving with a constant velocity over for a unit of time, the system does not reach the intended configuration.

Since the direction is only valid for the current configuration, one would prefer to only take small steps toward the intended configuration. Therefore, using a Gauss-Newton approach might not be appropriate, since moving for a full unit of time might move the configuration further away from the solution! Approaches using the Levenberg-Marquardt method to dampen the step length can therefore be appropriate. However, as the indented purpose is not to cover different optimization methods, the topic is not covered.

4 Algorithms

To determine a set of algorithms for solving the inverse kinematics problem, solutions can be tailored to solve specific problems. And depending on the possible system and end effector configurations, three scenarios are covered:

- An open serial link chain with end effectors parameterized by position coordinates [*section.*4.1].
- A serial link chain on the form of a tree structure (forming multiple open chains), controlled by end effectors position coordinates [*section.*4.2].
- An open serial link chain consisting of spheroid joints controlled by a single end effector (each spheroid joint is constructed from 3 revolute joints). End effectors are parameterized by independent position and exponential coordinates [*section*.4.3].

Each approach is discussed in a separate section as referenced above. Algorithms are also constructed in consideration to a row-major memory alignment (similar to the C language) and designed for simple Numpy (Python) implementation. Before going into the IK algorithms themselves, an example of the Gauss-Newton optimization algorithm is presented in *algo.*1, and some of the notation used can be summarized by:

- A_i - Either represent access of the i:th element in a list/tensor or i:th column of a matrix. First element is indexed by 0 to match general code conventions.
- $A_{i,:}$ - Access of the i:th row of a matrix (unintuitive with regard to a row-major memory alignment but is used to match the math notation).
- $0_{n\times m}$ - Matrix of zeros with n columns and m rows.
- $\mathbb{1}_{n\times m}$ - Matrix of ones with n columns and m rows.
- $\{\{a, b\}, \dots\}$ - A list of ordered sets, with each subset containing elements a and b.
- $A \cup B$ - Insertion of B as the last element in the list/set A.
- $algo_i(\dots)$ - Function call referring to the i:th algorithm using some input arguments.
- # - Comment, used to highlight implementation details but are sparsely used.

Algorithm 1: A Gauss-Newton optimization algorithm.

Input : System function JE returning the Jacobian and residual, initial condition $\boldsymbol{q}_0$, maximum iteration count m, residual threshold ϵ, and system arguments $args$.
Output: Optimized $\boldsymbol{q}$ and residual $\boldsymbol{r}$.
begin
  while $i < m$ **do**
    $\{ J, \boldsymbol{r} \} \leftarrow JE(\boldsymbol{q}, args)$
    if $\|\boldsymbol{r}\| < \epsilon$ **then**
      break
    end
    $\boldsymbol{\delta} \leftarrow J^{-1} y$ `// Use an appropriate solver when` J^{-1} `is not square.`
    $\boldsymbol{q} \leftarrow \boldsymbol{q} - \boldsymbol{\delta}$ `// Subtract` $\boldsymbol{\delta}$ `since` y `should be negated.`
  end
end

4.1 Open revolute link chain

Algorithm for a simple system consisting of a set of revolute joints. End effectors are defined by position coordinates sampled in task space and modeled as translative offsets in joint space. Parameters used within this section's algorithms are:

- $\boldsymbol{q}$: Configuration state for the system as defined by the generalized coordinates. For this section, the vector only contains angles associated with revolute joints.
- B: Set of rigid body parameters on form $\{\{\hat{\boldsymbol{k}}, \boldsymbol{p}_c\}, \ldots\}$, containing rotation axis $\hat{\boldsymbol{k}}$ and translation offset $\boldsymbol{p}_c$.
- O: Set of end effector parameters on form $\{\{j, \boldsymbol{p}_e\}, \ldots\}$, with elements containing a translation offset $\boldsymbol{p}_e$ relative to the j:th joint.
- S: Intended end effector positions $\boldsymbol{p}_s$ sampled in task space where $S = \left|\boldsymbol{p}_{s_1}, \boldsymbol{p}_{s_2}, \ldots, \boldsymbol{p}_{s_n}\right|^T$.

Algorithm 2: Compute the set of joint parameters W in task space.

Input : Configuration $\boldsymbol{q}$ and chain parameters B.
Output: Ordered set of joint parameters W.
begin
  $R \leftarrow I$
  $\boldsymbol{c} \leftarrow 0_{3\times 1}$
  $W \leftarrow \{\ \}$
  for $\{\hat{\boldsymbol{k}}, \boldsymbol{p}\} \in B$ **and** $a \in \boldsymbol{q}$ **do**
    $c \leftarrow R\boldsymbol{c} + \boldsymbol{p}$
    $R \leftarrow Re^{[\hat{\boldsymbol{k}}]_\times a}$
    $W \leftarrow W \cup \{R\hat{\boldsymbol{k}}, R, \boldsymbol{c}\}$
  end
end

Algorithm 3: Compute position of modeled end effectors in task space.

Input : Ordered set of joint parameters W and end effector parameters O.
Output: Matrix $E \in \mathbb{R}^{n\times 3}$, containing n rows of end effector positions in task space.
begin
  $E \leftarrow 0_{n\times 3}$
  $i \leftarrow 0$
  for $\{ j, \boldsymbol{p}_e \} \in O$ **do**
    $\boldsymbol{\omega}, R, \boldsymbol{c} \leftarrow W_j$
    $E_{i,:} \leftarrow R\boldsymbol{p}_e + \boldsymbol{c}$
    $i \leftarrow i + 1$
  end
end

Algorithm 4: Compute index matrix I.

Input : Set of end effector parameters O.
Output: Matrix $I \in \mathbb{R}^{n\times 1}$, containing joint index associated with the n end effectors.
begin
  $i \leftarrow 0$
  for $\{ j, \boldsymbol{p}_e \} \in O$ **do**
    $I_i \leftarrow j$
    $i \leftarrow i + 1$
  end
end

Algorithm 5: Compute Jacobian.

Input : Ordered set of joint parameters W, end effector positions E, and index matrix I.
Output: Jacobian $J \in \mathbb{R}^{3n\times m}$.
begin
  $J \leftarrow 0_{3n\times m}$
  for $\{\boldsymbol{\omega}, R, \boldsymbol{c}\} \in W$ **do**
    $X \leftarrow (E - \mathbb{1}_{m\times 1}\boldsymbol{c}^T)[-\omega]_\times$
    $X \leftarrow X \circ (j <= I)\mathbb{1}_{1\times 3}$ # Elementwise less equal.
    $J_i \leftarrow \begin{vmatrix} X_{0,:} & X_{1,:} & \dots & X_{n-1,:} \end{vmatrix}^T$
    $j \leftarrow j + 1$
  end
end

Algorithm 6: Compute system Jacobian and residual.

Input : System configuration $\boldsymbol{q}$, rigid body parameters B, end effector parameters O, and intended end effector configuration S.
Output: Jacobian $J \in \mathbb{R}^{3n\times m}$, and residual vector $\boldsymbol{r} \in \mathbb{R}^{3n}$.
begin
  $W \leftarrow algo_2(\boldsymbol{q}, B)$
  $E \leftarrow algo_3(W, O)$
  $I \leftarrow algo_4(O)$
  $J \leftarrow algo_5(W, E, I)$
  $D \leftarrow E - S$
  $\boldsymbol{r} \leftarrow \begin{vmatrix} D_{0,:} & D_{1,:} & \dots & D_{n-1,:} \end{vmatrix}^T$
end

Optimization of a configuration state using the Gauss-Newton optimizer in *algo.*1 is then possible by feeding the optimizer *algo.*6 and the necessary system parameters. And as shown by the algorithms, a displacement constant T_{c_i} are not required since only a translation offset is used for each joint. However, using both rotation and translation offsets relative to a parent joint avoids arbitrary ranges for the angle limits for the joints, among other things.

4.2 Open link chain hierarchy

Algorithms for a system consisting of a hierarchy of revolute and prismatic joints with end effector defined by position coordinates. Note that the algorithms are constructed as an extended version from previous section. Parameters used within this section's algorithms are:

- $\boldsymbol{q}$: Configuration state for the system as defined by the generalized coordinates. In this section parameters represent either a translation along direction $\hat{\boldsymbol{d}}$ or rotation around axis $\hat{\boldsymbol{k}}$, or both (for screw joints).
- B: Set of rigid body parameters on form $\{\{\hat{\boldsymbol{k}},\ \hat{\boldsymbol{d}},\ R_c,\ \boldsymbol{p}_c\},\ \ldots\}$, containing parent link i, rotation axis $\hat{\boldsymbol{k}}$, translation direction $\hat{\boldsymbol{d}}$ and transformation offsets R_c and $\boldsymbol{p}_c$.
- O: Set of end effector parameters on form $\{\{j, \boldsymbol{p}_e\},\ \ldots\}$, with elements containing a translation offset $\boldsymbol{p}_e$ relative to the j:th joint.
- S: Intended end effector positions $\boldsymbol{p}_s$ sampled in task space where $S = \left|\boldsymbol{p}_{s_1},\ \boldsymbol{p}_{s_2},\ \ldots,\ \boldsymbol{p}_{s_n}\right|^T$.

Algorithm 7: Compute the set of joint parameters W in task space.

Input : Configuration $\boldsymbol{q}$ and rigid body parameters B.
Output: Ordered set of joint parameters W.
begin
  $R \leftarrow 0_{(m+1)\times 3\times 3}$
  $C \leftarrow 0_{(m+1)\times 3\times 1}$
  $W \leftarrow \{\ \}$
  $R_0 \leftarrow I$
  $j \leftarrow 1$
  for $\{i, \hat{\boldsymbol{k}}, \hat{\boldsymbol{d}}, R_c, \boldsymbol{p}_c\} \in B$ **and** $q_j \in \boldsymbol{q}$ **do**
    $R_c \leftarrow R_{i+1}R_c$
    $C_j \leftarrow R_{i+1}\boldsymbol{p}_c + C_{i+1} + q_j R_c \hat{\boldsymbol{d}}$
    $R_j \leftarrow R_c e^{[\hat{\boldsymbol{k}}]_\times q_j}$
    $W \leftarrow W \cup \{R\hat{\boldsymbol{k}}, R_c\hat{\boldsymbol{d}}, R_j, C_j\}$
    $j \leftarrow j + 1$
  end
end

Algorithm 8: Compute position of modeled end effectors in task space.

Input : Ordered set of joint parameters W and end effector parameters O.
Output: Matrix $E \in \mathbb{R}^{n \times 3}$, containing n rows of end effector positions in task space.
begin
  $E \leftarrow 0_{n \times 3}$
  $i \leftarrow 0$
  for $\{ j, \boldsymbol{p}_e \} \in O$ **do**
    $\boldsymbol{\omega}, \hat{\boldsymbol{d}}, R, \boldsymbol{c} \leftarrow W_j$
    $E_{i,:} \leftarrow R\boldsymbol{p}_e + \boldsymbol{c}$
    $i \leftarrow i + 1$
  end
end

Algorithm 9: Compute dependency matrix I.

Input : Rigid body parameters B and end effector parameters O.
Output: Dependency matrix $I \in \mathbb{R}^{m \times n}$, i:th row entry is marked by 1 if the i:th end effector is dependent joint associated with the row (0 represents independence).
begin
  $X \leftarrow 0_{m,m}$
  $I \leftarrow 0_{m,n}$
  $j \leftarrow m - 1$
  # Form dependency matrix X for the joint hierarchy.
  while $j >= 1$ **do**
    $\{ i, \hat{\boldsymbol{k}}, \hat{\boldsymbol{d}}, R_c, \boldsymbol{p}_c \} \leftarrow B_j$
    $X_{j,j} = 1$
    $X_{i,:} \leftarrow X_{i,:} \parallel X_{j,:}$ # Elementwise/bitwise or.
    $j \leftarrow j - 1$
  end
  $X_{0,0} = 1$

  # Form end effector dependency matrix.
  for $\{ j, \boldsymbol{p}_e \} \in O$ **do**
    $I_i \leftarrow X_j$ # Dependencies are defined in columns of X.
  end
end

Algorithm 10: Compute Jacobian.

Input : Ordered set of joint parameters W, end effector positions E, and dependency matrix I.
Output: Jacobian $J \in \mathbb{R}^{3n \times m}$.
begin
$J \leftarrow 0_{3n \times m}$
$j \leftarrow 0$
for $\{\boldsymbol{\omega}, \hat{\boldsymbol{d}}, R, \boldsymbol{c}\} \in W$ **do**
$X \leftarrow (E - \mathbb{1}_{m \times 1}\boldsymbol{c}^T)[-\omega]_\times$
$X \leftarrow I_{j,:}\mathbb{1}_{1\times 3} \circ X$
$J_i \leftarrow \begin{vmatrix} X_{0,:} & X_{1,:} & \dots & X_{n-1,:} \end{vmatrix}^T$
$j \leftarrow j + 1$
end
end

Algorithm 11: Compute system Jacobian and residual.

Input : System configuration $\boldsymbol{q}$, rigid body parameters B, end effector parameters O, and intended end effector configuration S.
Output: Jacobian $J \in \mathbb{R}^{3n \times m}$, and residual vector $\boldsymbol{r} \in \mathbb{R}^{3n}$.
begin
$W \leftarrow algo_7(\boldsymbol{q}, B)$
$E \leftarrow algo_8(W, O)$
$I \leftarrow algo_9(B, O)$
$J \leftarrow algo_{10}(W, E, I)$
$D \leftarrow E - S$
$\boldsymbol{r} \leftarrow \begin{vmatrix} D_{0,:} & D_{1,:} & \dots & D_{n-1,:} \end{vmatrix}^T$
end

4.3 Open spheroid link chain with target in $\mathbb{R}^6$

Algorithm for a system consisting of an open link chain of spheroid (ball and socket) joints with an end effector defined by both position and orientation coordinates. Algorithm for computing the Jacobian and residual is split in two parts which can be combined. Parameters used in this section are:

- $\boldsymbol{q}$: Configuration state for the system as defined by the generalized coordinates. System is parameterized using Euler angles.
- B: Set of rigid body parameters on form $\{\{R_c, \boldsymbol{p}_c\}, \dots\}$, with joint sets only determined through transformation offsets R_c and $\boldsymbol{p}_c$.
- O: The end effector is parameterized through the pair $\{\boldsymbol{p}_e, \boldsymbol{\omega}_e\}$, defining displacement from the last joint (using both a translation and orientation).
- S: Intended end effector configuration is parameterized in fixed space through the coordinate pair $\{\boldsymbol{p}_{sT}, \boldsymbol{\omega}_{sT}\}$.

Algorithm 12: Compute system configuration parameters.

Input : System configuration $\boldsymbol{q}$, rigid body parameters B.
Output: Joint positions $P \in \mathbb{R}^{3\times m}$, joint rotation axis $W \in \mathbb{R}^{3\times m}$, rotation of final joint R and its position $\boldsymbol{c}$.

begin
$P \leftarrow 0_{3\times m}$
$W \leftarrow 0_{3\times m}$
$R \leftarrow I$
$\boldsymbol{c} \leftarrow \boldsymbol{0}$

Compute configuration in task space.
$j \leftarrow 0$
for $\{R_c, \boldsymbol{p}_c\} \in B$ **do**
$\boldsymbol{c} = R_c\boldsymbol{p}_c + \boldsymbol{c}$
$R \leftarrow RR_c$
for $i \in 1, 2, 3$ **do**
$R \leftarrow Re^{[\boldsymbol{e}_i]_\times \boldsymbol{q}_j}$
$W_j \leftarrow R_i$
$P_j \leftarrow \boldsymbol{c}$
$j \leftarrow j + 1$
end
end
end

Algorithm 13: Compute system Jacobian and residual.

Input : System configuration $\boldsymbol{q}$, rigid body parameters B, end effector parameters O, and intended end effector configuration S.
Output: Jacobian $J \in \mathbb{R}^{3n\times m}$, and residual vector $\boldsymbol{r} \in \mathbb{R}^{3n}$.
begin
$\{P, W, R, \boldsymbol{c}\} \leftarrow algo_{12}$
$\{\boldsymbol{p}_e, \boldsymbol{\omega}_e\} \leftarrow O$
$\{\boldsymbol{p}_{sT}, \boldsymbol{\omega}_{sT}\} \leftarrow S$

Compute residual $\boldsymbol{i}_s(\boldsymbol{q}) - \boldsymbol{i}_s(\boldsymbol{q}_T)$.
$\boldsymbol{p}_{se} \leftarrow R\boldsymbol{p}_e + \boldsymbol{c}$
$\boldsymbol{\omega}_{\boldsymbol{r}} \leftarrow log(e^{[\omega_{sT}]_\times} e^{[-\omega_e]_\times} R^T)$
$\boldsymbol{r} \leftarrow \begin{vmatrix} \boldsymbol{p}_{se} - \boldsymbol{p}_{sT} \\ -\boldsymbol{\omega}_{\boldsymbol{r}} \end{vmatrix}$

Compute Jacobian.
$J_{\boldsymbol{p}} \leftarrow \begin{vmatrix} [W_0]_\times(\boldsymbol{p}_{se} - P_0) & [W_1]_\times(\boldsymbol{p}_{se} - P_1) & \dots & [W_{m-1}]_\times(\boldsymbol{p}_{se} - P_{m-1}) \end{vmatrix}$
$J \leftarrow \begin{vmatrix} J_{\boldsymbol{p}} \\ W \end{vmatrix}$
end

5 Conclusion

Deriving and solving the IK problem iteratively involves solving multiple mathematical problems. However, due to properties of rotations at the identity element, algorithms for solving the problem can be reduced to a few operations and straight forward to implement.

As a last note, there are a few interesting references not directly applicable to the topics discussed. But since they are relevant for further considerations, I summarized them below:

- Derivative for rotations with respect to exponential coordinates (rotation vector) [3].
- Expression for the logarithm of a rotation product as a sum of two parallel and a orthogonal rotation vectors [2].
- Collection of formulas for rigid body dynamics [1]. For example, it contains formulas for converting rotation derivatives between different representations.

References

[1] Robot dynamics lecture notes. `https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2016/RD2016script.pdf`, 2017.

[2] Kenth Engø. On the bch-formula in so(3). *BIT Numerical Mathematics*, 41(3):629–632, Jun 2001.

[3] Guillermo Gallego and Anthony Yezzi. A compact formula for the derivative of a 3-d rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision*, 51(3):378–384, Aug 2014.

[4] Ivar Gustafsson and Kjell Holmåker. *Numerisk Analys*. Liber AB, Stockholm, Sweden, 2016.

[5] Kevin Lynch M. and Park Frank C. *Modern Robotics Mechanics Planning and Control*. Cambridge University Press, Cambridge, United Kingdoom, 2017.

[6] Malcolm D. Shuster. Survey of attitude representations. *Journal of the Astronautical Sciences*, 41(4):439–517, October 1993.