# Exp[licit]

## A Robot Modelling Software based on Exponential Maps

Johannes Lachner*, Moses C. Nah*, Stefano Stramigioli *Fellow, IEEE*, Neville Hogan *Member, IEEE*

***Abstract*—The derivation of a robot's equation of motion is typically focused on placing multiple coordinate frames to express the kinematic and dynamic relationships between those coordinate frames. The Denavit-Hartenberg convention is commonly used to place the coordinate frames. This paper presents an alternative method using the Differential Geometric method of Exponential Maps, which reduces the number of coordinate frame choices required. The traditional and differential geometric method are compared, and the conceptual and practical differences are discussed in detail. The authors introduce their open-source software, Exp[licit]$^{TM}$, which is based on the differential geometric method and can be used by researchers and engineers with basic knowledge of geometry and robotics. Code snippets and an example application are provided to assist users in getting started with the software.**

***Index Terms*—Computational Geometry, Kinematics and Dynamics of Robot Manipulators, MATLAB Robotics Software.**

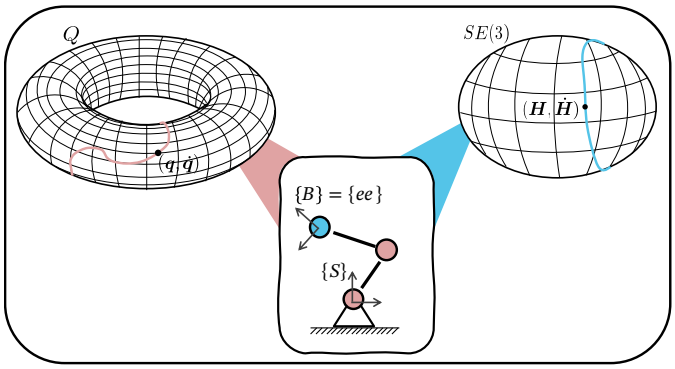


Fig. 1: Manifold $\mathcal{Q}$ and $SE(3)$, representing the robot joint and end-effector motion, respectively.

## I. Introduction

In standard robotic textbooks, orthonormal coordinate frames are used to describe robot kinematics and dynamics [1], [2]. To derive the robot parameters, predetermined rules have to be followed to position the coordinate frames and express the translational and rotational relations between them.

While this approach is highly popular, it has several limitations. First, a large number of coordinate frames has to be placed. This becomes especially unwieldy for robots with many degrees of freedom (DOF). Second, multiple conventions exist to define the coordinate frames. Within these conventions, different numbers of rules have to be applied. Some of the conventions need special treatment, e.g., for parallel axes where the description is not unique.

In contrast to the traditional approach, Differential Geometry can be used as a mathematical framework which lifts the coordinate-level descriptions to the more abstract space of manifolds. In robotics, two manifolds are of particular interest: the set of all robot configurations $\boldsymbol{q}$ constitute the manifolds $\mathcal{Q}$ and the set of all homogeneous transformations $\boldsymbol{H}$ constitute the manifolds $SE(3)$ (fig. 1). Robot kinematics and dynamics can therefore be described as actions on those manifolds [3]. This mathematical abstraction leads to a formulation that requires the least number of coordinate frames to represent the robot's kinematics and dynamics. While the theoretical strengths of geometric methods are widely accepted in robotics [4]–[7], we believe that conceptual and practical comparisons with traditional methods are not yet sufficiently emphasized.

The first contribution of this paper is to outline the conceptual and practical differences between traditional and differential geometric methods to derive robot kinematics and dynamics. We show that the geometric method is highly modular, flexible, and requires the least number of coordinate frames. In the second part of the paper, we introduce Exp[licit]$^{TM}$, a MATLAB robotic software which implements and leverages the advantages of the geometric method.

This document targets researchers and engineers who possess basic knowledge of geometry and robotics. While remaining technically sound, we will only present the essential techniques necessary to understand the conceptual and practical differences and be able to use the introduced software Exp[licit].

## II. Derivation of Robot Kinematics and Dynamics

This section summarizes how to obtain the kinematic and dynamic parameters of an open-chain $n$-DOF robot, using both methods. We will focus on deriving the Forward Kinematic Map, Jacobian matrix, and Mass Matrix of the robot.

### A. Preliminaries

Given the two manifolds $\mathcal{Q}$ and $SE(3)$, the robot's joint motion is a curve on $\mathcal{Q}$ and the robot's workspace (or end-effector) motion is a curve on $SE(3)$ [5], [6].

To represent the robot's workspace motion, either a stationary or body-fixed coordinate frame has to be chosen.[1] We assume one stationary frame $\{S\}$, attached to the fixed base

*J. Lachner and M. C. Nah contributed equally

J. Lachner, M. C. Nah, and N. Hogan are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA

N. Hogan is also with the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139 USA

S. Stramigioli is with the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7522 Enschede, The Netherlands

[1]From now on, we use "frame(s)" to refer to "coordinate frame(s)".

of the robot. Moreover, we denote $\{B\}$ as a body-fixed frame, which can be attached to any point of the robot. Often, $\{B\}$ coincides with the tool center point (i.e., the end-effector) of the robot. In this case, we denote $\{B\}$ as $\{ee\}$.

For a given joint configuration $\boldsymbol{q} \in \mathcal{Q}$, the orientation and translation of $\{ee\}$ with respect to $\{S\}$ can be derived via the *Forward Kinematic Map*, $\mathcal{Q} \rightarrow SE(3)$ and represented by the *Homogeneous Transformation Matrix* ${}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q}) = \begin{pmatrix} {}^{S}\boldsymbol{R}_{ee} & {}^{S}\boldsymbol{p}_{ee} \\ 0 & 1 \end{pmatrix} \in SE(3)$. Here, ${}^{S}\boldsymbol{R}_{ee} \in SO(3)$ is the *Rotation matrix* of $\{ee\}$ with respect to $\{S\}$ and ${}^{S}\boldsymbol{p}_{ee} \in \mathbb{R}^3$ is the translation from $\{S\}$ to $\{ee\}$.

For a given joint motion $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, the workspace motion of the robot's end-effector can be derived via the *Hybrid Jacobian Matrix*[2] ${}^{H}\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$, and represented by a 6D-vector of workspace velocities, called *Spatial Velocity* ${}^{S}\boldsymbol{V}_{ee} = \begin{pmatrix} {}^{S}\boldsymbol{v}_{ee} \\ {}^{S}\boldsymbol{\omega} \end{pmatrix} \in \mathbb{R}^6$. Here, ${}^{S}\boldsymbol{V}_{ee}$ incorporates the linear velocity ${}^{S}\boldsymbol{v}_{ee} \in \mathbb{R}^3$ of the origin of $\{ee\}$ with respect to $\{S\}$ and the angular velocity ${}^{S}\boldsymbol{\omega} \in \mathbb{R}^3$ of the end-effector body, both expressed in $\{S\}$.

The total kinetic co-energy $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}$ of an $n$-DOF robot is the sum of all contributions of kinetic co-energy stored by the individual bodies: $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$ [6]. The matrix $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is called the *Mass Matrix* of the robot.

### B. Traditional Method

*1) Forward Kinematic Map via DH-convention:* The Denavit-Hartenberg (DH) convention [8] is a widely used method to derive the Forward Kinematic Map. It is a set of rules to place body-fixed frames on the robot, and to derive the parameters that describe the kinematic relation between adjacent frames [6]. Within the multiple DH-conventions [2], [9], [10], we outline the modified DH-convention which consists of four DH-parameters: link length $a$, link twist $\alpha$, link offset $d$, and joint angle $\theta$ [1], [6], [11].

To derive the DH-parameters, multiple frames have to be placed on each link by using the following rules (fig. 2):

(i) Define frames $\{1\}$, $\{2\}$, $\cdots$, $\{n\}$ on each link, ordered from the base to the end-effector of the robot. Choose axis $\hat{Z}_i$ of frame $\{i\}$ to be aligned with the $i$-th joint. For a revolute (prismatic) joint, direction of $\hat{Z}_i$ is along the positive direction of rotation (translation).

(ii) For $i = 1, 2, ..., n-1$, find a line segment that is mutually perpendicular to axes $\hat{Z}_i$ and $\hat{Z}_{i+1}$. The intersection between this line and $\hat{Z}_i$ is the origin of frame $\{i\}$. Moreover, axis $\hat{X}_i$ is chosen to be aligned with this line segment, pointing from $\hat{Z}_i$ to $\hat{Z}_{i+1}$.

(iii) Attach the origin of frame $\{ee\}$ to the end-effector. To simplify the derivation of the DH-parameters, the $\hat{Z}_{ee}$ axis is usually chosen to be parallel to $\hat{Z}_n$ [1]. From $\hat{Z}_n$ and $\hat{Z}_{ee}$, $\hat{X}_n$ is defined using step (ii). Finally, choose $\hat{X}_{ee}$ such that valid DH-parameters can be defined [6].

(iv) The $\hat{Y}$ axes of frames $\{1\}$, $\{2\}$, $\cdots$, $\{n\}$, $\{ee\}$ are defined using the right-hand convention.

[2]We will elaborate the notion "Hybrid" in the next subsection. Moreover, superscript $H$ denotes "Hybrid," rather than referring to a frame.

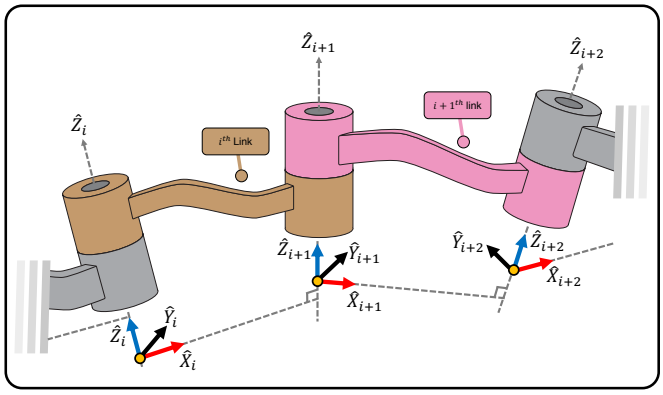


Fig. 2: Frames attached to an open-chain robot, using the DH-conventions.

(v) Attach frame $\{S\}$ to the robot base. Usually, it is chosen to coincide with frame $\{1\}$ when joint 1 has zero displacement.

After assigning $n+2$ frames, $\{S\}$, $\{1\}$, $\cdots$, $\{n\}$, $\{ee\}$, the $4(n+1)$ DH-parameters can be expressed. With these parameters, the Homogeneous Transformation Matrix ${}^{i-1}\boldsymbol{H}_i \in SE(3)$ between frame $\{i-1\}$ and $\{i\}$ is defined for $i = 1, 2, ..., n+1$, where $\{0\} \equiv \{S\}$ and $\{n+1\} \equiv \{ee\}$. Finally, by concatenating these matrices, the Forward Kinematic Map, ${}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q})$ can be derived:

$$ {}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q}) = {}^{S}\boldsymbol{H}_1(q_1)\ {}^{1}\boldsymbol{H}_2(q_2)...{}^{n-1}\boldsymbol{H}_n(q_n)\ {}^{n}\boldsymbol{H}_{ee} \quad (1) $$

*2) Jacobian Matrix by separating linear and angular velocities:* To derive the Jacobian Matrix, the traditional method separately relates joint velocities to linear and angular workspace velocities [2]. We denote the linear and rotational part of the Jacobian as $\boldsymbol{J}(\boldsymbol{q})_v \in \mathbb{R}^{3\times n}$ and $\boldsymbol{J}(\boldsymbol{q})_\omega \in \mathbb{R}^{3\times n}$, respectively.

To derive $\boldsymbol{J}(\boldsymbol{q})_v$, the position ${}^{S}\boldsymbol{p}_{ee}$ has to be extracted from ${}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q})$ (Section II-B1). Since ${}^{S}\boldsymbol{p}_{ee}$ is an analytical function of $\boldsymbol{q}$, $\boldsymbol{J}(\boldsymbol{q})_v$ collects the partial derivatives of ${}^{S}\boldsymbol{p}_{ee}$, with respect to the coordinate components of $\boldsymbol{q}$. Often, $\boldsymbol{J}(\boldsymbol{q})_v$ is called an "Analytical Jacobian" [2].

The matrix $\boldsymbol{J}(\boldsymbol{q})_\omega$ is commonly derived by using a geometric method and by specifying the frames based on DH-convention [2] (Section II-B1). More specifically, for $i = 1, 2, ..., n$:

- If the $i$-th joint is a revolute joint with unit-rotation axis ${}^{i}\hat{\boldsymbol{\omega}}_i$ expressed in $\{i\}$, the $i$-th column of $\boldsymbol{J}(\boldsymbol{q})_\omega$ is ${}^{S}\boldsymbol{R}_i\,{}^{i}\hat{\boldsymbol{\omega}}_i = {}^{S}\hat{\boldsymbol{\omega}}_i$.
- If the $i$-th joint is a prismatic joint, the $i$-th column of $\boldsymbol{J}(\boldsymbol{q})_\omega$ is a zero vector.

To calculate the spatial velocity ${}^{S}\boldsymbol{V}_{ee}$, $\boldsymbol{J}(\boldsymbol{q})_v$ and $\boldsymbol{J}(\boldsymbol{q})_\omega$ can be vertically concatenated:

$$ {}^{S}\boldsymbol{V}_{ee} = {}^{H}\boldsymbol{J}(\boldsymbol{q})\ \dot{\boldsymbol{q}} \quad (2) $$

Due to the analytical relation of $\boldsymbol{J}(\boldsymbol{q})_v$ and the geometrical derivation of $\boldsymbol{J}(\boldsymbol{q})_\omega$, we call ${}^{H}\boldsymbol{J}(\boldsymbol{q})$ the *Hybrid Jacobian Matrix*.

*3) Mass Matrix via Hybrid Jacobians:* To derive the Mass Matrix of the robot, it is necessary to attach $n$ additional frames to the center of mass (COM) of the $n$ bodies. These will be denoted as $\{C_1\}, \{C_2\}, \cdots, \{C_n\}$, ordered from the base to the end-effector of the robot. The moment of inertia of the $i$-th body with respect to $\{C_i\}$ is denoted ${}^{i}\boldsymbol{\mathcal{I}}_i \in \mathbb{R}^{3\times3}$. To express ${}^{i}\boldsymbol{\mathcal{I}}_i$ in $\{S\}$, the rotation matrix ${}^{S}\boldsymbol{R}_i$ is used (Section II-B1): ${}^{S}\boldsymbol{\mathcal{I}}_i = {}^{S}\boldsymbol{R}_i \, {}^{i}\boldsymbol{\mathcal{I}}_i \, {}^{S}\boldsymbol{R}_i{}^{T}$.

For each body $i$, the Hybrid Jacobian Matrix ${}^{H}\boldsymbol{J}_i(\boldsymbol{q})$ is derived, which describes the linear and angular velocity of $\{C_i\}$ with respect to $\{S\}$ (Section II-B2). Note that for each matrix ${}^{H}\boldsymbol{J}_i(\boldsymbol{q})$, the columns from $i+1$ to $n$ are set to be zero since they do not contribute to the motion of body $i$ [2].

Finally, for a given mass $m_i \in \mathbb{R}$ of the $i$-th body, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ can be calculated by:

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q}) = m_i \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{q})_v{}^{T} \; \boldsymbol{J}_i(\boldsymbol{q})_v \\ + \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{q})_\omega{}^{T} \; {}^{S}\boldsymbol{\mathcal{I}}_i \; \boldsymbol{J}_i(\boldsymbol{q})_\omega\end{aligned} \tag{3}$$

### C. Differential geometric method

*1) Forward Kinematic Map via the Product of Exponentials Formula:* For the geometric method, only two frames $\{S\}$ and $\{ee\}$ have to be chosen and assigned to the initial joint configuration of the robot $\boldsymbol{q}_0 \in \mathcal{Q}$. The initial Homogeneous Transformation Matrix is denoted ${}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q}_0) \equiv {}^{S}\boldsymbol{H}_{ee,0} \in SE(3)$. In practice it is convenient to select $\{S\}$ and $\{ee\}$ to have equal orientation (i.e., rotation matrix equals the identity matrix) such that only the translation between $\{S\}$ and $\{ee\}$ has to be identified to calculate ${}^{S}\boldsymbol{H}_{ee,0}$.

In the next step, the *Unit Joint Twists*[3] ${}^{S}\hat{\boldsymbol{\eta}}_i \in \mathbb{R}^6$ of each joint at initial joint configuration are expressed with respect to $\{S\}$. Depending on the type of the $i$-th robot joint, ${}^{S}\hat{\boldsymbol{\eta}}_i \in \mathbb{R}^6$ is defined by:

- If the $i$-th joint is a revolute joint, the unit-axis of rotation is ${}^{S}\hat{\boldsymbol{\omega}}_i$. *Any point* ${}^{S}\boldsymbol{p}_{\eta_i} \in \mathbb{R}^3$ along ${}^{S}\hat{\boldsymbol{\omega}}_i$ can be selected to define ${}^{S}\hat{\boldsymbol{\eta}}_i = (-[{}^{S}\hat{\boldsymbol{\omega}}_i]{}^{S}\boldsymbol{p}_{\eta_i}, \; {}^{S}\hat{\boldsymbol{\omega}}_i)^T$. Here, $[{}^{S}\hat{\boldsymbol{\omega}}_i] \in so(3)$ is the skew-symmetric matrix form of ${}^{S}\hat{\boldsymbol{\omega}}_i$ [6]. The operation $[{}^{S}\hat{\boldsymbol{\omega}}_i]{}^{S}\boldsymbol{p}_{\eta_i}$ is equal to ${}^{S}\boldsymbol{\omega} \times {}^{S}\boldsymbol{p}_{\eta_i}$.
- If the $i$-th joint is a prismatic joint, the unit-axis of translation is ${}^{S}\hat{\boldsymbol{v}}_i$ and therefore ${}^{S}\hat{\boldsymbol{\eta}}_i = ({}^{S}\hat{\boldsymbol{v}}_i, \boldsymbol{0})$.

Note that the $n$ Joint Twists ${}^{S}\hat{\boldsymbol{\eta}}_i$ are defined with respect to a single frame $\{S\}$. For most robots, the unit-axes of rotation (or translation) can be identified by visual inspection. The positions ${}^{S}\boldsymbol{p}_{\eta_i}$ can be determined by using CAD-programs.

Finally, the *Product of Exponentials Formula* [12] can be used to derive the Forward Kinematic Map:

$$\begin{aligned}{}^{S}\boldsymbol{H}_{ee}(\boldsymbol{q}) = \exp\left([{}^{S}\hat{\boldsymbol{\eta}}_1]q_1\right) \; \exp\left([{}^{S}\hat{\boldsymbol{\eta}}_2]q_2\right) \\ \cdots \exp\left([{}^{S}\hat{\boldsymbol{\eta}}_n]q_n\right) \; {}^{S}\boldsymbol{H}_{ee,0}\end{aligned} \tag{4}$$

[3]For simplicity, we will omit the term "Unit" in what follows.

In this equation, $[{}^{S}\hat{\boldsymbol{\eta}}_i] \in se(3)$ is a $4\times4$ matrix representation of ${}^{S}\hat{\boldsymbol{\eta}}_i$ [5]. Given ${}^{S}\hat{\boldsymbol{\eta}} = ({}^{S}\boldsymbol{v}, {}^{S}\hat{\boldsymbol{\omega}})$ and $q \in \mathbb{R}$, a closed-form solution of $\exp([{}^{S}\hat{\boldsymbol{\eta}}]q)$ is available [6]:

$$\begin{aligned}&\exp([{}^{S}\hat{\boldsymbol{\omega}}]q) = \mathbb{I}_3 + \sin q[{}^{S}\hat{\boldsymbol{\omega}}] + (1-\cos q)[{}^{S}\hat{\boldsymbol{\omega}}]^2 \\ &\boldsymbol{G}(q) = \mathbb{I}_3 q + (1-\cos q)[{}^{S}\hat{\boldsymbol{\omega}}] + (q-\sin q)[{}^{S}\hat{\boldsymbol{\omega}}]^2 \\ &\exp\left([{}^{S}\hat{\boldsymbol{\eta}}]q\right) = \begin{bmatrix}\exp([{}^{S}\hat{\boldsymbol{\omega}}]q) & \boldsymbol{G}(q){}^{S}\boldsymbol{v} \\ \boldsymbol{0} & 1\end{bmatrix}\end{aligned} \tag{5}$$

*2) Jacobian Matrices via The Adjoint Map:* For the geometric method, two Jacobian matrices exist: the Spatial Jacobian ${}^{S}\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ and the Body Jacobian ${}^{B}\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ [5]. The Spatial (respectively Body) Jacobian relates joint velocities $\dot{\boldsymbol{q}}$ to the Spatial (respectively Body) Twist ${}^{S}\boldsymbol{\xi}$ (${}^{B}\boldsymbol{\xi}$) [5], [6]:

$${}^{S}\boldsymbol{\xi} = \begin{bmatrix}{}^{S}\boldsymbol{v}_s \\ {}^{S}\boldsymbol{\omega}\end{bmatrix} = {}^{S}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad {}^{B}\boldsymbol{\xi} = \begin{bmatrix}{}^{B}\boldsymbol{v}_b \\ {}^{B}\boldsymbol{\omega}\end{bmatrix} = {}^{B}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{6}$$

Here, ${}^{S}\boldsymbol{\omega}$ (${}^{B}\boldsymbol{\omega}$) is the angular velocity of the body, expressed in $\{S\}$ (respectively $\{B\}$); ${}^{S}\boldsymbol{v}_s$ is *not* the velocity of the origin of $\{S\}$, which is zero; it is the linear velocity of a point on the robot structure, viewed as if it travels through the origin of $\{S\}$ [5], [6]; ${}^{B}\boldsymbol{v}_b$ is the velocity of the origin of $\{B\}$ with respect to $\{S\}$, expressed in $\{B\}$ [5], [6].

The columns of ${}^{S}\boldsymbol{J}(\boldsymbol{q})$ and ${}^{B}\boldsymbol{J}(\boldsymbol{q})$ are derived by using the Joint Twists $\hat{\boldsymbol{\eta}}_i$ and the *Adjoint Map* $\boldsymbol{Ad}_{\boldsymbol{H}} : \mathbb{R}^6 \to \mathbb{R}^6$ associated with $\boldsymbol{H} \in SE(3)$ [5], [6], [13]. In matrix notation, $\boldsymbol{Ad}_{\boldsymbol{H}} = \begin{pmatrix}\boldsymbol{R} & [\boldsymbol{p}]\boldsymbol{R} \\ \boldsymbol{0} & \boldsymbol{R}\end{pmatrix}$.

For planar and simple robots, $\boldsymbol{\eta}_i'$ can be identified by visual inspection. In general, the $i$-th column $\boldsymbol{\eta}_i'$ of ${}^{S}\boldsymbol{J}(\boldsymbol{q})$ is:

$$\boldsymbol{\eta}_i' = \begin{cases}{}^{S}\hat{\boldsymbol{\eta}}_1 & i = 1 \\ \boldsymbol{Ad}_{{}^{S}\boldsymbol{H}_{i-1}} \, {}^{S}\hat{\boldsymbol{\eta}}_i & i = 2, ..., n\end{cases} \tag{7}$$

In this equation, ${}^{S}\boldsymbol{H}_{i-1}$ can be derived via the Product of Exponentials Formula, i.e., ${}^{S}\boldsymbol{H}_{i-1} = \exp\left([{}^{S}\hat{\boldsymbol{\eta}}_1]q_1\right)\exp\left([{}^{S}\hat{\boldsymbol{\eta}}_2]q_2\right)\cdots\exp\left([{}^{S}\hat{\boldsymbol{\eta}}_{i-1}]q_{i-1}\right)$.

With $\{B\}$ attached to the $j$-th body, the $i$-th column $\boldsymbol{\eta}_i^\dagger$ of ${}^{B}\boldsymbol{J}(\boldsymbol{q})$ for $i \le j$ is:

$$\boldsymbol{\eta}_i^\dagger = \left(\boldsymbol{Ad}_{{}^{i}\boldsymbol{H}_j \, {}^{S}\boldsymbol{H}_{B,0}}\right)^{-1} \, {}^{S}\hat{\boldsymbol{\eta}}_i \tag{8}$$

As for eq. (7), ${}^{i}\boldsymbol{H}_j$ can be derived via the Product of Exponentials Formula. Matrix ${}^{S}\boldsymbol{H}_{B,0} \in SE(3)$ is the Homogeneous Transformation of $\{B\}$ with respect to $\{S\}$ at initial joint configuration $\boldsymbol{q}_0$. For $j = 1, 2, ..., n-1$, the columns of ${}^{B}\boldsymbol{J}(\boldsymbol{q})$ from $j+1$ to $n$ are zero.

*3) Mass Matrix—Mapping Generalized Inertia with Body Jacobians:* For the geometric method, the translational and rotational body contributions do not have to be separated. Instead, using the $n$ frames $\{C_1\}$, $\{C_2\}$, ..., $\{C_n\}$ (Section II-B2), we define their corresponding Body Jacobian Matrices ${}^{B}\boldsymbol{J}_1(\boldsymbol{q})$, ${}^{B}\boldsymbol{J}_2(\boldsymbol{q})$, ..., ${}^{B}\boldsymbol{J}_n(\boldsymbol{q})$ (Section II-C2). Moreover, we use $m_i$ and ${}^{i}\boldsymbol{\mathcal{I}}_i$ to define the *Generalized Inertia matrix* $\boldsymbol{\mathcal{M}}_i = \begin{pmatrix}m_i\mathbb{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & {}^{i}\boldsymbol{\mathcal{I}}_{\boldsymbol{i}}\end{pmatrix} \in \mathbb{R}^{6\times6}$ for each body $i$. In practice, $\{C_i\}$ are aligned with the principal moments of inertia. Hence,

$\boldsymbol{\mathcal{M}}_i$ can be identified by using CAD-programs. Finally, the robot Mass Matrix can be calculated by:

$$\boldsymbol{M}(\boldsymbol{q}) = \sum_{i=1}^{n} {}^{B}\boldsymbol{J}_i(\boldsymbol{q})^T \; \boldsymbol{\mathcal{M}}_i \; {}^{B}\boldsymbol{J}_i(\boldsymbol{q}). \tag{9}$$

## III. Exp[licit]: Concept, Features and Use-Cases

This section is split into two parts. First, we highlight the conceptual and practical differences between the traditional and the geometric methods. To demonstrate the practical differences, we use a Franka robot.[4] Second, we introduce Exp[licit], a MATLAB-based robot software which leverages the advantages of the geometric method. By using Exp[licit], the model parameters of the Franka robot can be derived. The structure of Exp[licit] will be described by using code snippets and an example application. Finally, we compare the computational efficiency of Exp[licit] with the MATLAB-based open-source robotics software "Robotics, Vision and Control" (RVC) [11].

### A. Conceptual and practical comparison between traditional and geometric methods

*1) Forward Kinematic Map:* The DH-convention provides a minimal parameter representation (four parameters) to derive the Homogeneous Transformation Matrix [6]. This minimal representation comes at a cost: a set of rules has to be carefully stipulated, which requires an extensive preparation in placing and transforming $n+2$ frames. If adjacent axes intersect or are parallel to each other, additional rules have to be considered to handle these exceptions for step (ii) in Section II-B1 [1]. Since rotations and translations are only allowed along/about axes $\hat{X}$ and $\hat{Z}$, the choices for frames $\{S\}$ and $\{ee\}$ are restricted.

The geometric method requires two frames: the fixed inertial frame $\{S\}$ and the body-fixed frame $\{B\}$. Compared to the DH-approach, there are no restrictions on their position and orientation. The Product of Exponentials Formula enables a lot of flexibility. To calculate the Joint Twists at initial configuration, any point on the twist axis can be chosen. Once the Joint Twists are defined, the Forward Kinematic Map can be derived for any point on the robot structure. This conceptual advantage yields a reduced computation time for the Forward Kinematic Map (Section III-B3)

The practical benefit of the geometric method for the Franka robot can be seen in fig. 3. Compared to the DH-convention with nine frames [14], only two frames are needed. For our choice of initial configuration, the calculation of ${}^{S}\boldsymbol{H}_{ee,0}$ is straightforward since only the position of the end-effector has to be calculated. For our example, ${}^{S}\boldsymbol{p}_{ee,0} = (0.088, 0, 1.033)$ and ${}^{S}\boldsymbol{R}_{ee,0} = \mathbb{I}_3$. The Joint Twists of the Franka robot are shown in Table A1.

*2) Jacobian Matrices:* For the traditional method, the Hybrid Jacobian Matrix ${}^{H}\boldsymbol{J}(\boldsymbol{q})$ is separated into linear and angular parts. Before the linear part of ${}^{H}\boldsymbol{J}(\boldsymbol{q})$ can be derived, a choice for end-effector frame $\{ee\}$ has to be made. Changing the frame at a later stage will need a recalculation of the position, extracted from the Forward Kinematic Map.

[4]https://www.franka.de/

The geometric approach derived two different Jacobian matrices, ${}^{S}\boldsymbol{J}(\boldsymbol{q})$ and ${}^{B}\boldsymbol{J}(\boldsymbol{q})$. The basis of the derivation are the Joint Twists in initial configuration. Hence, no separation into linear and rotational parts is needed. ${}^{S}\boldsymbol{J}(\boldsymbol{q})$ and it's output ${}^{S}\boldsymbol{\xi}$ (eq. (6)) only depend on one frame $\{S\}$. By using the Adjoint Map, ${}^{S}\boldsymbol{\xi}$ can be mapped to any point on the robot structure. By choosing a point equal to the origin of $\{ee\}$, the Spatial Velocity can be derived:

$${}^{S}\boldsymbol{V}_{ee} = \underbrace{\begin{pmatrix} \mathbb{I}_3 & -[{}^{S}\boldsymbol{p}_{ee}] \\ \boldsymbol{0} & \mathbb{I}_3 \end{pmatrix} {}^{S}\boldsymbol{J}(\boldsymbol{q})}_{{}^{H}\boldsymbol{J}(\boldsymbol{q})} \dot{\boldsymbol{q}}. \tag{10}$$

Here, no modification of the Forward Kinematic Map is needed, which is beneficial for the length and clarity of the code and reduces the computation time of ${}^{H}\boldsymbol{J}(\boldsymbol{q})$ (Section III-B3).

*3) Mass Matrix:* For both approaches, the frames $\{C_i\}$ have to be attached to the COM of the robot in initial configuration. For the traditional method, the orientation of these coordinate frames is restricted to obtain a valid set of DH-parameters. Commonly, $\{C_i\}$ is chosen to be aligned with frame $\{i\}$ (fig. 3A, fig. 4A) and separately rotated by ${}^{S}\boldsymbol{\mathcal{I}} = {}^{S}\boldsymbol{R}_i {}^{i}\boldsymbol{\mathcal{I}} {}^{S}\boldsymbol{R}_i^T$.

For the geometric approach, the orientation of body frames $\{C_1\}$, $\{C_2\}$, ..., $\{C_n\}$ can be freely chosen. For each COM, the Body Jacobians are derived, again by using the Adjoint Map (eq. (7), (8)).

While the traditional method divides the derivation into linear and rotational contributions, the geometric method uses the generalized inertia matrices $\boldsymbol{\mathcal{M}}_i$ (eq. (9)) to derive the Mass Matrix. Even though $\boldsymbol{\mathcal{M}}_i$ may not be aligned with $\{S\}$, it need not be separately transformed. The transformation is incorporated in the map ${}^{B}\boldsymbol{J}_i(\boldsymbol{q})$.

### B. Exp[licit]—Robot Modelling based on Exponential Maps

The software can be installed from our Github repository: https://github.com/explicit-robotics/Explicit-MATLAB/. The documentation of the software can be found here: https://explicit-robotics.github.io/.

*1) Software structure:* The core of the software is the `RobotPrimitives`-class, which is used as the parent class of the software. It provides the member functions `getForwardKinematics`, `getSpatialJacobian`, `getHybridJacobian`, `getBodyJacobian`, `getMassMatrix`, `getGravityVector`, and `getCoriolisMatrix` for deriving the robot parameters. By inheriting the `RobotPrimitives`-class, a new robot class can be defined that shares the attributes and the member functions of the parent class. Each robot class brings its kinematic and dynamic properties (e.g., axes of rotation, link lengths, masses, etc.). Note that all member functions also accept symbolic arguments. This feature is helpful for control methods that require an analytical formulation of the robot's equations of motion, e.g., adaptive control methods [16].

While our software supports various robots, we show here a Franka robot example (`main_franka.m`), which is inherited from the `RobotPrimitives`-class. The initialization is shown below:

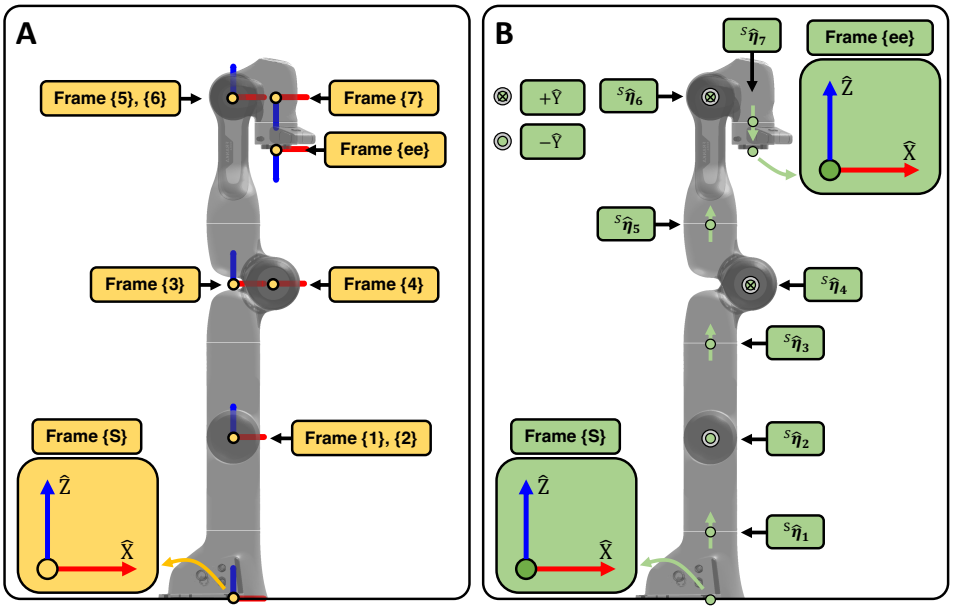


Fig. 3: Franka robot at initial configuration. The DH-convention is shown in (A) and the geometric method in (B). Only two frames are required for the geometric method (B). The frames shown in (A) are derived from [14].

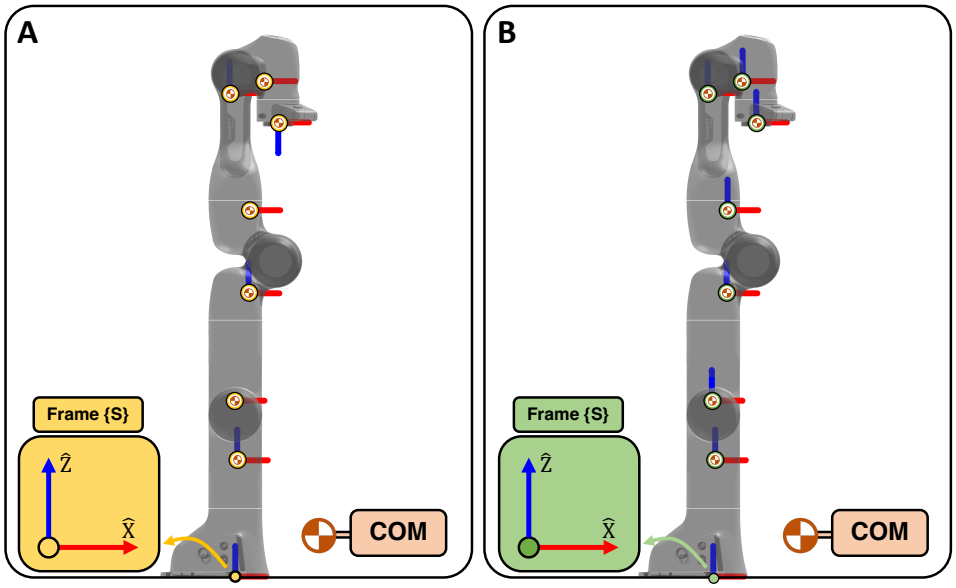


Fig. 4: COM-locations for each link of a Franka robot at initial configuration. The COM-locations are taken from [15].

```
% Call Franka Robot
robot = franka( );
robot.init( );
```

The `init`-function initializes all Joint Twists and Generalized Mass Matrices for the initial configuration (fig. 3).

For visualization, the robot object can be passed to a 2D or 3D-animation object:

```
% Create animation
anim = Animation('Dimension', 3, 'xLim',
    [-0.7,0.7], 'yLim', [-0.7,0.7], 'zLim'
    , [0,1.4]);
anim.init( );
anim.attachRobot( robot )
```

The `Animation`-class heavily relies on MATLAB graphic functions (e.g., axes, patches, lighting). The key to our animation is to create a chain of transform objects (`hgtransforms`) instead of transforming vertices. The `Animation`-class has an optional input that allows the recording of videos with adjustable playback speeds.

At run-time (simulation time `t`), the robot object (in configuration `q`) and the animation can be updated:

```
% Update kinematics
robot.updateKinematics( q );
anim.update( t );
```

*2) Example simulation:* For first-time users, we provide a simple example simulation. By default, the simulation loop is set to be real-time:

```
% Cyclic code starts here
while t <= simTime
    tic

    % YOUR CONTROLLER

    % Do not go faster than real time
    while toc < dt
        % do nothing
    end
end
```

The structure of the example application is the following: (1) calculation of all kinematic and dynamic robot parameters; (2) trajectory generation; (3) control law; (4) integration and update. For (1), the member functions of the robot object are used. While (2) and (3) are generally user specific, we implemented a simple impedance controller [17] for the Franka robot (`main_franka.m`) that moves the end-effector around a circular trajectory (fig. 5). For the integration (4), any integrator can be used, e.g., MATLAB's pre-built `ode45.m`.

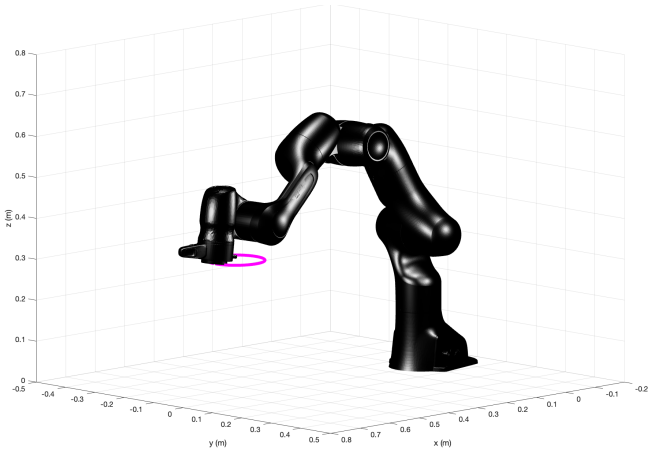


Fig. 5: Simulation of a simple impedance controller, using a Franka robot.

*3) Comparison with MATLAB robotic toolbox:* We compared the computational speed of Exp[licit] with the RVC MATLAB software [11], which uses the DH-convention. For RVC, version RTB10+MVTB4 (2017) was used.[5] By using native MATLAB scripts, the computation time was compared for the Forward Kinematic Map, Hybrid Jacobian, and Mass Matrix of an $n$-DOF open-chain planar robot. The robot consists of $n$ identical and uniform-mass bars with length $l = 1$m and mass $m = 1$kg.

For the RVC software, the robot was constructed from the `SerialLink`-class which consists of $n$ `Revolute`-classes. For Exp[licit], the robot was constructed from the `SnakeBot`-class. Robots with various DOF were constructed and tested. The test was performed with a MacBook air (M1 Chip, 16GB Memory), using MATLAB 2022a. The `timeit()` function was used to measure the computation time.

The results of our computational comparisons are shown in Figure 6. For all three tasks, Exp[licit] was faster than the RVC software. For both software, the computation of the Forward Kinematic Map and the Hybrid Jacobian showed a linear trend. The RVC software was capable of computing the Forward Kinematic Map of a 15-DOF robot within 1ms, whereas Exp[licit] required less than 1ms for a 100-DOF robot. For the Hybrid Jacobian, the RVC software required 5ms for a 20-DOF robot, while Exp[licit] could accomplish the same for robots with up to 100-DOF. The computation of the Mass Matrix showed an exponential trend for both software. RVC took $\sim$3ms to compute the Mass Matrix for a 2-DOF robot, whereas Exp[licit] required $\sim$1ms for a 6-DOF robot.

These results highlight the computational advantages of the geometric approach, as already discussed in [18].

## IV. Summary and Conclusion

This paper summarizes traditional and geometric methods for deriving the kinematic and dynamic properties of an open-chain robot. We highlight the conceptual and practical differences between the two approaches. While the geometric method demands a more abstract perspective (i.e., actions on manifolds), we showed several advantages compared to traditional methods. In summary, the advantages of the geometric method are:

- Highly modular structure, since Joint Twists can be reused throughout the calculation
- Flexibility to express kinematic and dynamic relations without predefined rules and exceptions
- No more than two frames to describe robot kinematics and dynamics.

We introduce Exp[licit], a MATLAB-based software which implements the geometric method and leverages its advantages. Thanks to the computational advantages and highly modular structure, we believe that this software can support various robotic applications. We hope to show that differential geometric methods are not limited to their conceptual strengths but can be useful for practical implementations.

## V. Discussion and Future Work

Exp[licit] also offers functions to calculate the gravitational co-vector and centrifugal/Coriolis matrix of the robot. At the time of writing, the implementation was based on a closed-form algorithm and not specific to a differential geometric method. This is the reason why we did not include those components in the paper. RVC also has functions to compute gravity and centrifugal/Coriolis effects via recursive Newton-Euler methods (RNE). When compared, we found that the RNE algorithm outperformed Exp[licit] for those components. Future work will improve the computation time of these terms within Exp[licit], e.g., by implementing differential geometric methods with a closed-form algorithm [6] or by using recursive algorithms.

[5]The software can be downloaded at https://petercorke.com/toolboxes/robotics-toolbox/

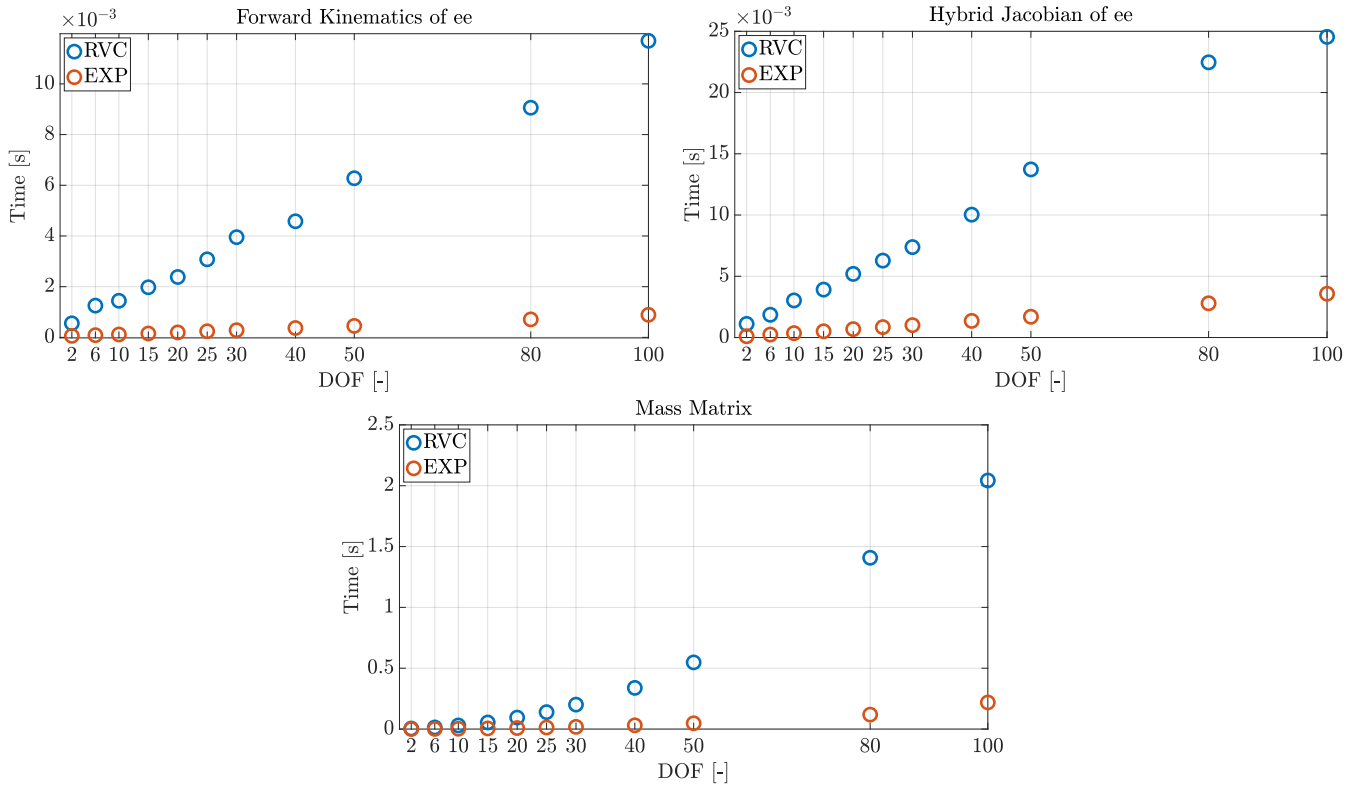


Fig. 6: Computation time of Exp[licit] and RVC for three tasks. 1) Forward Kinematic Map for end-effector ee; 2) Hybrid Jacobian for ee; 3) Robot Mass Matrix.

The current version of Exp[licit] uses `.m`-MATLAB scripts. To make a fair computational comparison, `.m`-MATLAB scripts were also used for the RVC-method. We discovered, that the RVC software provides an option to invoke `MEX`-files to improve the computation speed. `MEX`-files are native `C` or `C++` files that are dynamically linked to the MATLAB application at runtime. We discovered that for the Mass Matrix calculation, the `MEX`-file option of RVC showed better results than our Exp[licit] software. In a future version, we will implement `MEX`-files to further improve computational efficiency.

So far, the purpose of our software is to simulate different 2D and 3D robots using MATLAB. In future, Exp[licit] will offer a `C++` and `Python` option that can be used for real-time control of robots, e.g., for torque control with the KUKA LBR iiwa. At that point, it will be necessary to compare our methods with [19] which is also a library implemented in `C++`.

At the moment, Exp[licit] is limited to supporting open-chain robot structures. In the future, we are exploring the possibility of incorporating branched structures such as robotic hands, as well as closed-loop structures like delta robots.

## REFERENCES


[1] J. J. Craig, *Introduction to robotics : mechanics & control / John J. Craig*. Reading, Mass.: Addison-Wesley Pub. Co.,, 1986.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010.

[3] J. Lachner, "A geometric approach to robotic manipulation in physical human-robot interaction," Ph.D. dissertation, University of Twente, Netherlands, Jul. 2022.

[4] F. Park, "Computational aspects of the product-of-exponentials formula for robot kinematics," *IEEE transactions on automatic control.*, vol. 39, no. 3, pp. 643–647, 1994.

[5] R. Murray, Z. Li, S. Sastry, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Taylor & Francis, 1994.

[6] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[7] A. Mueller, "An o(n)-algorithm for the higher-order kinematics and inverse dynamics of serial manipulators using spatial representation of twists," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 397–404, 2021.

[8] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. ASME E, Journal of Applied Mechanics*, vol. 22, pp. 215–221, June 1955.

[9] J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms (Mechanical Engineering Series)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[10] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[11] P. Corke and O. Khatib, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 73.

[12] R. W. Brockett, *Robotic manipulators and the product of exponentials formula*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984.

[13] S. Stramigioli, *Modeling and IPC control of interactive mechanical systems—A coordinate-free approach*. Springer, 2001.

[14] F. Emika, "Robot and interface specifications," https://frankaemika.github.io/docs/control_parameters.html, 2027, [Online; accessed 12-April-2023].

[15] F. D. Ledezma and S. Haddadin, "Ril: Riemannian incremental learning of the inertial properties of the robot body schema," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9354–9360.

[16] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *The international journal of robotics research*, vol. 6, no. 3, pp. 49–59, 1987.

[17] N. Hogan, "Impedance control (an approach to manipulation) part i, ii,

iii," *Trans the ASME, J. of Dynamic systems, Measurement and Control*, vol. 107, pp. 1–24, 1985.

[18] F. C. Park, "Computational aspects of the product-of-exponentials formula for robot kinematics," *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 643–647, 1994.

[19] M. L. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, pp. 1–17, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10514-016-9574-0

## APPENDIX A
## PARAMETERS FOR THE FORWARD KINEMATIC MAP OF FRANKA ROBOT.

Parameters needed to derive the Forward Kinematic Map of a Franka robot for both methods. The four DH-parameters can be found in [14]. The corresponding frames are shown in fig. 3.

TABLE A1

| **Traditional Approach** | | | | |
|---|---|---|---|---|
| | $a$ (m) | $d$ (m) | $\alpha$ (rad) | $\theta$ (rad) |
| ${}^{S}\boldsymbol{H}_1$ | 0.0 | 0.333 | 0 | $q_1$ |
| ${}^{1}\boldsymbol{H}_2$ | 0.0 | 0.000 | $-\pi/2$ | $q_2$ |
| ${}^{2}\boldsymbol{H}_3$ | 0.0 | 0.316 | $\pi/2$ | $q_3$ |
| ${}^{3}\boldsymbol{H}_4$ | 0.0825 | 0.000 | $\pi/2$ | $q_4$ |
| ${}^{4}\boldsymbol{H}_5$ | $-0.0825$ | 0.384 | $-\pi/2$ | $q_5$ |
| ${}^{5}\boldsymbol{H}_6$ | 0.0 | 0.000 | $\pi/2$ | $q_6$ |
| ${}^{6}\boldsymbol{H}_7$ | 0.088 | 0.000 | $\pi/2$ | $q_7$ |
| ${}^{7}\boldsymbol{H}_{ee}$ | 0.0 | 0.107 | 0 | 0 |
| **Geometric Approach** | | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_1$ | (0, 0, 0, 0, 0, 1) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_2$ | (0.333, 0, 0, 0, −1, 0) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_3$ | (0, 0, 0, 0, 0, 1) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_4$ | (−0.649, 0, 0.0825, 0, 1, 0) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_5$ | (0, 0, 0, 0, 0, 1) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_6$ | (−1.033, 0, 0, 0, 1, 0) | | | |
| ${}^{S}\widehat{\boldsymbol{\eta}}_7$ | (0, 0.088, 0, 0, 0, −1) | | | |